Laurea Magristrale in Ingegneria dell'Ambiente e dell'Energia

Corso: Environmental Transport Phenomena

Docente: Cristian Marchioli

SIMULAZIONE NUMERICA DI UNO SHEAR LAYER TURBOLENTO

1 Introduzione

L'obiettivo e' quello di analizzare, tramite simulazioni numeriche, il mescolamento di particelle inerziali prodotto da uno shear layer turbolento. Per shear layer si intende la regione di flusso localizzata fra due strati di fluido adiacenti fra i quali esiste una certa differenza di velocità. Tale differenza di velocità determina l'insorgere, all'interno del flusso, di una instabilità idrodinamica nota come *instabilità di Kelvin-Helmholtz*. L'instabilità di Kelvin-Helmholtz é comune a molti fenomeni ambientali, osservabili sia internamente ai grandi corpi idrici (un noto esempio sono le correnti di gravità, nelle quali l'esistenza di una regione di mescolamento turbolento é associata alla formazione di strutture vorticose dette billows, prodotte appunto dall'instabilità di Kelvin-Helmholtz) oppure nell'atmosfera (tipicamente in presenza di correnti d'aria che scorrono una sopra all'altra con verso differente, ovvero una a riposo al livello del suolo e una in movimento sopra di essa, con relativa formazione di nubi in corrispondenza dei vortici che evidenzia la forma dell'instabilità). Tali fenomeni sono di primaria importanza per processi ambientali quali la deposizione di sedimenti sul fondo di laghi, mari e/o oceani oppure la dispersione di inquinati pesanti in atmosfera.

Per studiare il mescolamento si puó utilizzare un approccio numerico di tipo Eulerian-Lagrangiano in cui la fase continua (costituita dal fluido) é descritta dalle equazioni di Continuità e di Navier-Stokes, mentre la fase dispersa (costituita dalle particelle inerziali) é descritta da una equazione di moto che deriva da un bilancio delle forze agenti sulla particella durante il suo moto all'interno del fluido.

Nel seguito, vengono descritti (1) il codice di calcolo, scritto in linguaggio fortran, utilizzato per integrate nel tempo le equazioni di governo per il fluido (con riferimento ad un flusso turbolento bi-dimensionale in coordinate Cartesiane); e (2) il codice di calcolo, sempre in fortran, utilizzato per integrare nel tempo le equazioni di governo per le particelle.

2 Metodologia

2.1 Equazioni di governo del fluido

Le equazioni che governano il moto di un fluido viscoso incomprimibile sono la Continuità e l'equazione di Navier-Stokes. Dall'equazione di Continuità si ha:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{1}$$

dove u è la componente di velocità lungo l'asse orizzontale x, v lungo l'asse verticale y, w lungo l'asse trasversale z. L'equazione di Navier-Stokes in direzione x é:

$$\frac{\partial u}{\partial t}=-\frac{\partial uu}{\partial x}-\frac{\partial vu}{\partial y}-\frac{\partial wu}{\partial z}-\frac{1}{\rho}\frac{\partial P}{\partial x}+\frac{\partial}{\partial x}\left[2\nu\frac{\partial u}{\partial x}\right]+\frac{\partial}{\partial y}\left[\nu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[\nu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]-g_x \tag{2}$$

nella quale ν è la viscosità cinematica, P la pressione equivalente e g_x una forza volumetrica nella direzione x (tipicamente la gravità). L'equazione di Navier-Stokes in direzione y é:

$$\frac{\partial v}{\partial t} = -\frac{\partial uv}{\partial x} - \frac{\partial vv}{\partial y} - \frac{\partial wv}{\partial z} - \frac{1}{\rho}\frac{\partial P}{\partial y} + \frac{\partial}{\partial x}\left[\nu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[2\nu\frac{\partial v}{\partial y}\right] + \frac{\partial}{\partial z}\left[\nu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right] - g_y \quad (3)$$

L'equazione di Navier-Stokes in direzione z:

$$\frac{\partial w}{\partial t} = -\frac{\partial uw}{\partial x} - \frac{\partial vw}{\partial y} - \frac{\partial ww}{\partial z} - \frac{1}{\rho}\frac{\partial P}{\partial w} + \frac{\partial}{\partial x}\left[\nu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right] + \frac{\partial}{\partial y}\left[\nu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)\right] + \frac{\partial}{\partial z}\left[2\nu\frac{\partial w}{\partial z}\right] - g_z \quad (4)$$

Per convenienza le equazioni di Navier-Stokes saranno scritte così:

$$\frac{\partial u}{\partial t} = -ADV - \nabla P + DIFF - G. \quad (5)$$

dove ADV indica i termini convettivi, DIFF quelli diffusivi, P la pressione e G l'accelerazione di gravità. Nel modello numerico la turbolenza è simulata tramite la viscosità turbolenta di Smagorinsky (Smagorinsky Eddy Viscosity):

$$\nu_t = l_{mix}^2\sqrt{\frac{1}{2}S_{i,j}S_{i,j}} \quad (6)$$

Nella quale $S_{i,j} = S_{j,i}$ è il generico elemento del tensore dei gradienti di velocità, definito come:

$$\mathcal{S} = \begin{pmatrix} 2\frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & 2\frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} & 2\frac{\partial w}{\partial z} \end{pmatrix}$$

La lunghezza di mescolamento l_{mix} è ricavata in funzione della spaziatura della griglia Δ mediante la seguente formula: $l_{mix}= C_s\Delta$, nella quale $C_s = 0.1$ è la costante di Smagorinsky. Di seguito indicheremo la somma della viscosità turbolenta e la viscosità cinematica come segue:

$$\nu = \nu_{mol} + \nu_t$$

2.2 Discretizzazione del dominio fluido

Per poter essere integrate, le equazioni di continuità e conservazione della quantità di moto devono venire discretizzate rispetto alla griglia computazione che costituisce il dominio di calcolo. Detto $P_{i,j,k}$ il centro geometrico della generica cella della griglia, l'equazione di Continuità discretizzata attorno al punto $P_{i,j,k}$ diviene:

$$\frac{\partial u}{\partial x} = \frac{1}{dx}(U_{i,j,k} - U_{i-1,j,k}); \quad \frac{\partial v}{\partial y} = \frac{1}{dy}(U_{i,j,k} - U_{i,j-1,k}); \quad \frac{\partial w}{\partial z} = \frac{1}{dz}(U_{i,j,k} - U_{i,j,k-1}) \quad (7)$$

Dall'equazione di continuità si ha quindi:

$$\nabla(u,v,w) = \frac{1}{dx}(U_{i,j,k} - U_{i-1,j,k}) + \frac{1}{dy}(U_{i,j,k} - U_{i,j-1,k}) + \frac{1}{dz}(U_{i,j,k} - U_{i,j,k-1}). \quad (8)$$

Possiamo ricavare l'ultima equazione anche in una altro modo: considerando il volume di controllo dato in Figura 4, dal bilancio di massa su di esso si ottiene:

$$-\rho W_{i,j,k}dxdy + \rho W_{i,j,k-1}dxdy - \rho V_{i,j,k}dxdz + \rho V_{i,j-1,k}dxdz - \rho U_{i,j,k}dydz + \rho U_{i-1,j,k}dydz = 0 \quad (9)$$

Dividendo l'equazione per $\rho dxdydz$ si riottiene esattamente l'equazione di partenza (8).

L'equazione di Navier-Stokes in direzione x sarà approssimata nel seguente modo:

$$\frac{\partial uu}{\partial x} = \frac{1}{dx}\left[\left(\frac{U_{i+1,j,k}+U_{i,j,k}}{2}\right)^2 - \left(\frac{U_{i,j,k}+U_{i-1,j,k}}{2}\right)^2\right] \tag{10}$$

$$\frac{\partial uv}{\partial y} = 0.25\frac{1}{dy}[(U_{i,j,k}+U_{i,j+1,k})(V_{i,j,k}+V_{i+1,j,k})] - [(U_{i,j,k}+U_{i,j-1,k})(V_{i,j-1,k}+V_{i+1,j-1,k})] \tag{11}$$

$$\frac{\partial uw}{\partial z} = 0.25\frac{1}{dz}[(U_{i,j,k}+U_{i,j,k+1})(W_{i,j,k}+W_{i+1,j,k})] - [(U_{i,j,k}+U_{i,j,k-1})(W_{i,j,k-1}+W_{i+1,j,k-1})] \tag{12}$$

$$\frac{\partial p}{\partial x} = \frac{1}{dx}(P_{i+1,j,k} - P_{i,j,k}) \tag{13}$$

$$\frac{\partial}{\partial x}\left[2\nu\frac{\partial u}{\partial x}\right] = \frac{1}{dx}\left[2\nu\left(\frac{U_{i+1,j,k}-U_{i,j,k}}{dx}\right) - 2\nu\left(\frac{U_{i,j,k}-U_{i-1,j,k}}{dx}\right)\right] \tag{14}$$

$$\frac{\partial}{\partial y}\left[\nu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right] = \frac{1}{dy}\left[\nu\left(\frac{U_{i,j+1,k}-U_{i,j,k}}{dy}+\frac{V_{i+1,j,k}-V_{i,j,k}}{dx}\right)-\nu\left(\frac{U_{i,j,k}-U_{i,j-1,k}}{dy}+\frac{V_{i+1,j-1,k}-V_{i,j-1,k}}{dx}\right)\right] \tag{15}$$

$$\frac{\partial}{\partial z}\left[\nu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right] = \frac{1}{dz}\left[\nu\left(\frac{U_{i,j,k+1}-U_{i,j,k}}{dz}+\frac{W_{i+1,j,k}-W_{i,j,k}}{dx}\right)-\nu\left(\frac{U_{i,j,k}-U_{i,j,k-1}}{dy}+\frac{W_{i+1,j,k-1}-W_{i,j,k-1}}{dx}\right)\right] \tag{16}$$

nelle quali $\nu = \nu(x, y, z)$ è funzione dello coordinate spaziali *(x,y,z)*.
L'equazione di Navier-Stokes in direzione y viene approsimata nel seguente modo:

$$\frac{\partial vv}{\partial x} = \frac{1}{dx}\left[\left(\frac{V_{i,j+1,k}+V_{i,j,k}}{2}\right)^2 - \left(\frac{V_{i,j,k}+V_{i,j-1,k}}{2}\right)^2\right] \tag{17}$$

$$\frac{\partial uv}{\partial x} = 0.25\frac{1}{dx}[(U_{i,j,k}+U_{i,j+1,k})(V_{i,j,k}+V_{i+1,j,k})] - [(U_{i-1,j,k}+U_{i-1,j+1,k})(V_{i,j,k}+V_{i-1,j,k})] \tag{18}$$

$$\frac{\partial wv}{\partial x} = 0.25\frac{1}{dx}[(V_{i,j,k}+V_{i,j,k+1})(W_{i,j,k}+W_{i,j+1,k})] - [(V_{i,j,k}+V_{i,j,k-1})(W_{i,j,k-1}+W_{i,j+1,k-1})] \tag{19}$$

$$\frac{\partial p}{\partial y} = \frac{1}{dx}(P_{i,j+1,k} - P_{i,j,k}) \tag{20}$$

$$\frac{\partial}{\partial y}\left[2\nu\frac{\partial v}{\partial y}\right] = \frac{1}{dy}\left[2\nu\left(\frac{V_{i,j+1,k}-V_{i,j,k}}{dy}\right) - 2\nu\left(\frac{V_{i,j,k}-V_{i,j-1,k}}{dy}\right)\right] \tag{21}$$

$$\frac{\partial}{\partial x}\left[\nu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right] = \frac{1}{dx}\left[\nu\left(\frac{U_{i,j+1,k}-U_{i,j,k}}{dx}+\frac{V_{i+1,j,k}-V_{i,j,k}}{dx}\right)-\nu\left(\frac{U_{i-1,j+1,k}-U_{i-1,j,k}}{dx}+\frac{V_{i,j,k}-V_{i-1,j,k}}{dx}\right)\right] \tag{22}$$

$$\frac{\partial}{\partial z}\left[\nu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right] = \frac{1}{dz}\left[\nu\left(\frac{V_{i,j,k+1}-V_{i,j,k}}{dz}+\frac{W_{i,j+1,k}-W_{i,j,k}}{dy}\right)-\nu\left(\frac{V_{i,j,k}-V_{i-1,j,k}}{dx}+\frac{W_{i,j+1,k-1}-W_{i,j,k-1}}{dx}\right)\right] \tag{23}$$

L'equazione di *Navier-Stokes* in direzione z sarà discretizzata nel seguente modo:

$$\frac{\partial ww}{\partial z} = \frac{1}{dz}\left[\left(\frac{W_{i,j,k+1}+W_{i,j,k}}{2}\right)^2 - \left(\frac{W_{i,j,k}+W_{i,j,k-1}}{2}\right)^2\right] \tag{24}$$

$$\frac{\partial uw}{\partial x} = \frac{0.25}{dx}[(U_{i,j,k}+U_{i,j,k+1})(W_{i,j,k}+W_{i+1,j,k})]-[(U_{i-1,j,k}+U_{i-1,j,k+1})(W_{i,j,k}+W_{i-1,j,k})] \quad (25)$$

$$\frac{\partial vw}{\partial y} = \frac{0.25}{dy}[(V_{i,j,k}+V_{i,j,k+1})(W_{i,j,k}+W_{i,j+1,k})]-[(V_{i,j-1,k}+V_{i,j-1,k+1})(W_{i,j,k}+W_{i,j-1,k})] \quad (26)$$

$$\frac{\partial p}{\partial z} = \frac{1}{dz}(P_{i,j,k+1} - P_{i,j,k}) \quad (27)$$

$$\frac{\partial}{\partial z}\left[2\nu\frac{\partial w}{\partial z}\right] = \frac{1}{dz}\left[2\nu\left(\frac{W_{i,j,k+1} - W_{i,j,k}}{dz}) - 2\nu\left(\frac{W_{i,j,k} - W_{i-1,j,k}}{dz}\right)\right]\right. \quad (28)$$

$$\frac{\partial}{\partial x}\left[\nu\left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)\right] = \frac{1}{dx}\left[\nu\left(\frac{W_{i+1,j,k} - W_{i,j,k}}{dx}+\frac{U_{i,j,k+1} - U_{i,j,k}}{dz}\right)-\nu\left(\frac{W_{i,j,k} - W_{i-1,j,k}}{dx}+\frac{U_{i-1,j,k+1} - U_{i-1,j,k}}{dz}\right)\right] \quad (29)$$

$$\frac{\partial}{\partial y}\left[\nu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right] = \frac{1}{dy}\left[\nu\left(\frac{W_{i,j+1,k} - W_{i,j,k}}{dy}+\frac{V_{i,j,k+1} - V_{i,j,k}}{dz}\right)-\nu\left(\frac{W_{i,j,k} - W_{i,j-1,k}}{dy}+\frac{V_{i,j-1,k+1} - V_{i,j-1,k}}{dz}\right)\right] \quad (30)$$

La discretizzazione di tutti i termini dell' equazione di Navier-Stokes è approssimata al secondo ordine nello spazio. La discretizzazione del modello di sottogriglia viene data in seguito.

2.3 Integrazione nel tempo

Esistono vari algoritmi, di tipo implicito o esplicito, per integrare nel tempo. Solitamente il massimo passo temporale (time-step) *dt* permesso è maggiore nei metodi impliciti rispetto a quelli espliciti. Comunque per le applicazioni LES/DNS 3D su larga scala gli schemi impliciti sono troppo dispendiosi a livello computazionale, invece quelli espliciti sebbene con uno *time step* piú piccolo sono piu veloci e maggiormente accurati. In questa simulazione si è scelto di utilizzare l'algoritmo esplicito di second'ordine di Adam-Bashfort che è facilmente ricavabile dallo sviluppo in serie di Taylor. Considerando la seguente equazione:

$$y^{n+1} - y^n = h(af^n + bf^{n-1}) \qquad con \qquad y' = f \quad (31)$$

L'espansione in serie di Taylor dà:

$$y^{n+1} = y(x+h) = y(x) + hf(x) + \frac{h^2}{2}f'(x) + 0(h^3) \quad (32)$$

$$y^n = y(x) \quad (33)$$

$$af^n = af(x) \quad (34)$$

$$bf^{n-1} = bf(x-h) = bf(x) - bhf'(x) + b\frac{h^2}{2}f''(x) + 0(h^3) \quad (35)$$

Sostituendo quindi l'espressione ottenuta dall'equazione (35) nell'equazione (31) ed eliminando i termini in piú si ottiene:

$$hf(x) + \frac{h^2}{2}f'(x) = h(a+b)f(x) - h^2bf'(x) + 0(h^3) \quad (36)$$

Il metodo è del'secondo ordine per $(y^{n+1} - y^n)/h$ quando $a + b = 1$ e $b = -0.5$ ottenendo così lo schema di second'ordine di *Adam-Bashfort*. Per l'equazione di Navier-Stokes questo metodo è utilizzato per la derivazione temporale della velocità ma non può essere utilizzato per la Pressione perchè non c'è la derivata temporale. Un modo possibile per evitare problemi d'integrazione del termine di pressione è il seguente:

integrare l'equazione di Navier-Stokes nel tempo senza i termini del gradiente di pressione e utilizzare poi l'equazione di continuità per trovare un equazione per la pressione. Così facendo ci troveremo con la seguente formula da risolvere:

$$u^{n+1} = u^n + \triangle t[1.5(-ADV + DIFF)^n - 0.5(-ADV + DIFF)^{n-1} - \nabla P^{n+1}] \tag{37}$$

Notare che la pressione è calcolata a $n+1$. Possiamo quindi riscrivere l'equazione di prima come:

$$u^{n+1} + \triangle t \nabla P^{n+1} = u^* = u^n + \triangle t[1.5(-ADV + DIFF)^n - 0.5(-ADV + DIFF)^{n-1}], \tag{38}$$

Dalla divergenza della Velocità si ottiene:

$$\triangle t \nabla^2 P^{n+1} = div(u^*) \tag{39}$$

Dove si è imposta nulla la divergenza del vettore Velocità a $n+1$.
La discrettizazione di questa equazione è molto semplice:

$$\frac{P_{i+1,j,k} - 2P_{i,j,k} + P_{i-1,j,k}}{dx^2} + \frac{P_{i,j+1,k} - 2P_{i,j,k} + P_{i,j-1,k}}{dy^2} + \frac{P_{i,j,k+1} - 2P_{i,j,k} + P_{i,j,k-1}}{dz^2} = \frac{1}{dt}\left(\frac{u^*_{i,j,k} - u^*_{i-1,j,k}}{dx} + \frac{v^*_{i,j,k} - v^*_{i,j-1,k}}{dy} + \frac{w^*_{i,j,k} - w^*_{i,j,k-1}}{dz}\right); \tag{40}$$

L'equazione di Poisson discretizzata (40) è risolta dal Pois3d nel pacchetto FISHPACK (vedi programma).

Ottenuta la Pressione si ottiene $u^{n+1} = u^* - \triangle t \nabla P_{n+1}$, cioè:

$$U_{i,j,k} = U^*_{i,j,k} - dt\left(\frac{P_{i+1,j,k} - P_{i,j,k}}{dx}\right) \tag{41}$$

$$V_{i,j,k} = V^*_{i,j,k} - dt\left(\frac{P_{i,j+1,k} - P_{i,j,k}}{dy}\right) \tag{42}$$

$$W_{i,j,k} = W^*_{i,j,k} - dt\left(\frac{P_{i,j,k+1} - P_{i,j,k}}{dz}\right). \tag{43}$$

2.4 Condizioni al contorno per il fluido

In fluidodinamica, la condizione di *non-slip* per fluido viscoso afferma che se limitato da una parete solida, il fluido avrà velocità zero sulla parete. Alla parete solida la condizione di Non-Scorrimento può essere implementata nel seguente modo:

$$W_{i,j,0} = 0 \tag{44}$$

$$V_{i,j,0} = -V_{i,j,1} \tag{45}$$

$$U_{i,j,0} = -U_{i,j,1} \tag{46}$$

Viceversa la condizione di libero scorrimento alla parete può essere implementata nel sequente modo:

$$W_{i,j,0} = 0 \tag{47}$$

$$V_{i,j,0} = V_{i,j,1} \tag{48}$$

$$U_{i,j,0} = U_{i,j,1} \tag{49}$$

2.4.1 Condizioni al contorno per la pressione

Teoricamente le condizioni al contorno della pressione sono già fissate dalle condizioni al contorno della velocità. Ad ogni modo il pacchetto FISHPACK necessita di condizioni al contorno per la pressione. Queste condizioni al contorno devono essere scelte in modo tale che non alterino le condizioni al contorno della velocità e perciò per la pressione devono essere utilizzate le condizioni al contorno di *Neumann* cioè specificano i valori che la derivata di una soluzione deve assumere sul contorno del dominio.

2.4.2 Condizioni al contorno periodiche

Un particolare tipo di condizioni al contorno che sono frequentemente utilizzate in LES/DNS sono le cosiddette condizioni al contorno periodiche. L'implementazione di queste condizioni al contorno è semplice, per esempio in direzione y:

$$U_{i,jmax+1,k} = U_{i,1,k} \tag{50}$$

$$V_{i,jmax+1,k} = V_{i,1,k} \tag{51}$$

$$W_{i,jmax,k} = W_{i,1,k} \tag{52}$$

$$U_{i,0,k} = U_{i,jmax,k} \tag{53}$$

$$V_{i,0,k} = V_{i,jmax,k} \tag{54}$$

$$W_{i,0,k} = W_{i,jmax,k} \tag{55}$$

In questo particolare caso anche le condizioni al contorno della pressione saranno periodiche.

2.5 Modello di Sottogriglia

In questa sezione si discretizza il modello di sottogriglia di Smagorinsky. La viscosità turbolenta è calcolata nel seguente modo:

$$\nu_t = (C_s\Delta)^2\sqrt{\frac{1}{2}S_{i,j}S_{i,j}} \tag{56}$$

nella quale si è posto $\Delta = (dx \cdot dy \cdot dz)^{\frac{1}{3}}$.
Dalla discretizzazione del tensore dei gradienti di velocità $S_{i,j}S_{i,j}$ attorno al punto $P_{i,j,k}$ si ha:

$$\begin{aligned}
S_{i,j}S_{i,j} = {} & 4\left(\frac{U_{i,j,k}-U_{i-1,j,k}}{dx}\right)^2 + 4\left(\frac{V_{i,j,k}-V_{i,j-1,k}}{dy}\right)^2 + 4\left(\frac{W_{i,j,k}-W_{i,j,k-1}}{dz}\right)^2 + \\
& 0.5\left[\left(\frac{U_{i,j+1,k}-U_{i,j,k}}{dy}+\frac{V_{i+1,j,k}-V_{i,j,k}}{dx}\right)^2+\left(\frac{U_{i,j,k}-U_{i,j-1,k}}{dy}+\frac{V_{i+1,j-1,k}-V_{i,j-1,k}}{dx}\right)^2+\right. \\
& \left.\left(\frac{U_{i-1,j+1,k}-U_{i-1,j,k}}{dy}+\frac{V_{i,j,k}-V_{i-1,j,k}}{dx}\right)^2+\left(\frac{U_{i-1,j,k}-U_{i-1,j-1,k}}{dy}+\frac{V_{i,j-1,k}-V_{i-1,j-1,k}}{dx}\right)^2\right]+ \\
& 0.5\left[\left(\frac{U_{i,j,k+1}-U_{i,j,k}}{dz}+\frac{W_{i+1,j,k}-W_{i,j,k}}{dx}\right)^2+\left(\frac{U_{i,j,k}-U_{i,j,k-1}}{dz}+\frac{W_{i+1,j,k-1}-W_{i,j,k-1}}{dx}\right)^2+\right. \\
& \left.\left(\frac{U_{i-1,j,k+1}-U_{i-1,j,k}}{dz}+\frac{W_{i,j,k}-W_{i-1,j,k}}{dx}\right)^2+\left(\frac{U_{i-1,j,k}-U_{i-1,j,k-1}}{dz}+\frac{W_{i,j,k-1}-W_{i-1,j,k-1}}{dx}\right)^2\right]+ \\
& 0.5\left[\left(\frac{V_{i,j,k+1}-V_{i,j,k}}{dz}+\frac{W_{i,j+1,k}-W_{i,j,k}}{dy}\right)^2+\left(\frac{V_{i,j,k}-V_{i,j,k-1}}{dz}+\frac{W_{i,j+1,k-1}-W_{i,j,k-1}}{dy}\right)^2+\right. \\
& \left.\left(\frac{V_{i,j-1,k+1}-V_{i,j-1,k}}{dz}+\frac{W_{i,j,k}-W_{i,j-1,k}}{dy}\right)^2+\left(\frac{V_{i,j-1,k}-V_{i,j-1,k-1}}{dz}+\frac{W_{i,j,k-1}-W_{i,j-1,k-1}}{dy}\right)^2\right]
\end{aligned} \tag{57}$$

La viscosità può essere ora scritta come:

$$\nu = \nu_{molc} + (C_s\Delta)^2\sqrt{\frac{1}{2}S_{i,j}S_{i,j}} \tag{58}$$

Notare che la viscosità è calcolata attorno al punto di pressione mentre i termini viscosi sono ricavati ai lati delle celle.

2.6 Equazioni di governo delle particelle

Da completare.

3 Il Programma

Il programma in FORTRAN (*formula translation*) riceve dal file INPUT:

1-la velocità imperturbata del fluido U_∞
2-la lunghezza l_x
3-la lunghezza l_y
4-la lunghezza l_z
5-il numero di Reynolds del fluido
6-il numero di intervalli temporali da considerare

Le simulazioni sono state effettuate variando il numero di *Reynolds* del flusso oppure la velocità d'ingresso U_∞. Dal file input è stata impostata la lunghezza del dominio in direzione x a 4, in direzione y a 0.01 e in direzione z a 1. Quindi sono state ottenute le dimensioni delle celle della griglia:

$$\Delta x = \frac{L_x}{i_{max}}$$
$$\Delta y = \frac{L_y}{j_{max}}$$
$$\Delta z = \frac{L_z}{k_{max}}$$

nelle quali $i_{max}, j_{max}, k_{max}$ sono il numero dei punti della griglia forniti dal file PARAM.TXT. Terminata la simulazione, si ottiene in output il file di testo TECDATA.TXT che contiene i valori delle grandezze interessate, in ordine: posizione X, Y, velocità U, velocità W, pressione P e vorticità. I file di output sono scritti in modo da poter essere elaborati direttamente tramite il software Gnuplot. I due file di testo esterni sono:

PARAM.TXT: IMPOSTA I PUNTI DELLA GRIGLIA: $i_{max}, j_{max}, z_{max}$

```
integer imax,jmax,kmax,i,j,k
      integer i1  ,j1  ,k1
      parameter ( imax =256)
      parameter ( jmax =  4)
      parameter ( kmax =256)
      parameter ( i1 =imax+1)
      parameter ( j1 =jmax+1)
      parameter ( k1 =kmax+1)
```

COMMON.TXT: DEFINISCE IL DOMINIO DELLE VELOCITÀ DELL'ISTANTE PRECEDENTE, *Uold Vold e Wold* E ISTANTE PRESENTE, *Unew Vnew e Wnew*, PRESSIONE E VISCOSITÀ

```
realUold(0:i1,0:j1,0:k1),Vold(0:i1,0:j1,0:k1),Wold(0:i1,0:j1,0:k1)
      common /old/Uold,Vold,Wold

      realUnew(0:i1,0:j1,0:k1),Vnew(0:i1,0:j1,0:k1),Wnew(0:i1,0:j1,0:k1)
      common /new/Unew,Vnew,Wnew

      realdUdt(0:i1,0:j1,0:k1),dVdt(0:i1,0:j1,0:k1),dWdt(0:i1,0:j1,0:k1)
      common /deriv/dUdt,dVdt,dWdt

      real  p(imax,jmax,kmax)
```

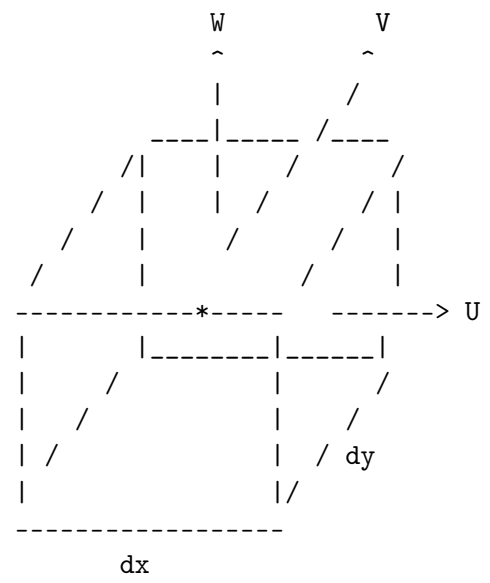
```
      common /pressure/p

      real dxi,dyi,dzi,Rei,dt,Ufree,cs
      common /parm/dxi,dyi,dzi,Rei,dt,Ufree,cs

      real   visc(0:i1,0:j1,0:k1)
      common /viscosity/visc
```

Il pacchetto **FISHPACK.F** ha il compito di integrare le equazioni Poisson (pois3d) per ottenere i termini della pressione.

3.1 Parametri e subroutine

Lista dei parametri:

```
imax : numero dei punti della griglia in direzione x = 256
jmax : numero dei punti della griglia in direzione y = 4
kmax : numero dei punti della griglia in direzione z = 256

lx   : Lunghezza del dominio di simulazione in direzione x, fornito dal file input = 4

ly   : Lunghezza del dominio di simulazione in direzione y, fornito dal file input = 0.01

lz   : Lunghezza del dominio di simulazione in direzione y, fornito dal file input = 1

Cs  : Costante di Smagorinsky impostata a 0.1

     dxi= 1/dx= imax / lx
     dyi= 1/dy= jmax / ly
     dzi= 1/dz= kmax / lx

Rei   : 1 / Re  con Re = numero di Reynolds

nstap : numero degli timesteps

Uold(0:imax+1,0:jmax+1,0:kmax+1) Velocità in direzione x al timestep precedente

Vold(0:imax+1,0:jmax+1,0:kmax+1) Velocità in direzione y al timestep precedente

Wold(0:imax+1,0:jmax+1,0:kmax+1) Velocità in direzione z al timestep precedente

Unew(.....)  Velocità al timestep presente in direzione x
Vnew(.....)  Velocità al timestep presente in direzione y
Wnew(.....)  Velocità al timestep presente in direzione z

dUdt(.....),dVdt(...),dWdt(....) vVelocità prevista al nuovo timestep


P(imax,jmax,kmax) : Vettore della Pressione

visc(0:imax+1,0:jmax+1,0:kmax+1): Vettore della viscosità turbolenta prevista in LES
```




```
U : Velocità in direzione x    [ U(i,j,k) ]
V : Velocità in direzione y    [ V(i,j,k) ]
W : Velocità in direzione z    [ W(i,j,k) ]
* : Punto di pressione
dx,dy,dz: dimensioni delle celle della griglia
```

SUBROUTINE : il pacchetto shear-layer.f utilizza le seguente subroutines per computare le operazione descritte nella Metodologia: **SUBROUTINE INIT** : IMPOSTA U,V,W A UN CERTO VALORE INIZIALE

```
subroutine init
    include 'param.txt'
    include 'common.txt'
    real delta,z,x
    delta = 1/28.
    do k=1,kmax
       do j=1,jmax
         do i=1,imax
         Uold(i,j,k)=0.
         Vold(i,j,k)=0.
         Wold(i,j,k)=0.
         Unew(i,j,k)=Ufree*tanh(28.* ((1.*k/kmax) -.5  ))
         Uold(i,j,k)=Unew(i,j,k)
         Vnew(i,j,k)=0.
         Wnew(i,j,k)=0.
         dUdt(i,j,k)=0.
         dVdt(i,j,k)=0.
         dWdt(i,j,k)=0.
         enddo
       enddo
    enddo
    do k=1,kmax
```

```
        z= (1.*k/kmax)-0.5
do j=1,jmax
           do i=1,imax
           x = 2.*i/imax
     Unew(i,j,k)=Unew(i,j,k)+0.1*Ufree*(-z/delta)*
     &    exp( -(z/delta)**2)*cos(0.44/delta*x)
             Wnew(i,j,k)=Wnew(i,j,k)+0.1*Ufree*
     &    exp( -(z/delta)**2)*sin(0.44/delta*x)
           enddo
        enddo
       enddo
```

Subroutine chkdt : imposta il *timestep dt*, calcolato caso per caso

```
    subroutine chkdt
c
c     Calculates the timestep dt.
c
      implicit none
      include 'param.txt'
      include 'common.txt'
      real    Courant,dtemp,dtmax
      Courant  = .5
      dtmax = -9999.9999
      do k=1,kmax
         do j=1,jmax
            do i=1,imax
            dtemp =
     1      abs(Unew(i,j,k) * dxi) +
     2      abs(Vnew(i,j,k) * dyi) +
     3      abs(Wnew(i,j,k) * dzi) +
     4       visc(i,j,k) * (dxi*dxi + dyi*dyi + dzi*dzi)
            dtmax = max ( dtmax , dtemp )
            enddo
         enddo
       enddo
       dt = Courant / dtmax
       return
       end
```

subrouitne chkdiv: Calcolo della divergenza della velocità e ne controlla in valori

```
 subroutine chkdiv
c
c checks the discrete divergence (should be of order of machine prec.)
c
      implicit none
      include 'param.txt'
      include 'common.txt'
      real div,divmax,divtot
      divmax = -9999.9999
      divtot = 0.
      do k=1,kmax
         do j=1,jmax
```

```
            do i=1,imax
            div = (
     1   ( Wnew(i,j,k)-Wnew(i,j,k-1) ) * dzi +
     2   ( Vnew(i,j,k)-Vnew(i,j-1,k) ) * dyi +
     3   ( Unew(i,j,k)-Unew(i-1,j,k) ) * dxi )
           divtot = divtot + div
           divmax = max ( div,divmax)
c       write(6,*) i,j,k,div
           enddo
         enddo
       enddo
       write(6,111) divtot,divmax
111    FORMAT('Mass loss/gain : Tot =',e13.6,' Max = ',e13.6)
       if (divtot .gt. 10e-3) stop 'Fatal Error **DIVERGENCE TOO LARGE**'
       return
       end
```

SUBROUTINE MOMX : INTEGRA L'EQUAZIONE DELLA CONSERVAZIONE DELLA QUANTITÀ DI MOTO IN DIREZIONE X

```
       subroutine momx(dudt,u,v,w,dxi,dyi,dzi,ib,ie,jb,je,kb,ke,i1,j1,k1,
      $                visc)
       implicit none
       integer ib,ie,jb,je,kb,ke,i1,j1,k1
       integer ip,im,jp,jm,kp,km,i,j,k
       real dudt(0:i1,0:j1,0:k1),
      $     u   (0:i1,0:j1,0:k1),
      $     v   (0:i1,0:j1,0:k1),
      $     w   (0:i1,0:j1,0:k1),
      $    dxi,dyi,dzi,visc(0:i1,0:j1,0:k1)
       real uuip ,uuim ,uvjp ,uvjm ,uwkp ,uwkm,vzp,vzm,vyp,vym
       real du1ip,du1im,du1jp,du1jm,du1kp,du1km
       do k=kb,ke
         do j=jb,je
           do i=ib,ie
*
           ip = i + 1
           jp = j + 1
           kp = k + 1
           im = i - 1
           jm = j - 1
           km = k - 1
           uuip  = 0.25 * ( U(ip,j,k)+U(i,j,k) )*( U(ip,j,k)+U(i,j,k)  )
           uuim  = 0.25 * ( U(im,j,k)+U(i,j,k) )*( U(im,j,k)+U(i,j,k)  )
           uvjp  = 0.25 * ( U(i,jp,k)+U(i,j,k) )*( V(ip,j,k)+V(i,j,k)  )
           uvjm  = 0.25 * ( U(i,jm,k)+U(i,j,k) )*( V(ip,jm,k)+V(i,jm,k))
           uwkp  = 0.25 * ( U(i,j,kp)+U(i,j,k) )*( W(ip,j,k) +W(i,j,k) )
           uwkm  = 0.25 * ( U(i,j,km)+U(i,j,k) )*( W(ip,j,km)+W(i,j,km))
           du1ip = 2.*( dxi * (U(ip,j,k)-U(i ,j,k) ))
           du1im = 2.*( dxi * (U(i ,j,k)-U(im,j,k) ))
           du1jp = dyi*(U(i,jp,k)-U(i, j,k))+dxi*(V(ip,j, k)-V(i ,j ,k))
           du1jm = dyi*(U(i,j ,k)-U(i,jm,k))+dxi*(V(ip,jm,k)-V(i ,jm,k))
           du1kp = dzi*(U(i,j,kp)-U(i ,j,k))+dxi*(W(ip,j, k)-W(i ,j ,k))
           du1km = dzi*(U(i,j ,k)-U(i,j,km))+dxi*(W(ip,j,km)-W(i ,j,km))
```

```
      vzp =
     &   0.25*(visc(i,j,k)+visc(ip,j,k)+visc(ip,j,kp)+visc(i,j,kp))
      vzm =
     &   0.25*(visc(i,j,k)+visc(ip,j,k)+visc(ip,j,km)+visc(i,j,km))
      vyp =
     &   0.25*(visc(i,j,k)+visc(ip,j,k)+visc(ip,jp,k)+visc(i,jp,k))
      vym =
     &   0.25*(visc(i,j,k)+visc(ip,j,k)+visc(ip,jm,k)+visc(i,jm,k))

          dudt(i,j,k) =
     1   dxi*( -uuip + visc(ip,j,k)*du1ip+uuim - visc(i,j,k)*du1im )+
     2   dyi*( -uvjp + vyp*du1jp+uvjm - vym*du1jm )+
     3   dzi*( -uwkp + vzp*du1kp+uwkm - vzm*du1km )
            enddo
         enddo
      enddo
      return
      end
```

SUBROUTINE MOMY : INTEGRA L'EQUAZIONE DELLA CONSERVAZIONE DELLA QUANTITÀ DI MOTO IN DIREZIONE Y

```
  subroutinemomy(dvdt,u,v,w,dxi,dyi,dzi,ib,ie,jb,je,kb,ke,i1,j1,k1,
     $                visc)
      implicit none
      integer ib,ie,jb,je,kb,ke,i1,j1,k1
      integer im,ip,jm,jp,km,kp,i,j,k
      real dvdt(0:i1,0:j1,0:k1),
     $     u   (0:i1,0:j1,0:k1),
     $     v   (0:i1,0:j1,0:k1),
     $     w   (0:i1,0:j1,0:k1),
     $    dxi,dyi,dzi,visc(0:i1,0:j1,0:k1)
      real uvip ,uvim ,vvjp ,vvjm ,wvkp ,wvkm
      real dv1ip,dv1im,dv1jp,dv1jm,dv1kp,dv1km
      real vxp,vxm,vzp,vzm
      do k=kb,ke
        do j=jb,je
          do i=ib,ie
*
          ip = i + 1
          jp = j + 1
          kp = k + 1
          im = i - 1
          jm = j - 1
          km = k - 1
          uvip  = 0.25 * (U(i ,j,k)+U(i ,jp,k))*( V(i,j,k)+V(ip,j,k) )
          uvim  = 0.25 * (U(im,j,k)+U(im,jp,k))*( V(i,j,k)+V(im,j,k) )
          vvjp  = 0.25 * (V(i,j,k )+V(i,jp,k) )*( V(i,j,k)+V(i,jp,k) )
          vvjm  = 0.25 * (V(i,j,k )+V(i,jm,k) )*( V(i,j,k)+V(i,jm,k) )
          wvkp  = 0.25 * (W(i,j,k )+W(i,jp,k) )*( V(i,j,kp)+V(i,j,k) )
          wvkm  = 0.25 * (W(i,j,km)+W(i,jp,km))*( V(i,j,km)+V(i,j,k) )

          dv1ip = dxi*(V(ip,j,k)-V(i ,j,k))+dyi*(U(i,jp ,k)-U(i ,j,k))
          dv1im = dxi*(V(i ,j,k)-V(im,j,k))+dyi*(U(im,jp,k)-U(im,j,k))
```

```
          dv1jp = 2.*dyi*(V(i,jp,k)-V(i,j ,k))
          dv1jm = 2.*dyi*(V(i,j ,k)-V(i,jm,k))
          dv1kp = dzi*(V(i,j,kp)-V(i, j,k))+dyi*(W(i,jp,k )-W(i ,j,k))
          dv1km = dzi*(V(i,j,k )-V(i,j,km))+dyi*(W(i,jp,km)-W(i,j,km))
       vxp =
     & 0.25*(visc(i,j,k)+visc(ip,j,k)+visc(ip,jp,k)+visc(i,jp,k))
       vxm =
     & 0.25*(visc(i,j,k)+visc(im,j,k)+visc(im,jp,k)+visc(i,jp,k))
       vzp =
     & 0.25*(visc(i,j,k)+visc(i,jp,k)+visc(i,jp,kp)+visc(i,j,kp))
       vzm =
     & 0.25*(visc(i,j,k)+visc(i,jp,k)+visc(i,jp,km)+visc(i,j,km))

          dvdt(i,j,k)=
     1 dxi*((-uvip + vxp         *dv1ip)-(-uvim + vxm        *dv1im))+
     2 dyi*((-vvjp + visc(i,jp,k)*dv1jp)-(-vvjm + visc(i,j,k)*dv1jm))+
     3 dzi*((-wvkp + vzp         *dv1kp)-(-wvkm + vzm        *dv1km))
          enddo
        enddo
       enddo
       return
       end
```

SUBROUTINE MOMZ : INTEGRA L'EQUAZIONE DELLA CONSERVAZIONE DELLA QUANTITÀ DI MOTO IN DIREZIONE Z

```
      subroutinemomz(dwdt,u,v,w,dxi,dyi,dzi,ib,ie,jb,je,kb,ke,i1,j1,k1,
       $                visc)
        implicit none
        integer ib,ie,jb,je,kb,ke,i1,j1,k1
        integer im,jm,km,ip,jp,kp,i,j,k
        real dwdt(0:i1,0:j1,0:k1),
       $     u   (0:i1,0:j1,0:k1),
       $     v   (0:i1,0:j1,0:k1),
       $     w   (0:i1,0:j1,0:k1),
       $    dxi,dyi,dzi,visc(0:i1,0:j1,0:k1)
        real uwip ,uwim ,vwjp ,vwjm ,wwkp ,wwkm
        real dw1ip,dw1im,dw1jp,dw1jm,dw1kp,dw1km
        real  vxm,vym,vxp,vyp
        do k=kb,ke
          do j=jb,je
            do i=ib,ie
*
            ip = i + 1
            jp = j + 1
            kp = k + 1
            im = i - 1
            jm = j - 1
            km = k - 1
            uwip  = 0.25 * ( W(i,j,k)+W(ip,j,k))*(U(i ,j,k)+U(i ,j,kp) )
            uwim  = 0.25 * ( W(i,j,k)+W(im,j,k))*(U(im,j,k)+U(im,j,kp) )
            vwjp  = 0.25 * ( W(i,j,k)+W(i,jp,k))*(V(i ,j,k)+V(i,j ,kp) )
            vwjm  = 0.25 * ( W(i,j,k)+W(i,jm,k))*(V(i,jm,k)+V(i,jm,kp) )
            wwkp  = 0.25 * ( W(i,j,k)+W(i,j,kp))*(W(i ,j,k)+W(i,j,kp ) )
            wwkm  = 0.25 * ( W(i,j,k)+W(i,j,km))*(W(i ,j,k)+W(i,j,km ) )
```

```
          dw1ip = dxi*(W(ip,j,k)-W(i ,j,k))+dzi*(U(i ,j,kp)-U(i ,j,k))
          dw1im = dxi*(W(i ,j,k)-W(im,j,k))+dzi*(U(im,j,kp)-U(im,j,k))
          dw1jp = dyi*(W(i,jp,k)-W(i, j,k))+dzi*(V(i ,j,kp)-V(i ,j,k))
          dw1jm = dyi*(W(i,j ,k)-W(i,jm,k))+dzi*(V(i,jm,kp)-V(i,jm,k))
          dw1kp = 2*dzi*(W(i,j,kp)-W(i,j ,k))
          dw1km = 2*dzi*(W(i,j,k )-W(i,j,km))
          vxp =
     & 0.25*(visc(i,j,k)+visc(ip,j,k)+visc(ip,j,kp)+visc(i,j,kp))
          vxm =
     & 0.25*(visc(i,j,k)+visc(im,j,k)+visc(im,j,kp)+visc(i,j,kp))
          vyp =
     & 0.25*(visc(i,j,k)+visc(i,jp,k)+visc(i,jp,kp)+visc(i,j,kp))
          vym =
     & 0.25*(visc(i,j,k)+visc(i,jm,k)+visc(i,jm,kp)+visc(i,j,kp))
          dwdt(i,j,k) =
     1dxi*((-uwip + vxp         *dw1ip)-(-uwim + vxm        *dw1im))+
     2dyi*((-vwjp + vyp         *dw1jp)-(-vwjm + vym        *dw1jm))+
     3dzi*((-wwkp + visc(i,j,kp)*dw1kp)-(-wwkm + visc(i,j,k)*dw1km))
            enddo
           enddo
       enddo
       return
       end
```

SUBROUTINE ADAM : INTEGRA L'EQUAZIONE DELLA CONSERVAZIONE DELLA QUANTITÀ DI MOTO NEL TEMPO

```
 subroutine adamsb(string)
C
C
C  Integrate in time with second order Adams-Bashfort (or Euler forward)
C  dold(...),dnew(....) are dummy arrays
C
      implicit none
      include 'param.txt'
      include 'common.txt'
      real   dold(0:i1,0:j1,0:k1),dnew(0:i1,0:j1,0:k1)
      real   cof1,cof2
      character*5 string
*     Adams-Bashfort  cof1=1.5  , cof2 = -0.5
*     Euler-Forward   cof1=  1  , cof2 =    0
      if (string .eq. 'euler') then
      cof1=1.
      cof2=0.
      endif
      if (string .eq. 'adams') then
      cof1= 1.5
      cof2=-0.5
      endif
c
c   integrate momentum equations at old and new timelevel
c
      call momx(dold,uold,vold,wold,dxi,dyi,dzi,
     &          1,imax,1,jmax,1,kmax,i1,j1,k1,visc)
```

```
      call momx(dnew,unew,vnew,wnew,dxi,dyi,dzi,
     &          1,imax,1,jmax,1,kmax,i1,j1,k1,visc)
c
c   calculate predicted velocity
c
      do k=1,kmax
        do j=1,jmax
          do i=1,imax
          dudt(i,j,k)=Unew(i,j,k) +
     1    dt * ( cof1 * dnew(i,j,k) + cof2 * dold(i,j,k) )
          enddo
        enddo
      enddo
      call momy(dold,uold,vold,wold,dxi,dyi,dzi,
     &          1,imax,1,jmax,1,kmax,i1,j1,k1,visc)
      call momy(dnew,unew,vnew,wnew,dxi,dyi,dzi,
     &          1,imax,1,jmax,1,kmax,i1,j1,k1,visc)

      do k=1,kmax
        do j=1,jmax
          do i=1,imax
          dvdt(i,j,k)=Vnew(i,j,k)+
     1    dt * ( cof1 * dnew(i,j,k) + cof2 * dold(i,j,k) )
          enddo
        enddo
      enddo
      call momz(dold,uold,vold,wold,dxi,dyi,dzi,
     &          1,imax,1,jmax,1,kmax,i1,j1,k1,visc)
      call momz(dnew,unew,vnew,wnew,dxi,dyi,dzi,
     &          1,imax,1,jmax,1,kmax,i1,j1,k1,visc)
      do k=1,kmax
        do j=1,jmax
          do i=1,imax
          dwdt(i,j,k)=Wnew(i,j,k)+
     1    dt * ( cof1 * dnew(i,j,k) + cof2 * dold(i,j,k) )
          enddo
        enddo
      enddo
      return
      end
```

SUBROUTINE FILLPS: FISSA IL SISTEMA DI RIFERIMENTO PER L'EQUAZIONE DI POISSON

```
   subroutine fillps
      implicit none
      include 'param.txt'
      include 'common.txt'
      real   dti
      dti =1./dt
c
c
c     fill right hand side of the poisson equation
c
c     discrete divergence is
c   w(i,j,k)-w(i,j,k-1)  v(i,j,k)-v(i,j-1,k)  u(i,j,k)-u(i-1,j,k)
c   ------------------  ------------------  ------------------  = div
c         d z                 d y                 d x
c
c  note that p is not yet the pressure
c  but only the rhs of the poisson equation
c
      do k=1,kmax
       do j=1,jmax
  do i=1,imax
      p(i,j,k)= dti*(
     1   ( dWdt(i,j,k)-dWdt(i,j,k-1))*dzi+
     2   ( dVdt(i,j,k)-dVdt(i,j-1,k))*dyi+
     3   ( dUdt(i,j,k)-dUdt(i-1,j,k))*dxi
     e              )
          enddo
       enddo
      enddo
      return
      end
```

SUBROUTINE CORREC: CORREGGE I VALORI DELLA DIVERGENZA AFFINCHÈ SODDISFINO L'EQUAZIONE DI CONTINUITÀ $div(u, v, w) == 0$

```
     subroutine correc
c
c
c   Corrects the velocity in such a way that div(u,v,w) is exactly zero
c
c
      implicit none
      include 'param.txt'
      include 'common.txt'
      do k=1,kmax
      do j=1,jmax
      do i=1,imax-1
       dudt(i,j,k)=
     1  dUdt(i,j,k) - dt * dxi * ( P(i+1,j,k)-P(i,j,k) )
      enddo
      enddo
      enddo
      do k=1,kmax
```

```
      do j=1,jmax
      dudt(imax,j,k)=
     1  dudt(imax,j,k) -dt*  dxi *(P(1,j,k)-P(imax,j,k))
      enddo
      enddo

      do k=1,kmax
      do j=1,jmax-1
      do i=1,imax
       dvdt(i,j,k)=
     1  dvdt(i,j,k) - dt*dyi * ( P(i,j+1,k)-P(i,j,k) )
      enddo
      enddo
      enddo
      do k=1,kmax
      do i=1,imax
      dvdt(i,jmax,k)=
     1  dvdt(i,jmax,k)-dt*dyi*(P(i,1,k)-P(i,jmax,k))
      enddo
      enddo
      do k=1,kmax-1
      do j=1,jmax
      do i=1,imax
       dwdt(i,j,k)=
     1 dwdt(i,j,k) - dt*dzi * ( P(i,j,k+1)-P(i,j,k) )
      enddo
      enddo
      enddo
c
c
c     Update the velocity fields
c
c
      do k=1,kmax
         do j=1,jmax
            do i=1,imax
            Uold(i,j,k)=Unew(i,j,k)
            Unew(i,j,k)=dUdt(i,j,k)
            Vold(i,j,k)=Vnew(i,j,k)
            Vnew(i,j,k)=dVdt(i,j,k)
            Wold(i,j,k)=Wnew(i,j,k)
            Wnew(i,j,k)=dWdt(i,j,k)
            enddo
          enddo
      enddo
      return
      end
```

SUBROUTINE BOUND : IMPOSTA LE CONDIZIONI AL CONTORNO

```
  subroutine bound(U,V,W)
      include 'param.txt'
      include 'common.txt'
      real U(0:i1,0:j1,0:k1),V(0:i1,0:j1,0:k1),
     $     W(0:i1,0:j1,0:k1)
c
c   boundary conditions at z=zmin and z =zmax
c
      do i=0,i1
        do j=0,j1
         W(i,j,0    ) =      0.
         W(i,j,kmax ) =      0.
         V(i,j,0    ) =  V(i,j,1   )
         V(i,j,k1   ) =  V(i,j,kmax)
         U(i,j,0    ) =  U(i,j,1   )
         U(i,j,k1   ) =  U(i,j,kmax)
        enddo
      enddo
c
c  periodic boundary conditions in the x direction
c
      do k=0,k1
        do j=0,j1
          U(0 ,j,k)=U(imax,j,k)
          V(0 ,j,k)=V(imax,j,k)
          W(0 ,j,k)=W(imax,j,k)
          U(i1,j,k)=U(1   ,j,k)
          V(i1,j,k)=V(1   ,j,k)
          W(i1,j,k)=W(1   ,j,k)
         enddo
       enddo
c
c periodic boundary conditions in the y-direction
c
       do k=0,k1
         do i=0,i1
           U(i,0,k )=U(i,jmax,k)
           V(i,0,k )=V(i,jmax,k)
           W(i,0,k )=W(i,jmax,k)
           U(i,j1,k)=U(i,1   ,k)
           V(i,j1,k)=V(i,1   ,k)
           W(i,j1,k)=W(i,1   ,k)
         enddo
       enddo
       return
       end
```

SUBROUTINE SOLVER: RICHIAMA LA ROUTINE POIS3D PER RISOLVERE L'EQUAZIONE DI *Poisson*

```
      subroutine solver(p,dxi,dyi,dzi)
      implicit none
      include 'param.txt'
```

```
integer  ier
real     p(imax,jmax,kmax)
real     dxi,dyi,dzi,dzi2,c1,c2,a(kmax),b(kmax),c(kmax)
real     rhs(kmax,jmax,imax),d(kmax)
real     save(k1*jmax*imax+3*kmax+3*jmax+4*imax+1000)
real     w(i1*j1*k1)
c1   = dxi*dxi
c2   = dyi*dyi
dzi2 = dzi*dzi
do k=1,kmax
a(k)=dzi2
c(k)=dzi2
b(k)=-(a(k)+c(k))
d(k)=c2
enddo
b(1)=b(1)+a(1)
a(1)=0.
b(kmax)=b(kmax)+c(kmax)
c(kmax)=0.
callpois3d(0,imax,c1,0,jmax,c2,1,kmax,a,b,c,imax,jmax,p,ier,w)
if (ier .ne. 0) write(6,*) ier
If (ier .ne. 0) stop 'Fatal Error **Poisson Solver**'
return
end
```

4 Sommario delle simulazioni da effettuare

Parametri per il fluido (da fornire in ingresso per poter effettuare la simulazione):

Simulazione	Velocità, U_∞	Numero di Reynolds	Numero di time step	Total mill time
1	25	10	5000	0.1006210
2	25	100	5000	0.3002787
3	25	1000	3000	0.2022880
4	25	10000	3000	0.1454996
5	1	1000	3000	3.847478
6	5	1000	3000	0.9983597
7	50	1000	3000	0.1353284
8	125	1000	3000	$3.0382756 \cdot 10^{-2}$

Tabella 1: Elenco delle simulazione dello shear-layer:le unità sono tutte adimensionali