CALC-LES: A Fortran Code for LES and Hybrid LES-RANS

Lars Davidson
Div.. of Fluid Dynamics
Dept. of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Göteborg, Sweden

Contents

1 Introduction

2 Geometrical Details of the Grid

2.1 Grid

The coordinates of the corners (X_C, Y_C, Z_C) of each control volume should be specified by the user, i.e. the grid must be generated by the user. The nodes of the control volume (X_P, Y_P, Z_P) are placed at the center of their control volumes. The control volume adjacent to the boundaries have two nodes, one in the center and one at the boundary, see Fig. 2.1. In any coordinate direction, lets say ξ, there are NI nodes, NI-1 control volume faces, and NI-2 control volumes. The nodes are numbered (from low ξ to high ξ) from 1 to NI , the control volume faces are numbered from 1 to NI-1, and the control volumes from 2 to NI-1. This is the same for η and ζ directions. Note that (ξ, η, ζ) must form a right-hand coordinate system.

CALC-LES employs curvilinear grids but the code can also be used with Cartesian as well as cylindrical coordinate systems.

2.1.1 Nomenclature for the Grid

A schematic control volume grid is shown in Fig. 2.2. Single capital letters define nodes [E(ast), S(outh), etc.], and single small letters define faces of the control volumes.

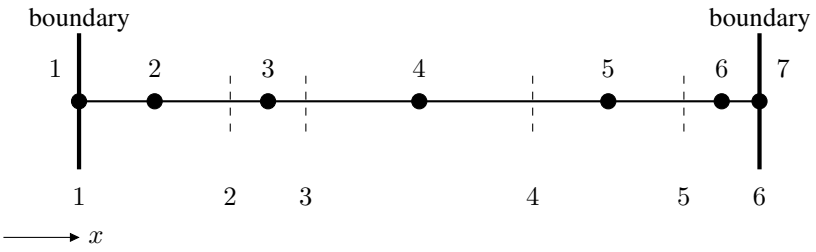

Figure 2.1: 1D grid. $NI = 7$. The bullets denote cell centers which are labeled 1–7. Dashed lines denote control volume faces labeled 1–6. The number of interior control volumes is 5.

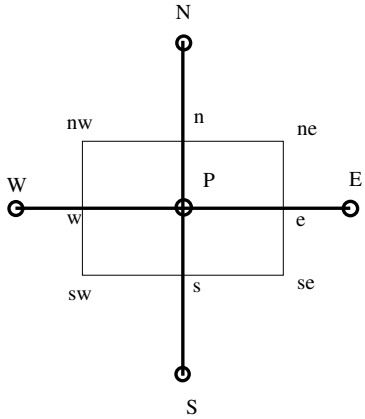

Figure 2.2: Control volume

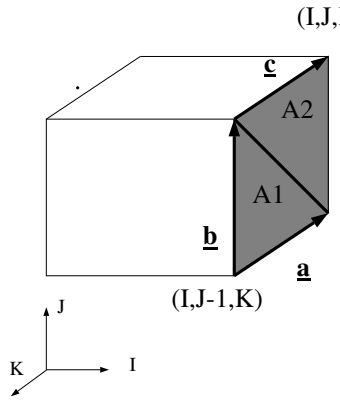

Figure 2.3: Calculation of control volume faces

When a location can not be referred to by a single character, combination of letters are used. The order in which the characters appear is: first east-west, then north-south, and finally high-low.

2.1.2 Area Calculation of Control Volume Faces

The area of the control volume faces are calculated as the sum of two triangles. The x-coordinates of the corners of the east face are, for example, X_C(I,J,K), X_C(I,J-1,K), X_C(I,J,K-1) and X_C(I,J-1,K-1); the Y and Z- coordinates are Y_C and Z_C with the same indices, see Fig. 2.3. The area of the two triangles, $A1$, $A2$, is calculated as the cross product

$$A1 = \frac{1}{2}|\vec{a} \times \vec{b}|; \qquad A2 = \frac{1}{2}|-\vec{b} \times \vec{c}| \tag{2.1}$$

Above is has been assumed that the fourth edge [from (I,J-1,K-1) to (I,J,K-1)] of the east face (shaded area in Fig. 2.3) is approximately equal to $\vec{b}$. The vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ for faces in Fig. 2.3) are set in a manner that the normal vectors $\vec{n1}$ and $\vec{n2}$ are pointed outwards. For the east face, for example, they are defined as

$\vec{a}$: from corner (I,J-1,K) to (I,J-1,K-1)
$\vec{b}$: from corner (I,J-1,K) to (I,J,K)
$\vec{c}$: from corner (I,J,K) to (I,J,K-1)

The Cartesian components of $\vec{a}$ are thus

$$a_x = X(I, J-1, K-1) - X(I, J-1, K) \tag{2.2}$$
$$a_y = Y(I, J-1, K-1) - Y(I, J-1, K) \tag{2.3}$$
$$a_z = Z(I, J-1, K-1) - Z(I, J-1, K) \tag{2.4}$$

The total area for the east face is obtained as

$$|\vec{A}|_e = |\vec{A1}|_e + |\vec{A2}|_e \tag{2.5}$$

The normal vector of the vector area is obtained as the average of the normal vectors of the two triangles

$$\vec{n} = \frac{1}{2}\left(\vec{a} \times \vec{b} - \vec{b} \times \vec{c}\right) \tag{2.6}$$

The Cartesian areas are calculated as

$$\vec{A}_{ex} = |\vec{A}|_e \vec{n} \cdot \vec{e}_x \tag{2.7}$$
$$\vec{A}_{ey} = |\vec{A}|_e \vec{n} \cdot \vec{e}_y \tag{2.8}$$
$$\vec{A}_{ez} = |\vec{A}|_e \vec{n} \cdot \vec{e}_z \tag{2.9}$$

where $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$ are the Cartesian unit base vector

The areas and the normal vectors of the north and high control volumes faces are calculated in exactly the same way. For the north face the vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ are defined as

$\vec{a}$: from corner (I-1,J,K) to (I,J,K)
$\vec{b}$: from corner (I-1,J,K) to (I-1,J,K-1)
$\vec{c}$: from corner (I-1,J,K-1) to (I,J,K-1)

For the high faces the vectors a, b and c are defined as

$\vec{a}$: from corner (I-1,J,K) to (I-1,J-1,K)
$\vec{b}$: from corner (I-1,J,K) to (I,J,K)
$\vec{c}$: from corner (I,J,K) to (I,J-1,K)

In **CALC-LES** the Cartesian areas for each face (east, north and high) are calculated only once and stored in three dimensional arrays.

2.1.3 Volume Calculation of Control Volume

The volume is calculated using Gauss' law, see Burns and Wilkes [4]. Gauss' law for a vector field $\vec{B}$ reads

$$\int_V \nabla \cdot \vec{B} dV = \int_A \vec{B} \cdot d\vec{A} \tag{2.10}$$

setting $\vec{B}$ = $\vec{x}$ gives

$$\delta V = \int_V dV = \frac{1}{3} \int_A \vec{x} \cdot d\vec{A} \tag{2.11}$$

In **CALC-LES** the volume is calculated once only and stored in a three dimensional array.

2.1.4 Interpolation

The nodes where all variables are stored are situated in the center of the control volume. When a variable is needed at a control volume face, linear interpolation is used. The value of the variable ϕ at the east face is

$$\phi_e = f_x \phi_E + (1 - f_x)\phi_P \tag{2.12}$$

where

$$f_x = \frac{|\vec{Pe}|}{|\vec{Pe}| + |\vec{eE}|} \tag{2.13}$$

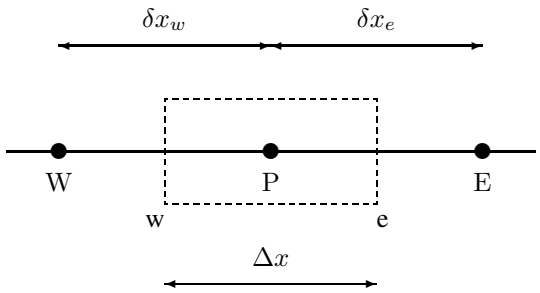

Figure 3.1: 1D control volume. Node P located in the middle of the control volume.

where $|\vec{Pe}|$ is the distance from P (the node) to e (the east face). In **CALC-LES** the interpolation factors (f_x, f_y, f_z) are calculated once only and stored in three-dimensional arrays.

2.2 Gradient

The derivatives of ϕ ($\partial\phi/\partial x_i$) at the cell center are in **CALC-LES** computed as follows. We apply Green's formula to the control volume, i.e.

$$\frac{\partial\Phi}{\partial x} = \frac{1}{V}\int_A \Phi n_x dA, \quad \frac{\partial\Phi}{\partial y} = \frac{1}{V}\int_A \Phi n_y dA, \quad \frac{\partial\Phi}{\partial z} = \frac{1}{V}\int_A \Phi n_z dA$$

where A is the surface enclosing the volume V. For the x component, for example, we get

$$\frac{\partial\Phi}{\partial x} = \frac{1}{V}\left(\Phi_e A_{ex} - \Phi_w A_{wx} + \Phi_n A_{nx} - \Phi_s A_{sx} + \Phi_h A_{hx} - \Phi_l A_{lx}\right) \tag{2.14}$$

where index w, e, s, n, l, h denotes east ($I+1/2$), west ($I-1/2$), north ($J+1/2$), south ($J-1/2$), high ($K+1/2$) and low ($K-1/2$). The derivative $\partial\Phi/\partial x$ is computed in the function `dphidx`.

3 Diffusion

We start by looking at 1D diffusion, e.g. the 1D heat conduction equation

$$\frac{d}{dx}\left(k\frac{dT}{dx}\right) + S = 0.$$

To discretize (i.e. to go from a *continuous* differential equation to an algebraic *discrete* equation) this equation is integrated over a control volume (C.V.), see Fig. 3.1.

$$\int_w^e \left[\frac{d}{dx}\left(k\frac{dT}{dx}\right) + S\right] dx = \left(k\frac{dT}{dx}\right)_e - \left(k\frac{dT}{dx}\right)_w + \bar{S}\Delta x = 0 \tag{3.1}$$

where (see Fig. 3.1):

P: an arbitrary node

E, W: its east and west neighbor node, respectively

e, w: the control volume's east and west face, respectively

$\bar{S}$: volume average of S

The temperature T and the coefficient of heat conductivity k are stored at the nodes W, P and E. Now we need the derivatives dT/dx at the faces w and e. These are estimated from a straight line connecting the two adjacent nodes, i.e.

$$\left(\frac{dT}{dx}\right)_e \simeq \frac{T_E - T_P}{\delta x_e}, \quad \left(\frac{dT}{dx}\right)_w \simeq \frac{T_P - T_W}{\delta x_w}. \tag{3.2}$$

The heat conductivity k is also needed at the faces. It is estimated by linear interpolation between the adjacent nodes. For the east face, for example, we obtain

$$k_e = f_x k_E + (1 - f_x) k_P, \; f_x = \frac{0.5\Delta x}{\delta x_e}. \tag{3.3}$$

For an equidistant mesh (constant $\Delta x \Rightarrow \; \Delta x = \delta x_w = \delta x_e$) $f_x = 0.5$.

Insertion of Eq. 3.2 into Eq. 3.1 gives

$$\begin{aligned} a_P T_P &= a_E T_E + a_W T_W + S_U \\ a_E &= \frac{k_e}{\delta x_e}, \; a_W = \frac{k_w}{\delta x_w}, \; S_U = \bar{S}\Delta x, \; a_P = a_E + a_W \end{aligned} \tag{3.4}$$

3.1 Convergence Criteria

Compute the residual for Eq. 3.4

$$R = \sum_{\text{all cells}} |a_E T_E + a_W T_W + S_U - a_P T_P|$$

Since we want Eq. 3.4 to be satisfied, the difference of the right-hand side and the left-hand side is a good measure of how well the equation is satisfied. Note that R has the units of the integrated differential equation. Thus, in the present case R has the same dimension as heat transfer rate, i.e. Joule per second $[J/s] = [W]$. If $R = 1[W]$, it means that the residual for the computation is 1. This does not tell us anything, since it is problem dependent. We can have a problem where the total heat transfer rate is $1000[W]$, and a another where it is only $1[W]$. In the former case $R = 1$ means that the solutions can be considered converged, but in the latter case this is not true at all. We realize that we must normalize the residual to be able to judge whether the equation system has converged or not. The criterion for convergence is then

$$\frac{R}{F} \leq \varepsilon$$

where $0.0001 < \varepsilon < 0.01$, and F represents the total flux of T, i.e. total heat transfer rate.

Regardless if we solve the continuity equation, the Navier-Stokes equation or the energy equation, the procedure is the same: F should represent the total flux of the dependent variable.

- Continuity equation. F is here the total incoming mass flux $\dot{m}$.

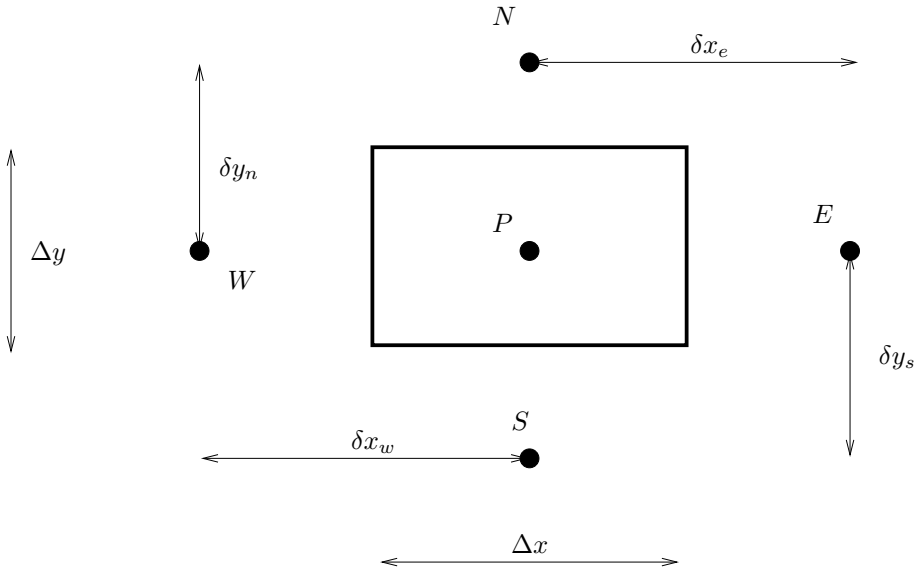

Figure 3.2: 2D control volume.

- Navier-Stokes equation. The unit is that of a force, i.e. Newton. A suitable value of F is obtained from $F = \dot{m}U$ at the inlet.
- Energy equation. F should be the total incoming heat flux. In a convection-diffusion problem we can take the convective flux at the inlet i.e. $F = \dot{m}c_pT$. In a conduction problem we can integrate the boundary flux, taking the absolute value at each cell, since the sum will be zero in case of internal source. If there are large heat sources in the computational domain, F could be taken as the sum of all heat sources.

3.2 2D Diffusion

The two-dimensional heat conduction equation reads

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + S = 0. \tag{3.5}$$

In the same way as we did for the 1D case, we integrate over our control volume, but now it's in 2D (see Fig. 3.2, i.e.

$$\int_w^e \int_s^n \left[\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + S\right] dxdy = 0.$$

We start by the first term. The integration in x direction is carried out in exactly the same way as in 1D, i.e.

$$\int_w^e \int_s^n \left[\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)\right] dxdy = \int_s^n \left[\left(k\frac{\partial T}{\partial x}\right)_e - \left(k\frac{\partial T}{\partial x}\right)_w\right] dy$$
$$= \int_s^n \left(k_e\frac{T_E - T_P}{\delta x_e} - k_w\frac{T_P - T_W}{\delta x_w}\right) dy$$

Now integrate in the y direction. We do this by estimating the integral

$$\int_s^n f(y)dy = f_P \Delta y + \mathcal{O}\left((\Delta y)^2\right)$$

(i.e. f is taken at the mid-point P) which is second order accurate, since it is exact if f is a linear function. For our equation we get

$$\begin{aligned}&\int_s^n \left(k_e \frac{T_E - T_P}{\delta x_e} - k_w \frac{T_P - T_W}{\delta x_w}\right) dy \\ &= \left(k_e \frac{T_E - T_P}{\delta x_e} - k_w \frac{T_P - T_W}{\delta x_w}\right) \Delta y\end{aligned}$$

Doing the same for the diffusion term in the y direction in Eq. 3.5 gives

$$\begin{aligned}&\left(k_e \frac{T_E - T_P}{\delta x_e} - k_w \frac{T_P - T_W}{\delta x_w}\right) \Delta y \\ &+ \left(k_n \frac{T_N - T_P}{\delta y_n} - k_s \frac{T_P - T_S}{\delta y_s}\right) \Delta x + \bar{S} \Delta x \Delta y = 0\end{aligned}$$

Rewriting it as an algebraic equation for T_P, we get

$$\begin{aligned}a_P T_P &= a_E T_E + a_W T_W + a_N T_N + a_S T_S + S_U \\ a_E &= \frac{k_e \Delta y}{\delta x_e},\ a_W = \frac{k_w \Delta y}{\delta x_w},\ a_N = \frac{k_n \Delta x}{\delta y_n},\ a_S = \frac{k_s \Delta x}{\delta y_s} \\ S_U &= \bar{S} \Delta x \Delta y,\ a_P = a_E + a_W + a_N + a_S - S_P.\end{aligned} \tag{3.6}$$

In this 2D equation we have introduced the general form of the source term; this could also be done in the 1D equation (Eq. 3.4).

For more detail on diffusion, see
http://www.tfd.chalmers.se/~lada/comp_fluid_dynamics/lecture_notes.html

4 Convection – Diffusion

The 1D convection-diffusion equation reads

$$\frac{d}{dx}(\rho U T) = \frac{d}{dx}\left(\Gamma \frac{dT}{dx}\right) + S,\ \Gamma = \frac{k}{c_p}$$

We discretize this equation in the same way as the diffusion equation. We start by integrating over the control volume (see Fig. 4.1).

$$\int_w^e \frac{d}{dx}(\rho U T)\, dx = \int_w^e \left[\frac{d}{dx}\left(\Gamma \frac{dT}{dx}\right) + S\right] dx. \tag{4.1}$$

We start by the convective term (the left-hand side)

$$\int_w^e \frac{d}{dx}(\rho U T)\, dx = (\rho U T)_e - (\rho U T)_w .$$

We assume the velocity U to be known, or, rather, obtained from the solution of the Navier-Stokes equation.

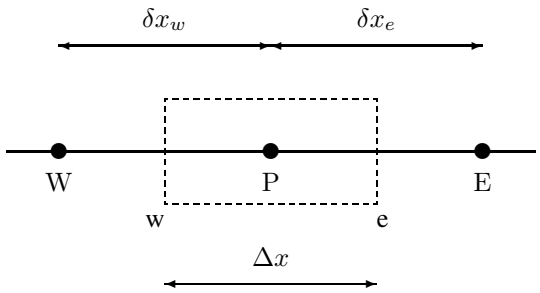

Figure 4.1: 1D control volume. Node P located in the middle of the control volume.

4.1 Central Differencing Scheme (CDS)

How to estimate T_e and T_w? The most natural way is to use linear interpolation (central differencing); for the east face this gives

$$(\rho UT)_e = (\rho U)_e T_e$$

where the convecting part, ρU, is taken by central differencing, and the convected part, T, is estimated with different differencing schemes. We start by using central differencing for T so that

$$(\rho UT)_e = (\rho U)_e T_e, \quad \text{where} \quad T_e = f_x T_E + (1 - f_x) T_P$$

where f_x is the interpolation function (see Eq. 3.3, p. 9), and for constant mesh spacing $f_x = 0.5$. Assuming constant equidistant mesh (i.e. $\delta x_w = \delta x_e = \Delta x$) so that $f_x = 0.5$, inserting the discretized diffusion and the convection terms into Eq. 4.1 we obtain

$$(\rho U)_e \frac{T_E + T_P}{2} - (\rho U)_w \frac{T_P + T_W}{2} =$$
$$= \frac{\Gamma_e (T_E - T_P)}{\delta x_e} - \frac{\Gamma_w (T_P - T_W)}{\delta x_w} + \bar{S} \Delta x$$

which can be rearranged as

$$a_P T_P = a_E T_E + a_W T_W + S_U$$
$$a_E = \frac{\Gamma_e}{\delta x_e} - \frac{1}{2}(\rho U)_e, \; a_W = \frac{\Gamma_w}{\delta x_w} + \frac{1}{2}(\rho U)_w$$
$$S_U = \bar{S} \Delta x, \; a_P = \frac{\Gamma_e}{\delta x_e} + \frac{1}{2}(\rho U)_e + \frac{\Gamma_w}{\delta x_w} - \frac{1}{2}(\rho U)_w$$

We want to compute a_P as the sum of its neighbor coefficients to ensure that $a_P \geq a_E + a_W$ which is the requirement to make sure that the iterative solver converges. We can add

$$(\rho U)_w - (\rho U)_e = 0$$

(the continuity equation) to a_P so that

$$a_P = a_E + a_W.$$

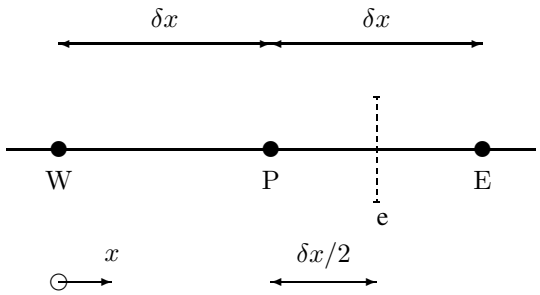

Figure 4.2: Constant mesh spacing. $U > 0$.

Central differencing is second-order accurate (easily verified by Taylor expansion), i.e. the error is proportional to $(\Delta x)^2$. This is very important. If the number of cells in one direction is doubled, the error is reduced by a factor of four. By doubling the number of cells, we can verify that the discretization error is small, i.e. the difference between our algebraic, numerical solution and the exact solution of the differential equation.

Central differencing gives negative coefficients when $|Pe| > 2$; this condition is unfortunately satisfied in most of the computational domain in practice. The result is that it is difficult to obtain a convergent solution in steady flow. However, in LES this does usually not pose any problems.

4.2 First-Order Upwind Scheme

For turbulent quantities upwind schemes must usually be used in order stabilize the numerical procedure. Furthermore, the source terms in these equations are usually very large which means that an accurate estimation of the convection term is less critical.

In this scheme the face value is estimated as

$$T_e = \begin{cases} T_P & \text{if } U_e \geq 0 \\ T_E & \text{otherwise} \end{cases}$$

- first-order accurate
- bounded

The large drawback with this scheme is that it is inaccurate.

4.3 Hybrid Scheme

This scheme is a blend of the central differencing scheme and the first-order upwind scheme. We learned that the central scheme is accurate and stable for $|Pe| \leq 2$. In the Hybrid scheme, the central scheme is used for $|Pe| \leq 2$; otherwise the first-order upwind scheme is used. This scheme is only marginally better than the first-order upwind scheme, as normally $|Pe| > 2$. It should be considered as a first-order scheme.

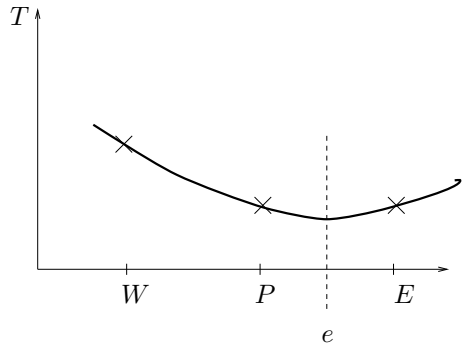

Figure 4.3: Constant mesh spacing. $U > 0$.

4.4 Second-Order Upwind Scheme

We use two nodes upstream and assume that the derivative between W and P is equal to that between P and e, i.e. (see Fig. 4.2)

$$\frac{T_P - T_W}{\delta x} = \frac{T_e - T_P}{\frac{1}{2}\delta x} \Rightarrow T_e \simeq \frac{3}{2}T_P - \frac{1}{2}T_W \tag{4.2}$$

- second-order accurate
- unbounded (negative coefficients), i.e. $T_e < T_W$, $T_e < T_P$ or $T_e < T_E$ (see Fig. 4.3), or vice versa.

4.5 Bounded Second-Order Upwind Scheme

Often, bounded second-order upwind schemes are used. One example is the Van Leer scheme [32]. This scheme reads as follows ($U_e > 0$ assumed):

$$T_e = \begin{cases} T_P + \frac{T_E - T_P}{T_E - T_W}(T_P - T_W) & \text{if } |T_E - 2T_P + T_W| \leq |T_E - T_W| \\ T_P & \text{otherwise} \end{cases} \tag{4.3}$$

see Fig. 4.4, If the variation of T is smooth then

$$\frac{T_E - T_P}{T_E - T_W} \simeq \frac{1}{2},$$

and we find that van Leer scheme gives $T_e = 1.5T_P - 0.5T_W$, i.e. it returns to the second-order upwind scheme (see p. 14).

Now let's illustrate what happens if T has a minimum at node P (dashed line in the Fig. 4.4). We want to show that in this case 1st order upwind is used in the van Leer scheme. When T has a minimum at node P the expression $T_E - 2T_P + T_W$ $[= (\Delta x)^2(d^2T/dx^2)_P^{CD}]$ is larger than $T_E - T_W$ $(= \Delta x(dT/dx)_P^{CD}$. This is seen by rewriting the the first expression as $T_E - 2T_P + T_W = T_E - T_W + 2(T_W - T_P)$ and noting that for the dashed line in Fig. 4.4 $T_W - T_P >$. When so, the condition on the first line of Eq. 4.3 is not satisfied, and thus the second line of of Eq. 4.3 is used, i.e. the first-order upwind scheme is used.

The van Leer scheme

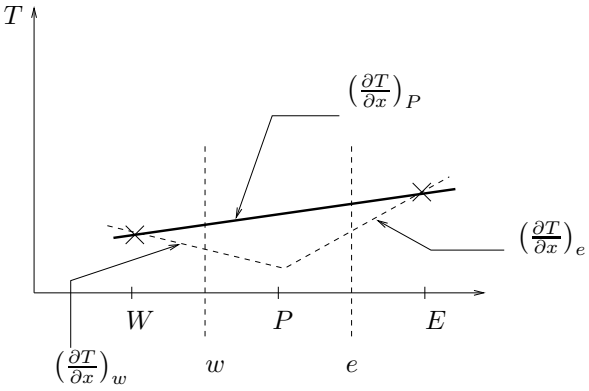

Figure 4.4: Van Leer scheme.

- is second-order accurate, except at local minima and maxima where is only first-order accurate. It can be regarded as a second-order scheme.
- is bounded

MUSCL [33], which is an improved Van Leer scheme is also available in CALC-LES.

5 The Fractional-step method

A numerical method based on an implicit, finite volume method with collocated grid arrangement, central differencing in space, and Crank-Nicolson ($\alpha = 0.5$) in time is briefly described below. An implicit, two-step time-advancement methods is used [6]. The Navier-Stokes equation for the $\bar{u}_i$ velocity reads

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\bar{u}_i \bar{u}_j\right) = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu \frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} \tag{5.1}$$

The discretized momentum equations read

$$\begin{aligned} \bar{v}_i^{n+1/2} &= \bar{v}_i^n + \Delta t H\left(\bar{v}^n, \bar{v}_i^{n+1/2}\right) \\ &-\alpha \Delta t \frac{\partial \bar{p}^{n+1/2}}{\partial x_i} - (1-\alpha)\Delta t \frac{\partial \bar{p}^n}{\partial x_i} \end{aligned} \tag{5.2}$$

where H includes convective, viscous and SGS terms. In SIMPLE notation this equation reads

$$a_P \bar{v}_i^{n+1/2} = \sum_{nb} a_{nb} \bar{v}^{n+1/2} + S_U - \alpha \frac{\partial \bar{p}^{n+1/2}}{\partial x_i} \Delta V$$

where S_U includes the explicit pressure gradient. The face velocities $\bar{v}_{f,i}^{n+1/2} = 0.5(\bar{v}_{i,J}^{n+1/2} + \bar{v}_{i,J-1}^{n+1/2})$ (note that J denotes node number and i is a tensor index) do not satisfy continuity. Create an intermediate velocity field by subtracting the implicit pressure gradient

from Eq. 5.2, i.e.

$$\bar{v}_i^* = \bar{v}_i^n + \Delta t H\left(\bar{v}^n, \bar{v}_i^{n+1/2}\right) - (1-\alpha)\Delta t \frac{\partial \bar{p}^n}{\partial x_i} \tag{5.3a}$$

$$\Rightarrow \bar{v}_i^* = \bar{v}_i^{n+1/2} + \alpha \Delta t \frac{\partial \bar{p}^{n+1/2}}{\partial x_i} \tag{5.3b}$$

Take the divergence of Eq. 5.3b and require that $\partial \bar{v}_{f,i}^{n+1/2}/\partial x_i = 0$ so that

$$\frac{\partial^2 \bar{p}^{n+1}}{\partial x_i \partial x_i} = \frac{1}{\Delta t \alpha} \frac{\partial \bar{v}_{f,i}^*}{\partial x_i} \tag{5.4}$$

The Poisson equation for $\bar{p}^{n+1}$ is solved with an efficient multigrid method [21]. In the 3D MG we use a plane-by-plane 2D MG. The face velocities are corrected as

$$\bar{v}_{f,i}^{n+1} = \bar{v}_{f,i}^* - \alpha \Delta t \frac{\partial \bar{p}^{n+1}}{\partial x_i} \tag{5.5}$$

A few iterations (typically two) solving the momentum equations and the Poisson pressure equation are required each time step to obtain convergence. More details can be found [18].

1. Solve the discretized filtered Navier-Stokes equation, Eq. 5.3a, for $\bar{v}_1$, $\bar{v}_2$ and $\bar{v}_3$.
2. Create an intermediate velocity field $\bar{v}_i^*$ from Eq. 5.3b.
3. Use linear interpolation to obtain the intermediate velocity field, $\bar{v}_{f,i}$, at the face
4. The Poisson equation (Eq. 5.4) is solved with an efficient multigrid method [21].
5. Compute the face velocities (which satisfy continuity) from the pressure and the intermediate face velocity from Eq. 5.5
6. Step 1 to 4 is performed till convergence (normally two or three iterations) is reached.
7. The turbulent viscosity is computed.
8. Next time step.

Since the Poisson solver in [21] is a nested MG solver, it is difficult to parallelize with MPI (Message Passing Interface) on large Linux clusters.

The discretized equation for $\bar{p}$ is assembled in subroutine `calcpe`. It calls the MG solver `peter_multi`. The MG solver includes a number of subroutines:

- `key`
- `key2`
- `mg_2d`
- `peter_init.f` (called once from `main`)
- `peter_2d_cyclic`
- `peter_2d_relax` `peter_cyclic`
- `peter_relax`

It is a stand alone solver; it does not use the usual COMMON block but all information is passed from `main` and `calcpe` via arguments in the `call` statement. The MG subroutines use their own COMMON blocks in `PETER_COMMON`.

	RANS	LES
Domain	2D or 3D	always 3D
Time domain	steady or unsteady	always unsteady
Space discretization	2nd order upwind	central differencing
Time discretization	1st order	2nd order (e.g. C-N)
Turbulence model	more than two-equations	zero- or one-equation

Table 5.1: Differences between a finite volume RANS and LES code.

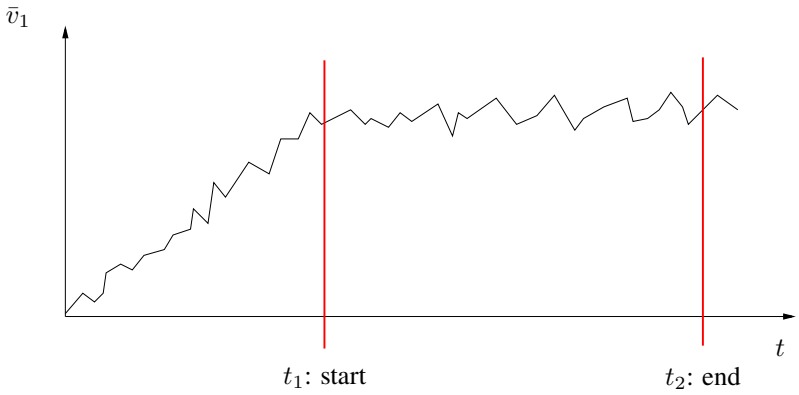

Figure 5.1: Time averaging in LES.

6 Boundary Conditions

6.1 Inlet

The velocity component normal to the inlet (e.g. U if inlet is a west boundary) is usually, as well other scalar variables such as temperature and concentration. Turbulent quantities, such as k and ε are normally not known, but they must be estimated. Usually k is set to $(\gamma U)^2$, where $0.01 \lesssim \gamma \lesssim 0.1$. The dissipation is set from

$$\varepsilon_{in} = c\frac{k_{in}^{3/2}}{L},$$

where $c = 0.54$, and $L = 0.1h$, where h denotes height of inlet. Note that the expressions for k and ε only are guide lines.

6.2 Exit velocity

For *small* outlets the exit velocity can be determined from global continuity. As the inlet is small a constant velocity over the whole outlet can be used. The exit velocity is set as (see Fig. 6.1)

$$U_{in}h_{in} = U_{out}h_{out} \;\Rightarrow U_{out} = U_{in}h_{in}/h_{out}$$

For *large* outlets the exit velocity must be allowed to vary over the outlet. The proper boundary condition in this case is $\partial U/\partial x = 0$. Hence it is important that the flow near the exit is fully developed, so that this boundary condition corresponds to the flow conditions. The best way to ensure this is to locate the exit boundary sufficiently

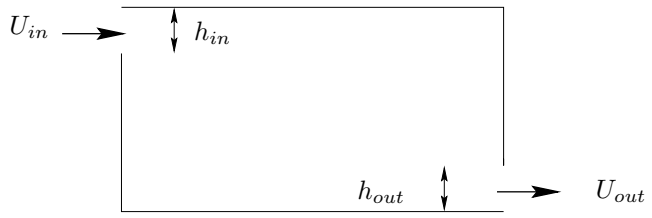

Figure 6.1: Outlet boundary condition. Small outlet

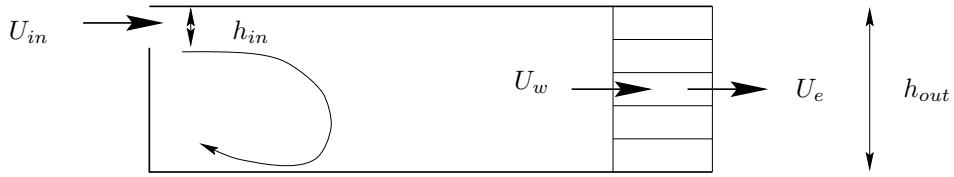

Figure 6.2: Outlet boundary condition. Large outlet.

far downstream. If we have a recirculation region in the domain (see Fig. 6.2), the exit should be located sufficiently far downstream of this region so that $\partial U/\partial x \simeq 0$.

The exit boundary condition is implemented as follows (see Fig. 6.2)

1. Set $U_e = U_w$ for all nodes (i.e. for $j = 2$ to 6, see Fig. 6.2);
2. In order to speed up convergence, enforce global continuity.
   - Inlet mass flow: $\dot{m}_{in} = \rho \sum_{inlet} U_{in} \Delta y$
   - Outlet mass flow: $\dot{m}_{out} = \rho \sum_{outlet} U_{out} \Delta y$
   - Compute correction velocity: $U_{corr} = (\dot{m}_{in} - \dot{m}_{out})/(\rho A_{out})$, where $A_{out} = \sum_{outlet} \Delta y$.
   - Correct U_e so that global continuity (i.e. $\dot{m}_{in} = \dot{m}_{out}$) is satisfied: $U_e^{new} = U_e + U_{corr}$

This boundary condition is implemented in subroutine `mod.f, entry modcon` for the hump flow.

6.3 Remaining variables

Set $\partial\Phi/\partial x = 0$, and implement it through $\Phi_{ni} = \Phi_{ni-1}$ each iteration.

6.4 Interior boundary conditions

Sometimes we want to prescribe a fixed value on a variable in an interior cell. A typical example is turbulent quantities. The wall boundary condition for dissipation, ε, and the specified dissipation, ω, are usually fixed at wall-adjacent nodes as

$$\begin{aligned} \varepsilon_w &= \frac{2\nu k}{y_w^2} \\ \omega_w &= \frac{6\nu}{\beta y_w^2} \end{aligned} \tag{6.1}$$

where y_w is the distance from the cell center to the wall and $\beta = 0.075$. These interior boundary conditions are conveniently prescribed using source terms. The 1D discretized equation with the general source term reads

$$a_P \Phi_P = a_E \Phi_E + a_W \Phi_W + S_U, \; a_P = a_E + a_W - S_P \tag{6.2}$$

with, for example, $\Phi = \varepsilon$. Now we prescribe ε at a wall-adjacent cell with source terms as

$$s_P = -10^{20}, \quad s_U = 10^{20} \varepsilon_w \tag{6.3}$$

Insert Eq. 6.3 into Eq. 6.2 gives

$$10^{20} \Phi_P \simeq 10^{20} \varepsilon_w \tag{6.4}$$

since $a_W \ll 10^{20}$ and $a_E \ll 10^{20}$ which gives $\Phi_P = \varepsilon_w$ as intended.

This boundary condition is implemented in subroutine `mod.f, entry moded`.

7 The Smagorinsky Model

subroutine: `vist_les`

The simplest model is the Smagorinsky model [31]:

$$\begin{aligned} \tau_{ij} - \frac{1}{3}\delta_{ij}\tau_{kk} &= -\nu_{sgs}\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right) = -2\nu_{sgs}\bar{s}_{ij} \\ \nu_{sgs} &= (C_S\Delta)^2 \sqrt{2\bar{s}_{ij}\bar{s}_{ij}} \equiv (C_S\Delta)^2 |\bar{s}| \end{aligned} \tag{7.1}$$

and the filter-width is taken as the local grid size

$$\Delta = (\Delta V_{IJK})^{1/3} \tag{7.2}$$

Near the wall, the SGS viscosity becomes quite large since the velocity gradient is very large at the wall. However, because the SGS turbulent fluctuations near a wall go to zero, so must the SGS viscosity. A damping function f_μ is added to ensure this

$$f_\mu = 1 - \exp(-x_2^+/26) \tag{7.3}$$

A more convenient way to dampen the SGS viscosity near the wall is simply to use the RANS length scale as an upper limit, i.e.

$$\Delta = \min\left\{(\Delta V_{IJK})^{1/3}, \kappa n\right\} \tag{7.4}$$

where n is the distance to the nearest wall. C_S is set to 0.1.

8 The WALE model

subroutine: `vist_wale`

The WALE model by [29] reads

$$\begin{aligned} g_{ij} &= \frac{\partial \bar{v}_i}{\partial x_j}, \; g_{ij}^2 = g_{ik}g_{kj} \\ \bar{s}_{ij}^d &= \frac{1}{2}\left(g_{ij}^2 + g_{ji}^2\right) - \frac{1}{3}\delta_{ij}g_{kk}^2 \\ \nu_{sgs} &= (C_m\Delta)^2 \frac{\left(\bar{s}_{ij}^d \bar{s}_{ij}^d\right)^{3/2}}{\left(\bar{s}_{ij}\bar{s}_{ij}\right)^{5/2} + \left(\bar{s}_{ij}^d \bar{s}_{ij}^d\right)^{5/4}} \end{aligned} \tag{8.1}$$

with $C_m = 0.325$ which corresponds to $C_s = 0.1$.

9 The PANS Model

subroutines: `calcte_pans, calced_pans, vist_pans`

The low-Reynolds number partially averaged Navier-Stokes (LRN PANS) turbulence model reads [25]

$$\begin{aligned}
\frac{Dk}{Dt} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_{ku}}\right)\frac{\partial k}{\partial x_j}\right] + P_k + P_{k_{tr}} - \varepsilon \\
\frac{D\varepsilon}{Dt} &= \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_{\varepsilon u}}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + C_{\varepsilon 1} P_k \frac{\varepsilon}{k} - C^*_{\varepsilon 2}\frac{\varepsilon^2}{k} \\
\nu_t &= C_\mu f_\mu \frac{k^2}{\varepsilon}, P_k = 2\nu_t \bar{s}_{ij}\bar{s}_{ij}, \bar{s}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right) \\
C^*_{\varepsilon 2} &= C_{\varepsilon 1} + \frac{f_k}{f_\varepsilon}(C_{\varepsilon 2} f_2 - C_{\varepsilon 1}), \sigma_{ku} \equiv \sigma_k \frac{f_k^2}{f_\varepsilon}, \sigma_{\varepsilon u} \equiv \sigma_\varepsilon \frac{f_k^2}{f_\varepsilon} \\
\sigma_k &= 1.4, \sigma_\varepsilon = 1.4, C_{\varepsilon 1} = 1.5, C_{\varepsilon 2} = 1.9, C_\mu = 0.09, f_\varepsilon = 1
\end{aligned} \tag{9.1}$$

where $D/Dt = \partial/\partial t + \bar{v}_j \partial/\partial x_j$ denotes the material derivative. The damping functions are defined as

$$\begin{aligned}
f_2 &= \left[1 - \exp\left(-\frac{y^*}{3.1}\right)\right]^2 \left\{1 - 0.3\exp\left[-\left(\frac{R_t}{6.5}\right)^2\right]\right\} \\
f_\mu &= \left[1 - \exp\left(-\frac{y^*}{14}\right)\right]^2 \left\{1 + \frac{5}{R_t^{3/4}}\exp\left[-\left(\frac{R_t}{200}\right)^2\right]\right\} \\
R_t &= \frac{k^2}{\nu\varepsilon}, \quad y^* = \frac{U_\varepsilon y}{\nu}, \quad U_\varepsilon = (\varepsilon\nu)^{1/4}
\end{aligned} \tag{9.2}$$

The term $P_{k_{tr}}$ in Eq. 9.1 is an additional term which is non-zero in the interface region because $Df_k/Dt \neq 0$. This is used at the inlet in the hump flow (see subroutine `mod.f, entry modte`).

The function f_ε, the ratio of the modeled to the total dissipation, is set to one since the turbulent Reynolds number is high. f_k is set to 1 in the RANS region and to 0.4 in the LES region (it may also be computed in the LES region).

10 The $k-\omega$ Model

subroutines: `calcte_kom, calcom, vist_kom`

The Wilcox $k-\omega$ turbulence model reads [36]

$$\begin{aligned}
\frac{\partial k}{\partial t} + \frac{\partial \bar{v}_i k}{\partial x_i} &= P^k - c_\mu k\omega + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \\
\frac{\partial \omega}{\partial t} + \frac{\partial \bar{v}_i \omega}{\partial x_i} &= C_{\omega_1}\frac{\omega}{k}P^k - C_{\omega 2}\omega^2 + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial \omega}{\partial x_j}\right] \\
\nu_t &= \frac{k}{\omega}
\end{aligned} \tag{10.1}$$

where $c_\mu = 0.09, c_{\omega_1} = 5/9, c_{\omega_2} = 3/40, \sigma_k = 0.5 = \sigma_\omega = 2.0$.

11 Inlet boundary conditions

In RANS it is sufficient to supply profiles of the mean quantities such as velocity and temperature plus the turbulent quantities (e.g. k and ε). However, in unsteady simulations (LES, URANS, DES ...) the time history of the velocity and temperature need to be prescribed; the time history corresponds to turbulent, resolved fluctuations. In some flows it is critical to prescribe reasonable turbulent fluctuations, but in many flows it seems to be sufficient to prescribe constant (in time) profiles [8, 10].

There are different ways to create turbulent inlet boundary conditions. One way is to use a pre-cursor DNS or well resolved LES of channel flow. This method is limited to fairly low Reynolds numbers and it is difficult (or impossible) to re-scale the DNS fluctuations to higher Reynolds numbers.

Another method based partly on synthesized fluctuations is the vortex method [24]. It is based on a superposition of coherent eddies where each eddy is described by a shape function that is localized in space. The eddies are generated randomly in the inflow plane and then convected through it. The method is able to reproduce first and second-order statistics as well as two-point correlations.

A third method is to take resolved fluctuations at a plane downstream of the inlet plane, re-scale them and use them as inlet fluctuations.

Below we present a method of generating synthesized inlet fluctuations.

11.1 Synthesized turbulence

The method described below was developed in [2, 3, 15] for creating turbulence for generating noise. It was later further developed for inlet boundary conditions [7, 9, 11].

A turbulent fluctuating velocity fluctuating field (whose average is zero) can be expressed using a Fourier series, see [13]. Let us re-write this formula as

$$\begin{aligned} a_n \cos(nx) + b_n \sin(nx) = \\ c_n \cos(\alpha_n) \cos(nx) + c_n \sin(\alpha_n) \sin(nx) = c_n \cos(nx - \alpha_n) \end{aligned} \tag{11.1}$$

where $a_n = c_n \cos(\alpha)$, $b_n = c_n \sin(\alpha_n)$. The new coefficient, c_n, and the phase angle, α_n, are related to a_n and b_n as

$$c_n = \left(a_n^2 + b_n^2\right)^{1/2} \quad \alpha_n = \arctan\left(\frac{b_n}{a_n}\right) \tag{11.2}$$

A general form for a turbulent velocity field can thus be written as

$$\mathbf{v}'(\mathbf{x}) = 2 \sum_{n=1}^{N} \hat{u}^n \cos(\boldsymbol{\kappa}^n \cdot \mathbf{x} + \psi^n) \boldsymbol{\sigma}^n \tag{11.3}$$

where $\hat{u}^n$, ψ^n and σ_i^n are the amplitude, phase and direction of Fourier mode n. The synthesized turbulence at one time step is generated as follows.

11.2 Random angles

The angles φ^n and θ^n determine the direction of the wavenumber vector $\boldsymbol{\kappa}$, see Eq. 11.3 and Eq. 11.1; α^n denotes the direction of the velocity vector, $\mathbf{v}'$. For more details, see [13].

It is implemented in function `random` in subroutine `synt_generate`.

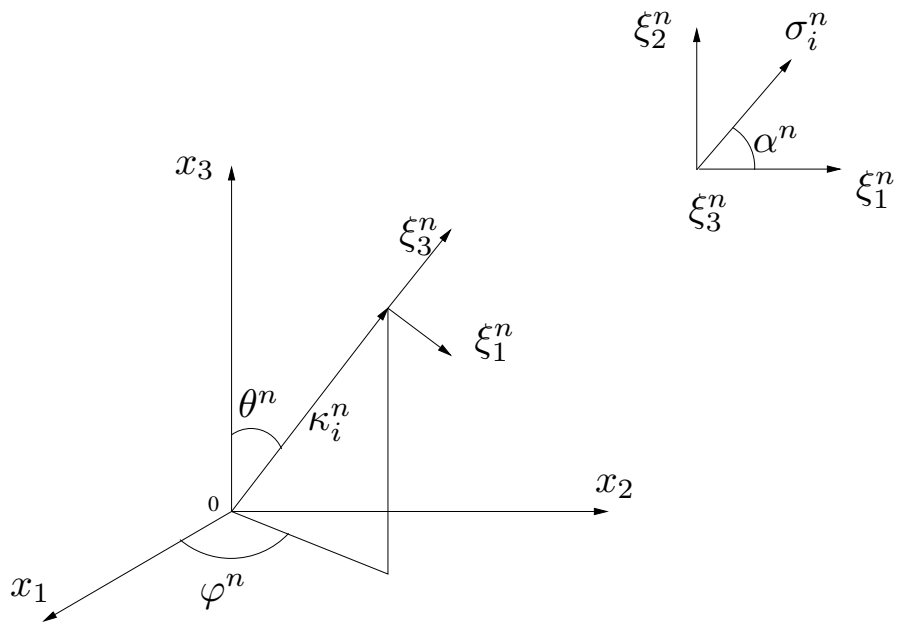

Figure 11.1: The wave-number vector, κ_i^n, and the velocity unit vector, σ_i^n, are orthogonal (in physical space) for each wave number n.

11.3 Highest wave number

Define the highest wave number based on mesh resolution $\kappa_{max} = 2\pi/(2\Delta)$ (see [13]), where Δ is the grid spacing. Often the smallest grid spacing near the wall is too small, and then a slightly larger values may be chosen. The fluctuations are generated on a grid with equidistant spacing (or on a weakly stretched mesh), $\Delta\eta = x_{2,max}/N_2$, $\Delta x_3 = x_{3,max}/N_3$, where η denotes the wall-normal direction and N_2 and N_3 denote the number of cells in the x_2 and x_3 direction, respectively. The fluctuations are set to zero at the wall and are then interpolated to the inlet plane of the CFD grid (the $x_2 - x_3$ plane).

This is implemented in subroutine `synt_init`.

11.4 Smallest wave number

Define the smallest wave number from $\kappa_1 = \kappa_e/p$, where $\kappa_e = \alpha 9\pi/(55L_t)$, $\alpha = 1.453$. The turbulent length scale, L_t, may be estimated in the same way as in RANS simulations, i.e. $L_t \propto \delta$ where δ denotes the inlet boundary layer thickness. In [7,9,11] it was found that $L_t \simeq 0.1\delta_{in}$ is suitable.

Factor p should be larger than one to make the largest scales larger than those corresponding to κ_e. A value $p = 2$ is suitable.

This is implemented in subroutine `synt_init`.

11.5 Divide the wave number range

Divide the wavenumber space, $\kappa_{max} - \kappa_1$, into N modes, equally large, of size $\Delta\kappa$.

This is implemented in subroutine `synt_init`.

11.6 von Kármán spectrum

A modified von Kármán spectrum is chosen, see Eq. 11.4 and Fig. 11.2. The amplitude $\hat{u}^n$ of each mode in Eq. 11.3 is then obtained from

$$\begin{aligned} \hat{u}^n &= (E(\kappa)\Delta\kappa)^{1/2} \\ E(\kappa) &= c_E \frac{u_{rms}^2}{\kappa_e} \frac{(\kappa/\kappa_e)^4}{[1+(\kappa/\kappa_e)^2]^{17/6}} e^{[-2(\kappa/\kappa_\eta)^2]} \\ \kappa &= (\kappa_i\kappa_i)^{1/2}, \quad \kappa_\eta = \varepsilon^{1/4}\nu^{-3/4} \end{aligned} \tag{11.4}$$

The coefficient c_E is obtained by integrating the energy spectrum over all wavenumbers to get the turbulent kinetic energy, i.e.

$$k = \int_0^\infty E(\kappa) d\kappa \tag{11.5}$$

which gives [22]

$$c_E = \frac{4}{\sqrt{\pi}} \frac{\Gamma(17/6)}{\Gamma(1/3)} \simeq 1.453 \tag{11.6}$$

where

$$\Gamma(z) = \int_0^\infty e^{-z'} x^{z-1} dz' \tag{11.7}$$

This is implemented in subroutine `synt_generate`.

11.7 Computing the fluctuations

Having $\hat{u}^n$, κ_j^n, σ_i^n and ψ^n, allows the expression in Eq. 11.3 to be computed, i.e.

$$\begin{aligned} v_1' &= 2\sum_{n=1}^{N} \hat{u}^n \cos(\beta^n)\sigma_1 \\ v_2' &= 2\sum_{n=1}^{N} \hat{u}^n \cos(\beta^n)\sigma_2 \\ v_3' &= 2\sum_{n=1}^{N} \hat{u}^n \cos(\beta^n)\sigma_3 \\ \beta^n &= k_1^n x_1 + k_2^n x_2 + k_3^n x_3 + \psi^n \end{aligned} \tag{11.8}$$

where $\hat{u}^n$ is computed from Eq. 11.4.

In this way inlet fluctuating velocity fields (v_1', v_2', v_3') are created at the inlet $x_2 - x_3$ plane.

This is implemented in subroutine `synt_generate`.

11.8 Introducing time correlation

A fluctuating velocity field is generated each time step as described above. They are independent of each other and their time correlation will thus be zero. This is non-physical. To create correlation in time, new fluctuating velocity fields, $\mathcal{V}_1'$, $\mathcal{V}_2'$, $\mathcal{V}_3'$, are

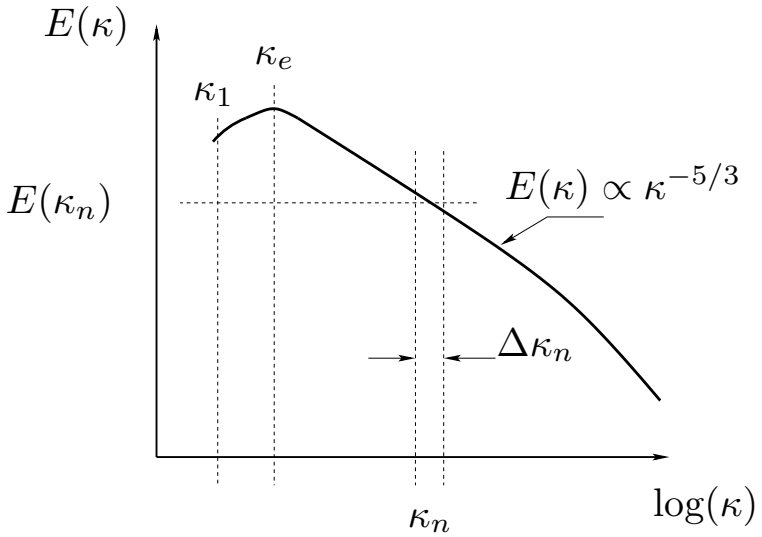

Figure 11.2: Modified von Kármán spectrum

computed based on an asymmetric time filter

$$\begin{aligned}(\mathcal{V}_1')_m &= a(\mathcal{V}_1')_{m-1} + b(v_1')_m \\ (\mathcal{V}_2')_m &= a(\mathcal{V}_2')_{m-1} + b(v_2')_m \\ (\mathcal{V}_3')_m &= a(\mathcal{V}_3')_{m-1} + b(v_3')_m\end{aligned} \tag{11.9}$$

where m denotes the time step number and

$$a = \exp(-\Delta t/T_{int}) \tag{11.10}$$

where Δt and T_{int} denote the computational time step and the integral time scale, respectively. The second coefficient is taken as

$$b = (1 - a^2)^{0.5} \tag{11.11}$$

which ensures that $\langle \mathcal{V}_1'^2 \rangle = \langle v_1'^2 \rangle$ ($\langle \cdot \rangle$ denotes averaging). The time correlation of will be equal to

$$\exp(-\hat{t}/T_{int}) \tag{11.12}$$

where $\hat{t}$ is the time separation and thus Eq. 11.9 is a convenient way to prescribe the turbulent time scale of the fluctuations. For more detail, see Section 11.8. The inlet boundary conditions are prescribed as (we assume that the inlet is located at $x_1 = 0$ and that the mean velocity is constant in the spanwise direction, x_3)

$$\begin{aligned}\bar{v}_1(0, x_2, x_3, t) &= V_{1,in}(x_2) + u_{1,in}'(x_2, x_3, t) \\ \bar{v}_2(0, x_2, x_3, t) &= V_{2,in}(x_2) + v_{2,in}'(x_2, x_3, t) \\ \bar{v}_3(0, x_2, x_3, t) &= V_{3,in}(x_2) + v_{3,in}'(x_2, x_3, t)\end{aligned} \tag{11.13}$$

where $v_{1,in}' = (\mathcal{V}_1')_m$, $v_{2,in}' = (\mathcal{V}_2')_m$ and $v_{3,in}' = (\mathcal{V}_3')_m$ (see Eq. 11.9). The mean inlet profiles, $V_{1,in}$, $V_{2,in}$, $V_{3,in}$, are either taken from experimental data, a RANS

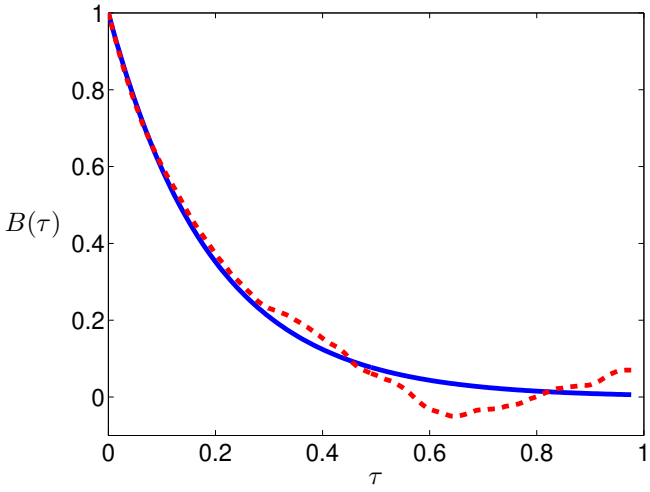

Figure 11.3: Auto correlation, $B(\tau) = \langle v_1'(t)v_1'(t-\tau)_t$ (averaged over time, t). ——: Eq. 11.12; - - -: computed from synthetic data, $(\mathcal{V}_1')^m$, see Eq. 11.9.

solution or from the law of the wall; for example, if $V_{2,in} = V_{3,in} = 0$ we can estimate $V_{1,in}$ as [35]

$$V_{1,in}^+ = \begin{cases} x_2^+ & x_2^+ \leq 5 \\ -3.05 + 5\ln(x_2^+) & 5 < x_2^+ < 30 \\ \frac{1}{\kappa}\ln(x_2^+) + B & x_2^+ \geq 30 \end{cases} \tag{11.14}$$

where $\kappa = 0.4$ and $B = 5.2$.

The method to prescribed fluctuating inlet boundary conditions have been used for channel flow [11], for diffusor flow [8] as well as for the flow over a bump and an axisymmetric hill [12].

This is implemented in subroutine `synt_generate`.

12 Procedure to generate anisotropic synthetic fluctuations

The methodology is as follows:

1. A pre-cursor RANS simulation is made using a RANS model. For fully-developed channel flow, this may be done with the Matlab script `rans` which can be found at [5].

2. After having carried out the pre-cursor RANS simulation, the Reynolds stress tensor is computed using the EARSM model [34]. This is done with the Matlab script `synt_main_earsm` at [5].

3. The Reynolds stress tensor is used as input for generating the anisotropic synthetic fluctuations. The integral length scale, L_{int}, need to be prescribed; it can be set to $0.1\delta < L_{int} < 0.3\delta$, where δ denotes half-channel width.

4. Since the method of synthetic turbulence fluctuations assumes homogeneous turbulence, we can only use the Reynolds stress tensor in one point. We need to choose a relevant location for the Reynolds stress tensor. In a turbulent boundary layer, the Reynolds shear stress is by far the most important stress component. Hence, the Reynolds stress tensor is taken at the location where the magnitude of the turbulent shear stress is largest.

5. Finally, the synthetic fluctuations are scaled with $\left(|\overline{u'v'}|/|\overline{u'v'}|_{max}\right)^{1/2}_{RANS}$, which is taken from the 1D RANS simulation.
   This is done in subroutine `mod` entry `modu` in the hump flow.

The only constant used when generating these synthetic simulations is the prescribed integral length scale.

Eigenvalues and eigenvectors are computed in the Matlab script `synt_main_earsm` which can be found in [5]. In that script, the eigenvalues are denoted by $a11$, $a22$, $a33$ and the eigenvectors by $r11$, $r12$, ... $r33$; they are stored in the files `a_synt_inlet_hump_easrm_j24.dat` and `R_synt_inlet_hump_easrm_j24.dat`, respectively, read by the subroutine `synt_init`.

13 Flow Chart

Go in to the directory `CodeTree` and open the file `index.html` with your Internet browser (in Firefox you should type `file://` in the address field). Or you can simply type `firefox index.html` at the prompter.

14 Subroutines

angle: computes angle in RANS wall function

calced, calced_pans, calcpe, calcph, calcte, calcte_pans, calcte_kom, calcu, calcv, calcw, calcom: compute source terms etc for ε, ε (PANS), p, Φ, k, k (PANS),k $(k-\omega)$, $\bar{v}$, $\bar{v}$, $\bar{w}$ and ω)

calcte_keqabl one-equation turbulence model (only used for ABL (atmospheric boundary layer))

coeff: compute discretized coefficients for different discretization schemes

conv: compute mass flux through control volume face

ctdma: cyclic TDMA

dphidx, dphidxo, dphidy, dphidyo, dphidz, dphidzo: $\dfrac{\partial \Phi}{\partial x}$, $\dfrac{\partial \Phi^o}{\partial x}$ (old time step), $\dfrac{\partial \Phi}{\partial y}$, ..., $\dfrac{\partial \Phi^o}{\partial z}$ (old time step)

echo1: echoes input data

forest: computes the drag force in the forest (only used for ABL)

forest_init: calculates the LAD (Leaf-Area Density) of a given forest (only used for ABL)

forest_heat: calculates the forest heat flux as a distributed source term (only used for ABL)

init: computes geometrical quantities, initialization, ...

key, key2: pointers used in MG solver for $\bar{p}$

main: main

mg_2d: 2D MG solver for $\bar{p}$

muscle: MUSCL discretization scheme

peter_2d_relax, peter_cyclic, peter_init, peter_multi, peter_relax: subroutines used in MG solver for $\bar{p}$

restr1: if `restrt.eq.true`, reads binary file `savres`

save1: `save.eq.true`, (over)writes binary file `savres`

solvt: TDMA solver

statisz: time averaging for post-processing

synt_generate, synt_init: files for generating synthetic inlet fluctuations

update: update `phi=phio`

van_leer: van Leer scheme

vist, vist_les, vist_pans, vist_wale, vis_kom: computes turbulent viscosity for the $k-\varepsilon$, Smagorinsky, PANS, WALE and $k-\omega$ model

vist_smagabl, vist_keqabl computes turbulent viscosity the Smagorinsky and the one-equation turbulence model (only used for ABL (atmospheric boundary layer))

15 Fully-developed channel flow

DNS of Fully-developed channel flow at $Re_\tau = 500$. The case is defined in subroutines `setup` and `mod`.

15.1 Setup

15.1.1 Section 1

`nphmax=6` Number of variables in `phi`. If you want to solve for more variables, increase this.

15.1.2 Section 2

Default values.

15.1.3 Section 3

`urfvis=0.5` Under-relaxation for turbulent viscosity (see `vist_wale`).

15.1.4 Section 4

`maxit=10` Maximum number of outer iterations.

15.1.5 Section 5

`sormax =1.e-3` Convergence limit at each time step.

15.1.6 Section 6

`densit =1.` density

`prt_lam(t)=0.72` viscous Prandtl number

`re =500.` Reynolds number

`viscos =1./re` viscosity

15.1.7 Section 7

`nsweep(u)=2` Number of sweeps in the TDMA solver for $\bar{v}$ (see `solvt`).

`nsweep(v)=2` Number of sweeps in the TDMA solver for $\bar{v}$

`nsweep(w)=2` Number of sweeps in the TDMA solver for $\bar{w}$

`nsweep(p)=3` Number of sweeps in the MB solver for $\bar{p}$

15.1.8 Section 8

`restrt=.true.` Read initial flow fields from binary file `savres`

`les=.false.` Not Smagorinsky turbulence model

`wale=.false.` Not WALE turbulence model

`pans=.false.` Not PANS turbulence model

`save=.true.` (over)writes flow fields in binary file `savres`

`cycli=.true.` Use cyclic boundary condition in x direction

`cyclk=.true.` Use cyclic boundary condition in z direction

`steady=.false.` Not steady

`echo=.true.` Echoes input data

15.1.9 Section 9

`scheme='c'` Use central scheme

`acrank_conv=0.51` Use Crank-Nicolson for convection-diffusion (see `solvt`)

`acrank=0.6` Use slightly more implicit than Crank-Nicolson for the pressure gradient (see, for instance, `calcu`)

`schtur='h'` Use hybrid central/upwinding for turbulent quantities (not relevant since we are doing DNS)

15.1.10 Section 10

`betap=1.` Driving pressure gradient (see `mod`)

15.1.11 Section 11

`solve(u)=.true.` Solve for $\bar{v}$

`solve(v)=.true.` Solve for $\bar{v}$

`solve(w)=.true.` Solve for $\bar{w}$

`solve(te)=.false.` Don't solve for k

`solve(ed)=.false.` Don't solve for ε

`solve(p)=.true.` Solve for $\bar{p}$

15.1.12 Section 12

`ni=98` Number of cells in x direction (including boundary nodes)

`nj=98` Number of cells in y direction (including boundary nodes)

`nk=98` Number of cells in z direction (including boundary nodes)

Note that the MG solver for pressure wants as many grid levels as possible. This means that we want to be able to divide the number of cells in each direction with 2^{m-1} where m is as large as possible. In this case $m = 6$ (i.e. 6 MG levels) so that the number of cells on the coarsest grid level is 3 ($96/2^5$). The number of MG levels are computed in `peter_init` and written to standard output

```
NUMBER OF LEVELS= 6
```

`xmax=2.*3.14` The domain in x direction is $2 \cdot 3.14$ (equidistant)

`zmax=1.*3.14` The domain in z direction is 3.14 (equidistant)

`yfac=1.065` Stretching away from the walls. Constant cells are used near walls and in the center

`uin=20` Approximate bulk velocity

`ntstep=40000` number of time steps

`iprint_start=20000` start time averaging from this time step

`dt(i)=0.5*xc(2,2,2)/uin` time step (should give a CFL number of approximately 0.4)

15.1.13 Section 13

`reref(p)=ymax*zmax*uin` Scale residuals of pressure (continuity equation)

`reref(u)=ymax*zmax*uin**2` Scale residuals of $\bar{v}$ (Navier-Stokes)

`reref(v)=ymax*zmax*uin**2` Scale residuals of $\bar{v}$ (Navier-Stokes)

`reref(w)=ymax*zmax*uin**2` Scale residuals of $\bar{w}$ (Navier-Stokes)

15.1.14 Section 14

`imon, jmon, kmon` Print time history of field variables for this cell

15.1.15 Section 15-17

Not relevant since we are doing DNS

15.2 mod

15.2.1 entry modini

It is called from `main`.
Compute `delta`.

15.2.2 entry modpro

It is called from `vist_les`, `vist_wale` and `vist_pans`.
Set cyclic boundary conditions on `vis`.

15.2.3 entry modcon

It is called from `conv`.
Set cyclic or zero boundary conditions on `conve`,`convn` and `convh`. Compute CFL.

15.2.4 entry modu

It is called from `calcu`.
Set the driving pressure gradient. It is either set to 1 which gives $\tau_w = 1$ (i.e. Re_τ is prescribed). Or it is set by setting the bulk velocity, U_b (i.e. Re_b is prescribed).

15.2.5 entry modv

It is called from `calcv`.

15.2.6 entry modw

It is called from `calcw`.

15.2.7 entry modpp

It is called from `calcpe`.
Set cyclic or zero boundary conditions on `p`.

15.2.8 entry modte

It is called from `calcte`.

15.2.9 entry moded

It is called from `calced`.

15.2.10 entry modphi

It is called from `calcph`.

15.3 Run the code

Go into the main directory `calc-les-2018-june` and continue down into the directory `channel-500-DNS`. The first time, remove all object files because the code has been compiled on my Linux machine with a different compiler. Do this by

```
rm -f *.o
rm -f ../*.o
rm -f ../*/*.o
```

When you run the code, the post-processing file, `vectz.dat`, will be over-written. Also the re-start file `savres`, will be over-written. Hence copy these two files to other names, e.g.

```
cp vectz.dat vectz_org.dat
cp savres savres_org.dat
```

Compile the code by typing

```
make
```

The default compiler is `gfortran`. The gfortan compiler is used In the file `makefile`. If you want to change compiler, remember to first remove all object files (*.o).
Let's run 100 time steps. Change `ntstep` and `iprint_start` in `setup` to

```
ntstep=100
iprint_start=1
```

The code will run 100 time steps and start time-averaging from the first time step. Run the code by typing

```
./calc-les > out&
```

Look at the file `out`. For example, you will find the line

```
NUMBER OF LEVELS= 6.
```

This show that the MG solver in `peter_init` has created six MG levels.

16 Fully-developed channel flow without re-start

In Section 15 we re-start the simulations from a previous simulations (i.e. `restrt=.true.`). Here we will start from scratch, i.e. we create a realistic initial flow field.

16.1 Setup

16.1.1 Section 8

`restrt=.false.` No re-start

`nfiles_restart=4:` save four fields ($\bar{v}, \bar{v}, \bar{w}, \bar{p}$) in `save1`

16.2 mod

16.2.1 entry modini

A RANS velocity and shear stress are read. These can be created with the RANS solver `rans.m` [5]. Synthetic turbulence is superimposed to the RANS field. This is done plane-by-plane ($y-z$ planes) in exactly the same way as inlet synthetic fluctuations

(see Section 11.1). In Section 11.1 many inlet planes are created and correlation in time is achieved by Eq. 11.9. Here we create many $y - z$ planes and correlation is introduced using the same formula but the timestep is taken as $\Delta t = \Delta x / U_{bulk}$; Δt is the time it takes to convect the turbulence from x to $x + \Delta x$.

17 Hill flow

LES of periodic hill flow using the PANS model. at $Re_b = 10\,595$. The case is defined in subroutines `setup` and `mod`. Here we will comment the sections that differ from those for the channel flow (see Section 15). We re-start the simulations from a previous simulations (i.e. `restrt=.true.`).

Go to the directory `hill-pans`.

17.1 Setup

17.1.1 Section 8

`pans=.true.` Use PANS turbulence model

17.1.2 Section 9

`acrank=0.8` Use more implicit than Crank-Nicolson for the pressure gradient (see, for instance, `calcu`)

17.1.3 Section 10

`betap=1.` Not used. It is recomputed in `mod`, `entry modu`

17.1.4 Section 11

`solve(te)=.true.` Solve for k

`solve(ed)=.true.` Solve for ε

17.1.5 Section 12

`open(unit=11,file='hill_161_81.dat',status='unknown')` Read grid

17.1.6 Section 16

`c1=1.5` $C_{\varepsilon 1}$ in ε equation

`c2=1.9` $C_{\varepsilon 2}$ in ε equation

17.1.7 Section 16

`prt(te)=-1.4` in k equation; it is re-computed in `coeff`

`prt(ed)=-1.4` in ε equation; it is re-computed in `coeff`

17.2 mod

17.2.1 entry modini

Compute `dist2d` which is the distance to the nearest wall. It is used in `vist_pans` and `calced_pans`.

17.2.2 entry modu

Compute the driving pressure gradient from a balance of all forces on the surfaces, i.e. wall shear stresses and pressure force. For more details, see Section 3.5 in Iran-nezhad [23].

17.2.3 entry moded

Sets wall boundary conditions

$$\varepsilon_w = 2\nu k/d^2$$

where d is the distance from the cell center to the wall.

18 Hump flow with re-start

LES of hump flow using the PANS model. at $Re_b = 9.36 \cdot 10^5$. This work is presented in [19].

The case is defined in subroutines `setup` and `mod`. Here we will comment the sections that differ from those for the hill flow. We re-start the simulations from a previous simulations (i.e. `restrt=.true.`).

Go to the directory `hump-computed-fk`.

18.1 Setup

18.1.1 Section 9

`scheme='m'` Use the MUSCL discretization scheme for $\bar{v}$, $\bar{v}$ and $\bar{w}$

`blend=0.95'` Use a blend of CDS and MUSCL (5% MUSCL)

18.1.2 Section 8

`cycli=.false.` No cyclic boundary condition in x direction

18.2 mod

18.2.1 entry modini

`fk(i,j,k)=0.4` Sets, initially, $f_k = 1$ near the wall and to 0.4 away from the wall.

`read(84,*)y2,umean_in(j),vmean_in(j),rkmean_in(j),epsmean_in(j)`

`dummy_uv,dummy_p,uv_rans(j)` Reads inlet b.c.

`phi(1,j,k,u)=umean_in(j)` Sets initial conditions from inlet b.c.

`nstep=500`

`do n=1,nstep` create 500 planes of synthetic fluctuations in order to compute the synthetic maximum shear stress `uv_synt` (it should be homogeneous)

`ss_u=(tauw/uv_synt)**0.5` scaling factor to get correct (i.e. same as RANS) peak of shear stress for the synthetic inlet fluctuations

18.2.2 entry modcon

Sets outlet b.c. according to Fig. 6.2.

18.2.3 entry modu

`umeanv(i,j)=umeanv(i,j)+phi(i,j,k,u)/fnk` Computes time-averaged normal Reynolds stresses (to be used when prescribing additional source term in k equation, see `modte`)

`an(i,njm1,k)=0.` Neumann b.c. on north boundary.

`term1=1.-(psi-1.)/(c2-c1)` Compute f_k [17]

`rl_in=0.2*0.015` Prescribe length scale of synthetic inlet fluctuations 0.015 is the boundary layer width

`call synt_init` Compute initial arrays etc for the synthetic fluctuations

`a=exp(-dt(itstep)/tturb)` a and b from Eqs. 11.10 and 11.11.

`call synt_generate` Create synthetic fluctuations

`ss_k=ss_u*(ruv/uvmax)**0.5` Scale inlet fluctuations (both peak and profile according to uv_rans)

`ufluct_inlet(j,k)=a*ufluct_inlet(j,k)+b*uprim` see Eq. 11.9.

`phi(1,j,k,u)=phi(1,j,k,u)-uinc` This make sure that the integral of `ufluct_inlet(j,k)` over the inlet is zero. This may help convergence.

18.2.4 entry modte

`rk_source=(0.5*(vprim2+uprim2+wprim2)+phi(i,j,k,te))*dfk_dt` This is the source term due to the commutation error, see Eq. 19 in [14]. The inlet boundary condition is set to `rkmean_in]` see `modini`.

`gente(i,njm1,k)=0.` This is done to fix convergence problems which sometime appear due to the irregular grid near the upper boundary (recall that it is a symmetry boundary and no turbulence production should occur there)

18.2.5 entry moded

`epsmean_in` The inlet boundary condition is set to `epsmean_in`, see `modini`.

19 Hump flow without re-start

When the simulations are not re-started from an earlier simulations, different tricks must often be used to make the simulations stable. There are two standard options during the first couple of 100 timesteps:

- use a smaller timestep
- use a dissipative first-order upwind scheme (the hybrid scheme).

Here, it is sufficient to use the first option to make the simulations stable.

19.1 setup

19.1.1 Section 8

`restrt=.false.` No re-start

`nfiles_restart=4:` save four fields ($\bar{v}, \bar{v}, \bar{w}, \bar{p}$) in `save1`

19.1.2 Section 12

`if (.not.restrt.and.i.le.200) dt(i)=0.0002.` To increase numerical stability, use smaller timestep the first 200 timesteps

19.2 mod

19.2.1 entry modini

Synthetic initial fluctuations are created in the same way as in Section 16.2.1. The difference is here that the flow is not homogeneous in x direction. Furthermore, a 2D RANS field must be created, see Section 20.

`open(unit=17,file='savres_rans',form='unformatted',status='old')` The 2D RANS field is read and used as initial mean field.

`call init_turb` create synthetic initial fluctuations

19.2.2 entry modu

`scheme=scheme_org` To increase numerical stability, use the first-order hybrid central/upwind discretization scheme the first `iadd_source` timesteps.

20 Hump flow, 2D RANS

CALC-LES cannot run 2D flow because the multi-grid solver for pressure is a 3D solver.

20.1 setup

20.1.1 Section 9

`scheme='h'` Choose hybrid central/upwind scheme to make the simulations more stable.

20.1.2 Section 12

`nkm1=9` We get eight cells in z direction. That should give three MG-levels

When you look in `out` you will find the line
`NUMBER OF LEVELS= 3.`
This shows that the MG solver in `peter_init` has created three MG levels.

20.1.3 Section 14

`imon =ni-5`

`jmon =3` It is difficult to know when the 2D RANS flowfield has reached a steady state. Here we look at the monitor point at the far end of the domain close to the wall. Note that it is not critical that the flow reaches a fully steady state; the object is only to use these results as initial conditions for the PANS simulations later on.

20.1.4 Section 18

`fk(i,j,k)=1.0` This turns the PANS model into the RANS AKN model [1].

When the simulations have been finished, copy the `savres` file to the PANS simulations, i.e.

```
cp savres ../hump-computed-fk-no-restart/savres_rans
```

21 Atmospheric boundary layer in a forest

The application of this case is windpower in forests. The work is presented in [26–28].

21.1 setup

21.1.1 Section 2a

Details on the forest and the ABL (Atmospheric Boundary Layer) are set here.

21.1.2 Section 8

`keq_abl=.true.` The one-equation turbulence model is used

21.1.3 Section 11

`if (stability) solve(t)=.true.` The temperature equation is solved

21.1.4 Section 12

`xmax=1000.` x domain extent is $1000m$ (streamwise)

`ymax=1000.` y domain extent is $1000m$ (vertical)

`zmax=1000.` z domain extent is $1000m$ (lateral)

`dly=2.0` the cells in the forest is set to $2m$

`if (y(j).gt.210.) yfac=1.105` stretch the cells by 1.105% above $210m$

`dly=min(dly,10.)` don't let the cells be larger than $10m$

21.1.5 Section 13

`dt(i)=0.2` Timestep is $0.2s$

21.2 mod

21.2.1 entry modini

`call forest_init` compute LAD

`deltaIDDES=min(max(h1,h2,h3),del_max)` compute Δ

`if (.not.restrt) then` if no re-start, initialize the flow

21.2.2 entry modpro

`vist=(0.41/term1)**2*yp1*upar` the wall-function boundary condition is implemented by setting the wall turbulent viscosity, see Section 3.4.1 in [16].

21.2.3 entry modu

`deltapx = alpha_geo*(udes-uhub)*apmean/volmean` the pressure gradient is adjusted so that the $\bar{v}$ should be `udes`.

`su(i,j,k)=su(i,j,k)-fcori*phi(i,j,k,w)*vol(i,j,k)` add the Coriolis force

`sp(i,j,k)=sp(i,j,k)+forceu(i,j,k)*vol(i,j,k)` add the drag force due to the forest

21.2.4 entry modv

`sp(i,j,k)=sp(i,j,k)+forcev(i,j,k)*vol(i,j,k)` add the drag force due to the forest

`su(i,j,k)=su(i,j,k)+grav*volextcoef*(phi(i,j,k,t)-tplane)*vol(i,j,k)` add the buoyancy force

21.2.5 entry modw

`deltapx = alpha_geo*(wdes-whub)*apmean/volmean` the pressure gradient is adjusted so that the $\bar{w}$ should be `wdes`.

`sp(i,j,k)=sp(i,j,k)+forcew(i,j,k)*vol(i,j,k)` add the drag force due to the forest

`su(i,j,k)=su(i,j,k)+fcori*phi(i,j,k,u)*vol(i,j,k)` add the Coriolis force

21.2.6 entry modphi(nphi)

`su(i,j,k) = su(i,j,k) + dqdy(j)*vol(i,j,k)` add the heat sink due to the forest

22 COMMON blocks

22.1 COMMON

Make sure that the dimension of the vectors is sufficient.

`it` Must be larger than `ni`

`jt` Must be larger than `nj`

`kt` Must be larger than `nk`

`nphit, nphito` Must be larger than `numph`

If you change anything in this file, you must re-compile the entire code. This is done by deleting all object files *.o and typing `make`.

22.2 PETER_COMMON

`iomax` Must be larger than `1.2 * ni * nj * nk`

`io2dmax` Must be larger than `1.5 * n1 * n2` where `n1` is the largest of (`ni, nj, nk`) and `n2` is the second largest

If you change anything in this file, you must re-compile the entire code. This is done by deleting all object files *.o and typing `make`.

22.3 Variables in COMMON blocks in file `COMMON`

`acrank:` time integration scheme for pressure (1: fully implicit)

`acrank_conv:` time integration scheme for convection and diffusion (1: fully implicit)

`alpha_geo:` some kind of underrelaxation factor on the pressure gradient (only for ABL)

`areaex,areaey,areaez:` the x, y and z component of the area of east face

`areahx,areahy,areahz:` the x, y and z component of the area of high face

`areanx,areany,areanz:` the x, y and z component of the area of north face

`aw,ae,as,an,al,ah,ap:` discretization coefficients, a_W, a_W, a_S, a_N, a_L, a_H, a_P

`betap:` driving pressure gradient

`c1,c2,cappa:` turbulence model constants, $C_{\varepsilon 1}$, $C_{\varepsilon 2}$, κ

`cdfor:` drag value of forest [30] (only for ABL)

`cdiss:` dissipation parameter [20] (only for ABL[1])

`cmu,cd,cmucd:` turbulence model constants, C_μ, $C_d = 1$, $cmucd = cmu * cd$

`conv_blend:` blending factor between CDS and MUSCL (1: fully CDS)

`conve,convn,convh:` convection through east, north and high face

`cycli,cyclk:` if true, cyclic in i and j direction

`delta_leaf:` leaf size, lf [30] (only for ABL)

`densit:` density, ρ

`dist2d:` distance to the nearest wall (used in PANS)

`dqdy:` y derivative of heat flux in forest (only for ABL)

`dt:` time step

`echo:` if true, echoes settings

`elog,pfun:` constants in RANS wall-function, E, P

`extinct:` extinction coefficient in forest (only for ABL)

`f_lad:` Leaf-Area Density (LAD) (only for ABL)

`fcori:` Coriolis coefficient (only for ABL)

`fk,fe:` f_k, f_ε, used in PANS

`flowangle:` the direction of the wind [degrees] (only for ABL)

`fmax_leaf_area_dens:` max value for the leaf area density (only for ABL)

`forceu,forcev,forcew:` Drag force due to forest in u, v and w momentum equations (only for ABL)

`forheight:` forest height (only for ABL)

`formaxdens,:` height at which the forest is most dense (only for ABL)

`fx,fy,fz:` f_x, f_y, f_y, the interpolation function in I, J and K direction

`gente:` velocity derivatives in P^k in the k and ε equations

`grav:` accelleration gravity coefficient (9.81) (only for ABL)

`great:` 1e20

`head(nphi):` name of variable `nphi`

`heat_forest:` total heat source in forest (only for ABL)

[1] Amtospheric boundary layers

`hWind`: the height at which the wind should be specified [m] (only for ABL)

`imon,jmon,kmon`: print time history of `phi` for this node

`iprint_start`: timestep at which time averaging begins

`it,jt,kt`: dimension of the arrays (i,j,k direction)

`iter`: current global iteration

`itstep`: current timestep

`jformax`: number of cells in forest (only for ABL)

`jhub`: hub height, index j (only for ABL)

`jl0`: interface between RANS and LES in PANS

`keq_abl`: one-equation SGS model for atmospheric boundary layers (only for ABL)

`kom`: if true, $k-\omega$ model

`les`: if true, LES (Smagorinsky)

`les_abl`: Smagorinsky model (only for ABL)

`maxit`: maximum number of global iterations at each timestep

`name(nphi)`: short name of variable `nphi`

`nfiles_restart`: number of variables in restart file

`ni,nj,nk`: number of cell centers (including boundaries) in i, j and k direction

`nim1,njm1,nkm1`: ni-1, nj-1, nk-1

`nphit,nphito`: size of `phi` and `phio` arrays

`nphmax`: number of nphi

`nsweep(nphi)`: number of sweeps in TDMA solver and MG solver

`ntstep`: number of timesteps

`pans`: if true, PANS model

`phi(nphi)`: dependent variable ($\bar{v}$, $\bar{v}$, ...)

`phio(nphi)`: dependent variable ($\bar{v}$, $\bar{v}$, ...) at old time step

`prt(nphi)`: σ_Φ, turbulent Prandtl number

`prt_lam(nphi)`: σ_Φ, viscous Prandtl number

`qfluxh`: constant heat flux at canopy top $[Km/s]$ (only for ABL)

`resor(nphi)`: scaling of residuals

`restrt`: if true, read the flowfield from file 'savres'

`rrl:` used to compute `cdiss` (only for ABL)

`save:` if true, save the flowfield to file 'savres' (it overwrites old file)

`scheme:` discretization scheme for $\bar{v}$, $\bar{v}$, $\bar{w}$, $\bar{\theta}$ ('c', CDS; 'h', hybrid; 'v', van Leer; 'm', MUSCL)

`schemet:` discretization scheme for temperature

`schtur:` discretization scheme for k, ε, ω

`small:` 1e-20

`solve(nphi):` true if `phi` is solved for

`sormax:` global convergence limit in each time step

`sp,su:` discretization source terms, S_p, S_U

`spvw:` discretization S_P source terms in $\bar{v}$ and $\bar{w}$ equations (used for van Leer and MUSCL)

`stability:` is set to true if the temperature equation should be solved for (only for ABL)

`steady:` if true, steady

`suv,suw:` discretization S_U source terms in $\bar{v}$ and $\bar{w}$ equations (used for van Leer and MUSCL)

`time:` current time

`tnoll:` reference temperature ($T_0 = 300K$) (only for ABL)

`u,v,w,p,pp,te,ed,om:` integers used in `phi`, `prt`, ... ($\bar{v}$, $\bar{v}$, $\bar{w}$, $\bar{p}$, p', k, ε, ω)

`u_geo,w_geo:` the desired wind speed magnitude in u and w (only for ABL)

`urfvis:` under-relaxation factor for turbulent viscosity

`vis:` effective dynamic viscosity, $\mu_{eff} = \mu + \mu_t$:

`viscos:` dynamic viscosity, μ

`vol:` volume of cell

`volextcoef:` volumetric expansion coefficient of air ($1/T_0$) (only for ABL)

`wale:` if true, WALE model

`xc,yc,zc:` X_C, Y_C, Z_C, coordinates of east-north-high corner of cell

`xp,yp,zp:` X_P, Y_P, Z_P, cell center coordinates in x, y and z direction

`ynoll:` surface roughness at ground [30] (only for ABL)

References

[1] ABE, K., KONDOH, T., AND NAGANO, Y. A new turbulence model for predicting fluid flow and heat transfer in separating and reattaching flows - 1. Flow field calculations. *Int. J. Heat Mass Transfer 37*, 1 (1994), 139–151.

[2] BILLSON, M. *Computational Techniques for Turbulence Generated Noise*. PhD thesis, Dept. of Thermo and Fluid Dynamics, Chalmers University of Technology, Göteborg, Sweden, 2004.

[3] BILLSON, M., ERIKSSON, L.-E., AND DAVIDSON, L. Jet noise prediction using stochastic turbulence modeling. AIAA paper 2003-3282, 9th AIAA/CEAS Aeroacoustics Conference, 2003.

[4] BURNS, A. D., AND WILKES, N. S. A finite difference method for the computation of fluid flow in complex three-dimensional geometries. Aere r 12342, Harwell Laboratory, U.K., 1986.

[5] DAVIDSON, L. Matlab scriptt for synthetic fluctuations.

[6] DAVIDSON, L. LES of recirculating flow without any homogeneous direction: A dynamic one-equation subgrid model. In *2nd Int. Symp. on Turbulence Heat and Mass Transfer* (Delft, 1997), K. Hanjalić and T. W. J. Peeters, Eds., Delft University Press, pp. 481–490.

[7] DAVIDSON, L. Hybrid LES-RANS: Inlet boundary conditions. In *3rd National Conference on Computational Mechanics – MekIT'05 (invited paper)* (Trondheim, Norway, 2005), B. Skallerud and H. I. Andersson, Eds., pp. 7–22.

[8] DAVIDSON, L. Hybrid LES-RANS: Inlet boundary conditions for flows including recirculation. In *5th International Symposium on Turbulence and Shear Flow Phenomena* (27-29 August, Munich, Germany, 2007), vol. 2, pp. 689–694.

[9] DAVIDSON, L. Hybrid LES-RANS: Inlet boundary conditions for flows with recirculation. In *Second Symposium on Hybrid RANS-LES Methods* (Corfu island, Greece, 2007).

[10] DAVIDSON, L. Inlet boundary conditions for embedded LES. In *First CEAS European Air and Space Conference* (10-13 September, Berlin, 2007).

[11] DAVIDSON, L. Using isotropic synthetic fluctuations as inlet boundary conditions for unsteady simulations. *Advances and Applications in Fluid Mechanics 1*, 1 (2007), 1–35.

[12] DAVIDSON, L. HYBRID LES-RANS: Inlet boundary conditions for flows with recirculation. In *Advances in Hybrid RANS-LES Modelling*, vol. 97 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*. Springer Verlag, 2008, pp. 55–66.

[13] DAVIDSON, L. Fluid mechanics, turbulent flow and turbulence modeling. Division of Fluid Dynamics, Dept. of Applied Mechanics, Chalmers University of Technology, Gothenburg, 2014.

[14] DAVIDSON, L. Zonal PANS: evaluation of different treatments of the RANS-LES interface. *Journal of Turbulence 17*, 3 (2016), 274–307.

[15] DAVIDSON, L., AND BILLSON, M. Hybrid LES/RANS using synthesized turbulent fluctuations for forcing in the interface region. *International Journal of Heat and Fluid Flow 27*, 6 (2006), 1028–1042.

[16] DAVIDSON, L., AND FARHANIEH, B. CALC-BFC: A finite-volume code employing collocated variable arrangement and cartesian velocity components for computation of fluid flow and heat transfer in complex three-dimensional geometries. Rept. 95/11, Dept. of Thermo and Fluid Dynamics, Chalmers University of Technology, Gothenburg, 1995.

[17] DAVIDSON, L., AND FRIESS, C. The PANS and PITM model: a new formulation of f_k. In *Proceedings of 12th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements (ETMM12), Montpelier, i France 26-28 September* (2018).

[18] DAVIDSON, L., AND PENG, S.-H. Hybrid LES-RANS: A one-equation SGS model combined with a $k - \omega$ model for predicting recirculating flows. *International Journal for Numerical Methods in Fluids 43* (2003), 1003–1018.

[19] DAVIDSON, L., AND PENG, S.-H. Embedded large-eddy simulation using the partially averaged Navier-Stokes model. *AIAA Journal 51*, 5 (2013), 1066–1079.

[20] DEARDORFF, J. W. Stratocumulus-capped mixed layers derived from a three-dimensional model. *Boundary-Layer Meteorology 18*, 4 (1980), 495–527.

[21] EMVIN, P. *The Full Multigrid Method Applied to Turbulent Flow in Ventilated Enclosures Using Structured and Unstructured Grids*. PhD thesis, Dept. of Thermo and Fluid Dynamics, Chalmers University of Technology, Göteborg, 1997.

[22] HINZE, J. O. *Turbulence*, 2nd ed. McGraw-Hill, New York, 1975.

[23] IRANNEZHAD, M. DNS of channel flow with finite difference method on a staggered grid. Msc thesis, Division of Fluid Dynamics, Department of Applied Mechanics, Chalmers University of Technology, Göteborg, Sweden, 2006.

[24] JARRIN, N., BENHAMADOUCHE, S., LAURENCE, D., AND PROSSER, R. A synthetic-eddy-method for generating inflow conditions for large-eddy simulations. *International Journal of Heat and Fluid Flow 27*, 4 (2006), 585–593.

[25] MA, J., PENG, S.-H., DAVIDSON, L., AND WANG, F. A low Reynolds number variant of Partially-Averaged Navier-Stokes model for turbulence. *International Journal of Heat and Fluid Flow 32*, 3 (2011), 652–669. 10.1016/j.ijheatfluidflow.2011.02.001.

[26] NEBENFÜHR, B., AND DAVIDSON, L. INFLUENCE OF A FOREST CANOPY ON THE NEUTRAL ATMOSPHERIC BOUNDARY LAYER - A LES STUDY. In *Proceedings of 10th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements (ETMM10), Marbella, Spain, 17-19 September* (2014).

[27] NEBENFHR, B., AND DAVIDSON, L. Large-eddy simulation study of thermally stratified canopy flow. *Boundary-Layer Meteorology* (2015), 1–24.

[28] NEBENFHR, B., AND DAVIDSON, L. Prediction of wind-turbine fatigue loads in forest regions based on turbulent les inflow fields. *Wind Energy* (2016). we.2076.

[29] NICOUD, F., AND DUCROS, F. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow, Turbulence and Combustion 62*, 3 (1999), 183–200.

[30] SHAW, R. H., AND SCHUMANN, U. Large-eddy simulation of turbulent flow above and within a forest. *Boundary-Layer Meteorology 61* (1992), 47 – 64.

[31] SMAGORINSKY, J. General circulation experiments with the primitive equations. *Monthly Weather Review 91* (1963), 99–165.

[32] VAN LEER, B. Towards the ultimate conservative difference scheme. Monotonicity and conservation combined in a second order scheme. *Journal of Computational Physics 14*, 4 (1974), 361–370.

[33] VAN LEER, B. Towards the ultimate conservative difference scheme. V. A second-order sequel to godonov's method. *Journal of Computational Physics 32* (1979), 101–136.

[34] WALLIN, S., AND JOHANSSON, A. V. A new explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *Journal of Fluid Mechanics 403* (2000), 89–132.

[35] WELTY, J. R., WICKS, C. E., AND WILSON, R. E. *Fundamentals of Momentum, Heat, and Mass Transfer*, 3 ed. John Wiley & Sons, New York, 1984.

[36] WILCOX, D. C. Reassessment of the scale-determining equation. *AIAA Journal 26*, 11 (1988), 1299–1310.