EasyCFD

User's Manual

António Manuel Gameiro Lopes

2022

A1.1 – General Equation 3
A1.2 – The Navier-Stokes Equations 3
A1.3 – Continuity Equation 4
A1.4 – Energy Conservation Equation 4
A1.5 – Porous Losses 5
A1.5.1 – Isotropic losses 5
A1.5.2 – Directional losses 5
A1.6 – Transport of Passive Scalars 6
A1.7 – Multi-component Fluid Flow 6
A1.8 – Turbulence Modelling 7
A1.8.1 – The *k-ε* turbulence model 7
A1.8.2 – The SST turbulence model 10
A1.9 – The Buoyancy Term 12
A1.9.1 – Non-isothermal single component flow with the Boussinesq approximation 12
A1.9.2 – Non-isothermal single component flow without the Boussinesq approximation 13
A1.9.3 – Isothermal multi-component component flow 13
A1.9.4 – Non-isothermal multi-component component flow with the Boussinesq approximation 13
A1.10 – 3D Axisymmetric form for transport equations 13
A2.1 – Transformation of Coordinates 15
A2.2 – Integration 19
A2.2.1 – Space integration 19
A2.2.2 – Time integration 20
A2.2.3 – Advection schemes 21
A2.3 – Pressure-Velocity coupling 26
A2.4 – Velocity Correction 26
A2.5 – The Pressure Correction Equation 28
A3.1 – The *p'* Equation 31
A3.2 – Pressure Values 31
A3.3 – Momentum 31
A3.3.1 – Outlet boundaries and symmetry boundaries 31
A3.3.2 – Walls 31
A3.4 – Energy 32
A3.4.1 – Outlet boundaries and symmetry boundaries 32
A3.4.2 – Walls 32
A3.5 – Turbulence Kinetic Energy and Dissipation Rate (k-ε model) 33
A3.5.1 – Outlet boundaries and symmetry boundaries 33
A3.5.2 – Walls 34
A3.6 – Other Scalars 34
A3.6.1 – Outlet boundaries and symmetry boundaries 34
A3.6.2 – Walls 34
A4.1 – Relaxation of the Equations Solution 35
A4.2 – Criterion for Stopping the Solution of the *p´ Equations* 35
A4.3 – Convergence Criteria 36
A4.3.1 – Normalized residual for the mass conservation equation 36
A4.3.2 – Normalized residual for the transport equations 36
A4.4 – The *SIMPLEC* Algorithm 36
A5.1 – 1D algebraic mesh generation on the domain boundaries 38
A5.2 – 2D numerical generation of structured mesh in the inner domain 40
A5.2.1 – Control functions: Clustering and orthogonality at boundaries 41

A5.2.2 – Control functions: Transmission of the mesh spacing into the interior of the domain 44
A5.2.3 – Solution of the equations 45
A5.3 – 2D numerical generation of unstructured mesh in the inner domain 46
A5.3.1 – Row projection 47
A5.3.2 – Seaming of rows 50
A5.3.3 – Intersection of two rows 50
A5.3.4 – Self intersection of a row 50
A5.3.5 – Control of mesh size 51
B1.1 – Brief Description of the Interface 55
B1.2 – Geometry Definition 57
B1.2.1 – Choosing 2D or 3D axisymmetric problems 57
B1.2.2 – Drawing Aids 57
B1.2.3 – Drawing Tools 58
B1.2.4 – Transform Tools 61
B1.2.5 – Connections for U-Type and C-Type meshes 63
B1.2.6 – Closed- and open-blocks; other general drawing concepts 63
B1.2.7 – Importing DXF data 66
B1.2.8 – Importing data from Point Data File 66
B1.2.9 – Final auto-verification before meshing 67
B1.3 – Mesh Generation: Structured type 70
B1.3.1 – Computational Domain; Concept of Computational Direction 71
B1.3.2 – Mesh distribution on the boundaries (Elements - Boundary Nodes tab) 73
B1.3.3 – Nodes clustering (Elements – N. Clustering tab) 73
B1.3.4 – More on the Computational Domain. The Comp. Map Tab 77
B1.3.5 – The concept of Closed Computational Domain 79
B1.3.6 – Blocks Computational Position and Blocks Meshing: The Blocks Tab 80
B1.3.7 – Control functions settings: The CF´s Tab 82
B1.3.8 – Numerical solution control and mesh quality: The Conv./Mesh Quality 82
B1.3.9 – Other menus 83
B1.3.10 – Using C-Type and U-Type meshes 84
B1.3.11 – Using split meshes 86
B1.3.12 – Some advices on mesh generation 86
B1.4 – Mesh Generation: Unstructured type 89
B1.4.1 – Inner Mesh Size 89
B1.4.2 – Inflation layers 89
B1.4.3 – Settings 91
B1.4.4 – Smoothing the mesh 91
B1.4.5 – Increasing mesh resolution through interpolation; multigrid. 91
B1.4.6 – Some advices on unstructured mesh generation 92
B1.5 – Physics and Properties 94
B1.5.1 – Main settings 94
B1.5.2 – Materials 94
B1.5.3 – Energy 94
B1.5.4 – Turbulence model 95
B1.5.5 – Reference pressure 95
B1.5.6 – Multi-component mixture properties 95
B1.6 – Regions and Boundary Conditions 95
B1.6.1 – *Wall* boundary condition 96
B1.6.2 – *Inlet* boundary condition 97
B1.6.3 – *Outlet* boundary condition 99

B1.6.4 – Symmetry boundary condition 99
B1.6.5 – *Lateral opening* boundary condition 99
B1.6.6 – *Pervious line* boundary condition 100
B1.6.7 – Void blocks 101
B1.6.8 – Pervious blocks 101
B1.6.9 – Solid blocks 102
B1.6.10 – Pervious line blocks and thin plates 102
B1.6.11 – Dummy boundary condition 102
B1.6.12 – Space/Time dependent boundary values: *TableFunctions* 102
B1.8 – Calculation Parameters 104
B1.8.1 – Iterations and Convergence 104
B1.8.2 – Sub-relaxation 104
B1.8.3 – The *p'* solution 105
B1.8.4 – Differencing schemes 105
B1.8.5 – Parameters for the turbulence model 106
B1.8.6 – Parameters for the energy equation 106
B1.8.7 – Parameters for transient problems 106
B1.9 – Initialisation of Variables 107
B1.10 – Running in batch 108
B2.1 – Listing of Monitoring Values and Residuals 109
B2.1.1 – Steady-state problems 109
B2.1.2 – Transient problems 110
B2.1.3 – Multigrid calculation 111
B2.2 – Graphical Representation of the Residuals 111
B2.3 – Vectorial Representation of the Velocity Field 111
B2.4 – Location of Maximum Residuals 111
B2.5 – Information messages 112
B3.1 – Example Problem 113
B3.2 – General Description 114
B3.3 – State File, Colour Settings and Visualisation Control 114
B3.4 – Contours 115
B3.4.1 – Shaded contours 115
B3.4.2 – Line contours 116
B3.5 – Vectors 116
B3.6 – Streamlines 117
B3.7 – Analysis 118
B3.8 – Representation of the Mesh 120
B3.9 – Data Export 120
B3.9.1 – Line 120
B3.9.2 – Selected region 122
B3.10 – Analysis of Transient Regime Problems 123
B3.10.1 – Single time instant 124
B3.10.2 – All time instants 124

Introduction

EasyCFD is a software for the numerical solution of fluid flow and heat transfer in two-dimensional geometries described by quadrilateral meshes (structured and non-structured). Support is also provided for 3D axisymmetric situations. The software deals with stationary and transient regime, laminar and turbulent flow (k-ε and *SST* k-ω turbulence models are available), including thermal effects and heat conduction in solids, dispersion of passive scalars, porous losses and multi-component fluid flow.

The present document is divided in three parts: the first part describes the main theoretical principles (not available in the trial version), while the second part is a user's guide to the software. Some computational results and comparisons with published data are presented, for illustration purposes, in part three.

The following Projects are supplied with the standard **EasyCFD** installation:

- ***Annulus***: natural convection laminar flow in an annular cavity. Tutorial movie available at www.easycfd.net;
- ***BMW2002***: turbulent flow around a car shape; structured mesh;
- ***BMW2002NST***: turbulent flow around a car shape; unstructured mesh;
- ***CHT***: conjugate heat transfer problem in transient regime. Tutorial movie available at www.easycfd.net;
- ***ConvLam***: natural convection laminar flow in a square cavity. Benchmark problem described in section C4;
- ***Cylinder***: turbulent flow around a cylinder (low resolution). Tutorial movie available at www.easycfd.net;
- ***LaminarProfile***: developed laminar profile between two parallel plates. Benchmark problem described in section C1;
- ***Naca2412_NST***: turbulent flow around an airfoil, computed on an unstructured mesh. Benchmark problem described in section C7;
- ***PulsatileFlow***: laminar oscillatory flow inside a wavy channel. Benchmark problem described in section C5;
- ***Step***: turbulent flow above a backward facing step. Benchmark problem described in section C3;
- ***C-Type Mesh***: demonstration of the utilization of a C-Type mesh.
- ***Split Mesh***: demonstration of the utilization of a split mesh and thin plates.
- ***Sphere***: calculation of the 3D axisymmetric flow around a sphere. Benchmark problem described in section C6B

- ***TurbulentProfile***: developed turbulent profile between two parallel plates. Benchmark problem described in section C2.

A – Theoretical Background

A1 – Transport Equations

For the description of the transport equations, the *x* and *z* coordinates of the Cartesian coordinate system will be taken as the independent variables. Transformations due to the generalized mesh will be described in section A2.

A1.1 – General Equation

The fundamental equations of fluid flow and heat transfer result from the application of thermodynamic laws to the moving fluid. The two-dimensional form of the transport equation for a generic variable ϕ is, in a Cartesian coordinate system is:

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{A}+\underbrace{\frac{\partial}{\partial x}(\rho u\phi)+\frac{\partial}{\partial z}(\rho w\phi)}_{B}=\underbrace{\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial z}\left(\Gamma\frac{\partial\phi}{\partial z}\right)}_{C}+\underset{D}{S_\phi} \tag{1.1}$$

where ρ [kg/m^3] is the fluid density, *u* and *w* [*m/s*] are the velocity components along the horizontal axis *x* [*m*] and the along the vertical axis *z* [*m*], *t* [*s*] represents time, Γ is the diffusion coefficient of ϕ and S_ϕ is the generation rate of ϕ per unit volume.

The interpretation of the different terms of the previous equation is:

A – transient term

B – advection term

C – diffusion term

D – source term

A1.2 – The Navier-Stokes Equations

If ϕ is the velocity, equation (1.1) represents the momentum conservation. In this case, we get the Navier-Stokes equations, which, for a 2D situation, may be stated as follows:

Horizontal component:

$$\frac{\partial(\rho u)}{\partial t}+\frac{\partial}{\partial x}\left(\rho u^2\right)+\frac{\partial}{\partial z}(\rho uw)=\frac{\partial}{\partial x}\left[\Gamma\left(2\frac{\partial u}{\partial x}-\frac{2}{3}div\vec{V}\right)\right]+\frac{\partial}{\partial z}\left[\Gamma\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]-\frac{\partial p}{\partial x}+S_{Lx} \tag{1.2}$$

or:

$$\frac{\partial(\rho u)}{\partial t}+\frac{\partial}{\partial x}\left(\rho u^2\right)+\frac{\partial}{\partial z}(\rho uw)=\frac{\partial}{\partial x}\left(\Gamma\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial z}\left(\Gamma\frac{\partial u}{\partial z}\right)-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\Gamma\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial z}\left(\Gamma\frac{\partial w}{\partial x}\right)-\frac{2}{3}\frac{\partial}{\partial x}\left(div\vec{V}\right)+S_{Lx} \tag{A1.2b}$$

Vertical component:

$$\frac{\partial(\rho w)}{\partial t}+\frac{\partial}{\partial x}(\rho uw)+\frac{\partial}{\partial z}\left(\rho w^2\right)=\frac{\partial}{\partial z}\left[\Gamma\left(2\frac{\partial w}{\partial z}-\frac{2}{3}div\vec{V}\right)\right]+\frac{\partial}{\partial x}\left[\Gamma\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]-\frac{\partial p}{\partial z}+I+S_{Lz} \tag{1.3}$$

or

$$\frac{\partial(\rho w)}{\partial t}+\frac{\partial}{\partial x}(\rho uw)+\frac{\partial}{\partial z}\left(\rho w^2\right)=\frac{\partial}{\partial x}\left(\Gamma\frac{\partial w}{\partial x}\right)+\frac{\partial}{\partial z}\left(\Gamma\frac{\partial w}{\partial z}\right)-\frac{\partial p}{\partial z}+\frac{\partial}{\partial x}\left(\Gamma\frac{\partial u}{\partial z}\right)+\frac{\partial}{\partial z}\left(\Gamma\frac{\partial w}{\partial z}\right)-\frac{2}{3}\frac{\partial}{\partial z}\left(div\vec{V}\right)+I+S_{Lz} \tag{1.3b}$$

In the previous equations, p [N/m^2] represents pressure, k is the turbulence kinetic energy [m^2/s^2] (cf. section A1.7), I represents the buoyancy forces (cf. section A1.8) and S_L represents porous losses (cf. section ###). The diffusion coefficient is, in this case, given by:

$$\Gamma=\mu+\mu_t \tag{1.4}$$

where μ [$N\,s/m^2$] is the dynamic viscosity and μ_t is the turbulent viscosity (*c.f.* section A1.7).

A1.3 – Continuity Equation

The conservation of mass law, or continuity equation, may be stated as:

$$\frac{\partial(\rho u)}{\partial t}+\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{\partial z}(\rho w)=0 \tag{1.5}$$

A1.4 – Energy Conservation Equation

In this case, the dependent variable is the enthalpy $\phi=c_p T$, where c_p [$J/kg\,K$] is the fluid heat capacity and T [K] is its temperature.

Fluid Medium:

The transport equation is the following:

$$\frac{\partial\left(\rho c_p T\right)}{\partial t}+\frac{\partial}{\partial x}\left(\rho c_p uT\right)+\frac{\partial}{\partial z}\left(\rho c_p wT\right)=\frac{\partial}{\partial x}\left(\Gamma\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial z}\left(\Gamma\frac{\partial T}{\partial z}\right)+S_T \tag{1.6}$$

Term S_T, in the previous equation, represents the heat generation rate per unit volume [W/m^3]. The diffusion coefficient is, for the case of a fluid domain:

$$\Gamma = \left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t} \right) c_p \tag{1.7}$$

where Pr and Pr_t are the laminar and the turbulent Prandtl numbers, respectively.

Solid Medium:

In the case of a solid medium, heat transfer is governed by conduction:

$$\frac{\partial \left(\rho c_p T \right)}{\partial t} = \frac{\partial}{\partial x} \left(\lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial z} \left(\lambda \frac{\partial T}{\partial z} \right) + S_T \tag{1.8}$$

where λ [W/mK] is the thermal conductivity of the material.

A1.5 – Porous Losses

A1.5.1 – Isotropic losses

The momentum loss through porous media is formulated using through the viscous permeability K_p and the quadratic loss (pressure loss) C, as follows:

$$S_{Li} = -\frac{\mu}{K_p} u_i - \frac{\rho}{2} C |V| u_i \tag{1.9}$$

where $u_i = u, v, w$ for $I = 1, 2, 3$.

A1.5.2 – Directional losses

The momentum loss through an anisotropic region is modelled by specifying the streamwise direction φ (minimum loss direction) and a transverse multiplier factor F_t. For convenience, the viscous term is formulated now in terms of viscous loss (instead of permeability), $D = \mu / K_p$.

The source term now reads:

$$S_{Li} = -\sum_{j=1}^{3} D_{ij} u_j - \frac{\rho}{2} \sum_{j=1}^{3} C_{ij} |V| u_j \tag{1.10}$$

The loss coefficients matrices are defined as follows (exemplified for the viscous term):

$$D_{ij} = \begin{bmatrix} D_1 \cos^2 \varphi + D_2 \sin^2 \varphi & 0.5 \left(D_1 - D_2 \right) \sin 2\varphi \\ 0.5 \left(D_1 - D_2 \right) \sin 2\varphi & D_1 \sin^2 \varphi + D_2 \cos^2 \varphi \end{bmatrix} \tag{1.11}$$

where D_1 is the streamwise loss coefficient and D_2 is the transverse loss coefficient, that is obtained using the transverse factor multiplier:

$$D_2 = F_t \, D_1 \tag{1.12}$$

A1.6 – Transport of Passive Scalars

Passive scalars represent a transported additional variable that has no influence on the flow. The presence of passive scalars is quantified by their concentration ϕ_{sc}, which represents the mass of scalar per mass of fluid. The transport equation is:

$$\frac{\partial(\rho\phi_{sc})}{\partial t} + \frac{\partial}{\partial x}(\rho u \phi_{sc}) + \frac{\partial}{\partial z}(\rho w \phi_{sc}) = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi_{sc}}{\partial x}\right) + \frac{\partial}{\partial z}\left(\Gamma \frac{\partial \phi_{sc}}{\partial z}\right) + S_{\phi_{sc}} \tag{1.13}$$

where $S_{\phi_{sc}}$ is the volumetric source term (mass of scalar generated per unit fluid volume per unit time) and the diffusion coefficient is given by:

$$\Gamma = \rho D_{\phi_{sc}} + \frac{\mu_t}{Sc_t} \tag{1.14}$$

In the previous equation, ρ is the fluid density, $D_{\phi_{sc}}$ [m^2/s] is the kinematic diffusivity for the scalar in the presence of the operating fluid and Sc_t is the turbulence Schmidt number.

A1.7 – Multi-component Fluid Flow

In a multi-component fluid flow, different fluids are present, sharing the same velocity, pressure, temperature and turbulence quantities. **EasyCFD** contemplates the case of two fluids, with a single phase. The concentration of one of the components (the secondary fluid) is computed with a normal transport equation for a scalar:

$$\frac{\partial(\rho\phi_{c2})}{\partial t} + \frac{\partial}{\partial x}(\rho u \phi_{c2}) + \frac{\partial}{\partial z}(\rho w \phi_{c2}) = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi_{c2}}{\partial x}\right) + \frac{\partial}{\partial z}\left(\Gamma \frac{\partial \phi_{c2}}{\partial z}\right) \tag{1.15}$$

where ϕ_{c2} is the mass fraction, or concentration, of the secondary fluid (mass of secondary fluid per mass of mixture). The diffusion coefficient is given by

$$\Gamma = \rho D_{\phi_c} + \frac{\mu_t}{Sc_t} \tag{1.16}$$

where D_{ϕ_c} [m^2/s] is the kinematic diffusivity that characterizes the fluids pair. No volumetric generation is contemplated, in **EasyCFD**, for multi-component fluid flow.

Unlike passive scalars, the concentration of the secondary fluid plays an active role in the flow field, as fluid properties depend on the concentration of each component. The mixture properties (such

as viscosity, heat capacity, etc.) are determined as a weighted average of the properties of each component. Thus, for a generic property X, we have:

$$X = \phi_{c1} X_1 + \phi_{c2} X_2 \tag{1.17}$$

Exception is made for density. A simple analysis may prove that the mixture density ρ is given by:

$$\rho = \left(\frac{\phi_{c1}}{\rho_1} + \frac{\phi_{c2}}{\rho_2} \right)^{-1} \tag{1.18}$$

Evidently, the following constrain applies:

$$\phi_{c1} + \phi_{c2} = 1 \tag{1.19}$$

A1.8 – Turbulence Modelling

The properties of a turbulent flow (velocity, pressure, etc.) are not constant in time; instead, they present oscillations about an average value. The numerical calculation of the instantaneous value is not amenable with present day techniques and resources, due to the high temporal and spatial frequencies that characterize these flows. We are, thus, left with the calculation of the average values, only. These can be described through the Reynolds decomposition:

$$\hat{\phi} = \phi + \phi' \tag{1.20}$$

where $\hat{\phi}$ is the instantaneous value, ϕ is the average value and ϕ' is the difference between the two (fluctuation). When the Reynolds decomposition is applied to the transport equations and the equations are the averaged, some extra terms appear, due to the following property (illustrated for the third term of equation (1.2)):

$$\rho \overline{u\, w} = \rho \bar{u}\, \bar{w} + \rho \overline{u'\, w'} \tag{1.21}$$

The last term in the previous equation has the dimensions of a stress and thus, can be expressed as the product of a viscosity (turbulent viscosity) by an average velocity gradient, as in the case of the "laminar" stresses (this hypothesis was proposed by Boussinesq). The computation of the turbulent viscosity is made recurring to a turbulence model. EasyCFD implements the k-ε and the STT (Shear Stress Transport) models, as described next.

A1.8.1 – The k-ε turbulence model

The standard formulation of this turbulence model is described in Launder and Spalding, 1972, Launder and Spalding, 1974, Djilali et al., 1989. The turbulent viscosity is given by:

$$\mu_t = C_\mu \frac{\rho k^2}{\varepsilon} \tag{1.22}$$

The turbulence kinetic energy, k, as well as its dissipation rate, ε [m^2/s^3], are computed with the following transport equations:

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial}{\partial x}(\rho u k)+\frac{\partial}{\partial z}(\rho w k)=\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x}\right]+\frac{\partial}{\partial z}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial z}\right]+P_k-\rho\varepsilon+G_T \qquad (1.23)$$

$$\frac{\partial(\rho \varepsilon)}{\partial t}+\frac{\partial}{\partial x}(\rho u \varepsilon)+\frac{\partial}{\partial z}(\rho w \varepsilon)=\frac{\partial}{\partial x}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x}\right]+\frac{\partial}{\partial z}\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial z}\right]+\frac{\varepsilon}{k}\left(C_1P_k-C_2\rho\varepsilon+C_3G_T\right) \qquad (1.24)$$

The term P_1 represents the production rate of k as the results of the velocity gradients:

$$P_k=\mu_t\left[2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial w}{\partial z}\right)^2+\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)^2\right] \qquad (1.25)$$

while the term G_T accounts for the production or destruction of k and ε due to the thermal gradients:

$$G_T=-\beta g\frac{\mu_t}{Pr_t}\frac{\partial T}{\partial z} \qquad (1.26)$$

The remaining model constants are:

$$C_\mu=0.09 \qquad \sigma_k=1.0 \qquad \sigma_\varepsilon=1.3 \qquad C_1=1.44 \qquad C_2=1.92 \qquad C_3=1.44 \qquad (1.27)$$

Wall laws

In the proximity of a wall, the previous equations should be modified account for the viscous effects that become predominant. The utilisation of a low-Reynolds number turbulence model represents the best approach to this problem. In this case a fine mesh resolution near the wall must be ensured, with mesh nodes inside the viscous sub-layer. In this approach, damping functions are employed as a way of modelling the effects of viscosity upon the near-wall flow. The alternative, which is of easier implementation and does not require as a fine mesh, consists on the utilisation of wall functions, which ensure the connection between the viscous sub-layer and the inertia layer. The distinction between these two layers is given by the y^+, value:

$$y^+=\frac{\rho u_\tau y}{\mu}=\frac{\rho C_\mu^{0.25}\sqrt{k}\,y}{\mu} \qquad (1.28)$$

where u_τ represents the friction velocity and y is the distance to the wall. Thus, we have:

$y^+ \le 11.63 \quad \Rightarrow$ viscous sub-layer

$y^+ > 11.63 \quad \Rightarrow$ inertia sub-layer

The utilisation of wall laws does not require mesh nodes in the viscous sub-layer, which is a major advantage from the computational standpoint.

Wall laws for momentum

The momentum flux per unit area along the direction normal to the wall is given by the wall shear stress, which is computed differently depending on the location of the wall neighbour node relatively to the transition between the viscous and the inertia sub-layers. Denoting by *v* the generic velocity component parallel to the wall (which may, actually, be *u* or *w*) and by *y* the distance to the wall (which may, actually, be *z* or *x*), we have:

$$y^+ \le 11.63 \quad \Rightarrow \quad \tau_0 = \mu \frac{v_0 - v}{y} \tag{1.29}$$

$$y^+ > 11.63 \quad \Rightarrow \quad \tau_0 = \frac{\rho C_\mu^{0.25} \chi \sqrt{k}}{ln\left(E^* y^+\right)} \left(v_0 - v\right) \tag{1.30}$$

where v_0 is the wall velocity, $E^* = 9.793$ for smooth walls and $\chi = 0.4187$ is the von Karman constant.

Wall laws for turbulence kinetic energy

In the wall neighbourhood, the production term (equation (1.25)) is computed assuming a Couette flow:

$$P_k = \mu_t \left(\frac{\partial v}{\partial y} \right)^2 \tag{1.31}$$

where, once again, *v* is the generic velocity component parallel to the wall and *y* is the generic distance to the wall. Due to its significant variations near the wall, ε is averaged for the calculation of the term $\rho\varepsilon$ in equation (1.23):

$$y^+ \le 11.63 \quad \Rightarrow \quad \rho\varepsilon = \rho C_\mu^{0.75} k^{1.5} \frac{y^+}{y} \tag{1.32}$$

$$y^+ > 11.63 \quad \Rightarrow \quad \rho\varepsilon = \rho C_\mu^{0.75} k^{1.5} \frac{ln\left(E^* y^+\right)}{\chi y} \tag{1.33}$$

For the turbulence kinetic energy, a zero flux along the direction perpendicular to the wall is assigned.

Wall laws for dissipation rate of turbulence kinetic

Equation (1.24) is not employed in the node adjacent to the wall. Instead, the dissipation rate is given by:

$$\varepsilon_0 = \frac{C_\mu^{0.75} k^{1.5}}{\chi y} \tag{1.34}$$

Wall laws for energy

As for momentum, energy flux is computed differently depending on the y^+ values. In this case, we have:

$$y^+ \le 12 \quad \Rightarrow \quad \dot{q} = \frac{\mu c_p}{Pr\, y}\left(T_0 - T\right) \tag{1.35}$$

$$y^+ > 12 \quad \Rightarrow \quad \dot{q} = \frac{\rho c_p C_\mu^{0.25}\sqrt{k}}{Pr_t\left[\frac{1}{\chi} ln\left(E^*\, y^+\right) + P^*\right]}\left(T_0 - T\right) \tag{1.36}$$

where T_0 is the wall temperature and P^* is the so called Jayatillaka function:

$$P^* = 9.24\left[\left(\frac{Pr}{Pr_t}\right)^{0.75} - 1\right]\left[1 + 0.28\, exp\left(-0.007\,\frac{Pr}{Pr_t}\right)\right] \tag{1.37}$$

Wall laws for other scalars

Other scalars are treated similarly as for energy, with the Prandtl number *Pr* replaced by the Schmidt number, *Sc*.

A1.8.2 – The SST turbulence model

The *SST* model (Menter, 1993) represents a combination of the *k*-ε and the *k*-ω models. According to Menter, Kuntz and Langtry, 2003, the *k*-ω model is more accurate near the wall but presents a high sensitivity to the ω values in the free stream region, where the *k*-ε model shows a better behavior. The *SST* model represents a blend of the two, through a weighting factor computed based on the nearest wall distance. The governing equations are:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho u k)}{\partial x} + \frac{\partial(\rho w k)}{\partial z} = \overline{P_k} - \beta^* \rho \omega k + \frac{\partial}{\partial x}\left(\left(\mu + \sigma_k\, \mu_t\right)\frac{\partial k}{\partial x}\right) + \frac{\partial}{\partial z}\left(\left(\mu + \sigma_k\, \mu_t\right)\frac{\partial k}{\partial z}\right) \tag{1.38}$$

and

$$\frac{\partial(\rho \omega)}{\partial t} + \frac{\partial(\rho u \omega)}{\partial x} + \frac{\partial(\rho w \omega)}{\partial z} = \frac{\partial}{\partial x}\left(\left(\mu + \sigma_\omega \mu_t\right)\frac{\partial \omega}{\partial x}\right) + \frac{\partial}{\partial z}\left(\left(\mu + \sigma_\omega \mu_t\right)\frac{\partial \omega}{\partial z}\right) +$$

$$+ \frac{\alpha \overline{P_k}}{\nu_t} - \beta \rho \omega^2 + 2\left(1 - F_1\right)\rho \sigma_{\omega 2}\frac{1}{\omega}\left(\frac{\partial k}{\partial x}\frac{\partial \omega}{\partial x} + \frac{\partial k}{\partial z}\frac{\partial \omega}{\partial z}\right) \tag{1.39}$$

where ω is the frequency of dissipation of turbulent kinetic energy [s^{-1}].

The production of turbulent kinetic energy is limited to prevent the build-up of turbulece in stagnant regions:

$$\overline{P_k} = min\left(P_k, 10\,\beta^* \rho k \omega\right) \tag{1.40}$$

The weighting function F_1 is given by:

$$F_1 = tanh\left\{\left\{min\left[max\left(\frac{\sqrt{k}}{\beta^* \omega y};\frac{500\nu}{y^2\,\omega}\right);\frac{4\,\rho\,\sigma_{\omega 2}\,k}{CD_{k\omega}\,y^2}\right]\right\}^4\right\} \tag{1.41}$$

and

$$CD_{k\omega} = max\left(2\,\rho\,\sigma_{\omega 2}\,\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j},10^{-10}\right) \tag{1.42}$$

where y represents the distance to the neighbour wall and ν is the laminar dynamic viscosity. F_1 is zero away from the wall (k-ε model) and changes to unit inside the boundary layer (k-ω model), with a smooth transition based on y.

The turbulent viscosity computed as:

$$\nu_t = \frac{a_1\,k}{max\left(a_1\,\omega; S\,F_2\right)} \tag{1.43}$$

where S is the invariant measure of the strain rate, given by:

$$S = \sqrt{S_{ij}S_{ij}} \quad ; \quad S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{1.44}$$

and

$$F_2 = tanh\left\{\left[max\left(\frac{2\sqrt{k}}{\beta^* \omega y};\frac{500\nu}{y^2\,\omega}\right)\right]^2\right\} \tag{1.45}$$

The constants are computed as a blend of the k-ε and the k-ε models, through the following generic equation:

$$\alpha = F_1\,\alpha_1 + \left(1 - F_1\right)\alpha_2 \tag{1.46}$$

The constants are:

α_1 = 5/9	β_1 = 3/40	σ_{k1} = 0.85	$\sigma_{\omega 1}$ = 0.5
α_2 = 0.44	β_2 = 0.0828	σ_{k2} = 1	$\sigma_{\omega 2}$ = 0.856
β^* = 0.09			

Near Wall treatment

The near wall treatment for momentum and turbulence equations implemented in EasyCFD follows the proposal described in Menter, Ferreira and Konno, 2003. The basic principle behind the

automatic wall functions is to switch from a low-Reynolds number formulation to a wall function based on the grid nodes proximity to the wall. According to these authors, the automatic wall treatment avoids the deterioration of results typical of low-Reynolds models when applied on too coarse meshes.

The known solutions for ω in the viscous (linear) and in the logarithmic near wall region are:

$$\omega_{vis} = \frac{6\nu}{0.075\, y^2} \;\; ; \;\;\; \omega_{log} = \frac{u_\tau}{0.3\kappa\, y} \tag{1.47}$$

The imposed value for ω at the first node close to a wall is:

$$\omega_1 = \sqrt{\omega_{vis}^2 + \omega_{log}^2} \tag{1.48}$$

For the turbulence kinetic energy, a zero flux along the direction perpendicular to the wall is assigned.

In turn, for the momentum equations, a similar reasoning applies, with expressions for the shear velocity in the viscous and in the logarithmic region:

$$u_\tau^{vis} = \frac{U_1}{y^+} \;\; ; \;\;\; u_\tau^{log} = \frac{\kappa U_1}{ln\left(E\, y^+\right)} \tag{1.49}$$

with U_1 as the fluid velocity relative to the wall velocity. The wall shear stress is computed as follows:

$$u_\tau = \sqrt[4]{\left(u_\tau^{vis}\right)^4 + \left(u_\tau^{log}\right)^4} \tag{1.50}$$

A1.9 – The Buoyancy Term

The buoyancy term in equations (1.3) and (A1.3b) is given by:

$$I = -g\left(\rho - \rho_{ref}\right) \tag{1.51}$$

where ρ_{ref} is a reference density and $g = 9.81 m/s^2$ is the gravity acceleration. The above term is computed differently depending on the problem physics.

A1.9.1 – Non-isothermal single component flow with the Boussinesq approximation

If the thermal amplitude within the calculation domain is relatively small ($\beta \Delta T \ll 1$, with β as the thermal expansion coefficient), it is viable to adopt the Boussinesq approximation, which considers the fluid density as constant, except in its contribution to the buoyancy term. In this case, density variations are approximated as:

$$\rho - \rho_{ref} = -\rho_{ref}\, \beta\left(T - T_{ref}\right) \tag{1.52}$$

and, thus:

$$I = \rho_{ref}\, g\, \beta\left(T - T_{ref}\right) \tag{1.53}$$

In the previous equation, ρ_{ref} is the fluid density at the reference temperature T_{ref}.

A1.9.2 – Non-isothermal single component flow without the Boussinesq approximation

If the Boussinesq approximation is not considered, density variations are accounted for in all the terms of the transport equations and the buoyancy term is computed directly with equation (1.51). This option is available, within **EasyCFD**, for the fluids *Air* and *Water*, for single component fluid flow. Density is computed as:

For air (considered as a perfect gas):

$$\rho = \frac{1.013\times 10^5}{287\left(T\left[^oC\right] + 273.15\right)} \tag{1.54}$$

For water:

$$\rho = 10^3 - 1.78\times 10^{-2}\left|T\left[^oC\right] - 4\right|^{1.7} \tag{1.55}$$

A1.9.3 – Isothermal multi-component component flow

In this case, equation (1.51) is used directly, with the density given by equation (1.18).

A1.9.4 – Non-isothermal multi-component component flow with the Boussinesq approximation

The local density, ρ, entering in equation (1.51), is a function of both temperature and mixture composition. Dependence of density with temperature is computed using the mixture thermal expansion coefficient. Equation (1.51) becomes:

$$I = -g\left[\rho - \rho\, g\, \beta\left(T - T_{ref}\right) - \rho_{ref}\right] \tag{1.56}$$

where the mixture properties ρ and β are given by equation (1.18). The reference density ρ_{ref} is taken as the density of the main fluid, ρ_1.

A1.10 – 3D Axisymmetric form for transport equations

Three-dimensional problems presenting axisymmetry can be solved using a 2D approach, provided the equations are written in a cylindrical coordinate system. Assuming the *x*-*r* coordinate system, where *r* is the radial coordinate, the general transport equation assumes the following form:

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial}{\partial x}(\rho u\,\phi) + \frac{1}{r}\frac{\partial}{\partial r}(r\rho w\phi) = \frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right) + \frac{1}{r}\frac{\partial}{\partial z}\left(r\Gamma\frac{\partial\phi}{\partial z}\right) + S_\phi \tag{1.57}$$

A similar form is obtained for the specific equations for momentum, continuity, energy, etc. In the implementation within EasyCFD, the user can choose the symmetry axis (x or z).

A2 – Numerical Method

A2.1 – Transformation of Coordinates

The following presentation is made for the Cartesian form of the transport equations. The 3D-axissimetric form is treated in a similar way.

Discretisation and integration of the transport equations described previously are performed using a non-orthogonal generalized mesh[1], as shown in Figure 2.1(a). The independent Cartesian coordinates (x, z), describing the physical domain, are, thus, replaced by a boundary fitted coordinate system (ξ, ζ), defined by the mesh lines which may have, locally, any orientation and inclination. The computational domain (as opposed to the physical domain) is the space regarded in terms of the boundary fitted coordinate system coordinates (ξ, ζ), as depicted in Figure 2.1(b). In the computational domain, the mesh spacing is considered, for convenience, as unitary:

$$\Delta\xi = \Delta\zeta = 1 \tag{2.1}$$

and the mesh lines are always horizontal (ξ lines) or vertical (ζ lines). The computational domain describes, in fact, the indexation for the bidimensional arrays storing the dependent and independent values for the variables.

The mesh arrangement is of the collocated type (as opposed to the staggered mesh), with the two velocity components and scalar quantities (temperature, pressure, turbulence kinetic energy and its dissipation rate, as well as concentrations) located at the control volume centre (cf. Figure A2.1)

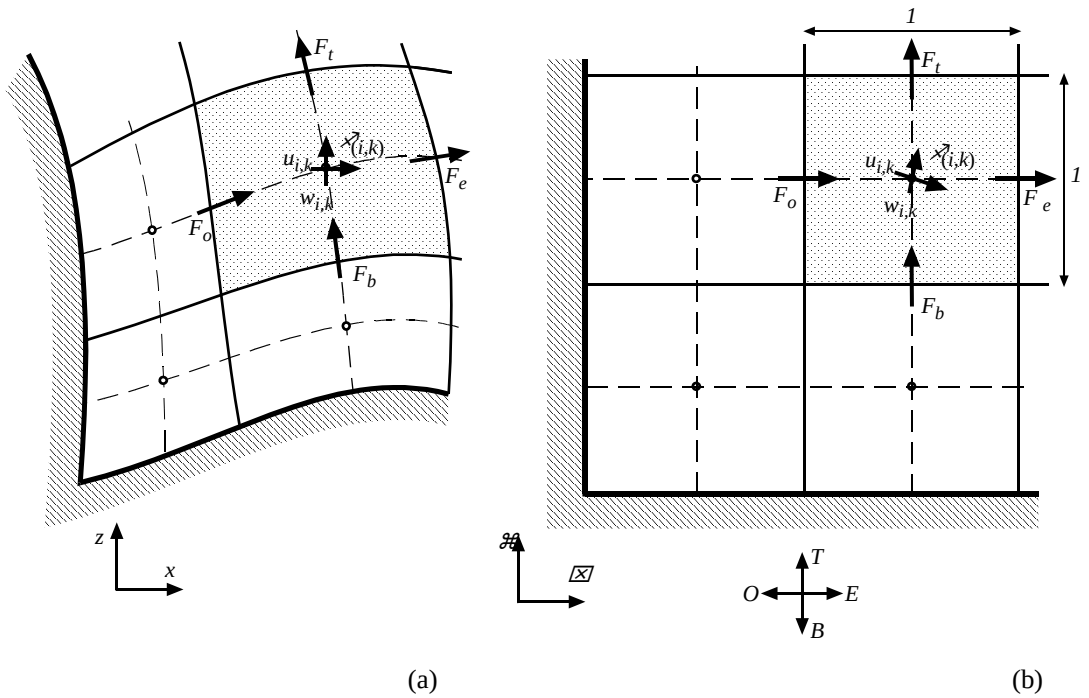


Figure A.1 – The two domains: (a) physical domain, showing the mesh lines. (b) computational domain.

[1] The mesh is the set of geometric locations where the solution (*i.e.*, values for velocity, pressure, etc.) takes place.

Transformation of the original equations is accomplished by replacing the independent variables, using the chain rule, which states that, generically:

$$\frac{\partial \phi}{\partial x} = \frac{\partial \phi}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial \phi}{\partial \zeta}\frac{\partial \zeta}{\partial x} = \xi_x \frac{\partial \phi}{\partial \xi} + \zeta_x \frac{\partial \phi}{\partial \zeta} \tag{2.2}$$

The Jacobian of the transformation:

$$J = \begin{vmatrix} x_\xi & x_\zeta \\ z_\xi & z_\zeta \end{vmatrix} = x_\xi z_\zeta - x_\zeta z_\xi \tag{2.3}$$

represents the ratio between the physical size of the control volume and its computational size (defined as unitary; for convenience, $\Delta\xi = \Delta\zeta = 1$).

The derivatives ξ_x, ξ_z, $\zeta_x \zeta_z$ are the contravariant metrics of the transformation. They are computed from the covariant metrics x_ξ, z_ξ, x_ζ, z_ζ, as follows:

$$\xi_x = \frac{z_\zeta}{J} \;\; ; \;\; \xi_z = -\frac{x_\zeta}{J} \;\; ; \;\; \zeta_x = -\frac{z_\xi}{J} \;\; ; \;\; \zeta_z = \frac{x_\xi}{J} \tag{2.4}$$

The metric identity, given by

$$\frac{\partial}{\partial \xi}\left(J\xi_x\right) + \frac{\partial}{\partial \zeta}\left(J\zeta_x\right) = 0 \quad ; \quad \frac{\partial}{\partial \xi}\left(J\xi_z\right) + \frac{\partial}{\partial \zeta}\left(J\zeta_z\right) = 0 \tag{2.5}$$

is then applied to get the following form for the general transport equation:

General Transport Equation:

To obtain the strong conservative form in the boundary fitted coordinate system (ξ, ζ), the transport equations (1.1) are transformed through the application of the chain rule (2.2) and multiplied by the Jacobian of the transformation. The metric identity (2.5) is then used to recast some terms. The result is:

$$\begin{aligned} &J\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial}{\partial \xi}\left(J\rho U\phi\right) + \frac{\partial}{\partial \zeta}\left(J\rho W\phi\right) = \\ &\frac{\partial}{\partial \xi}\left[\Gamma J g^{11}\frac{\partial \phi}{\partial \xi}\right] + \frac{\partial}{\partial \zeta}\left[\Gamma J g^{33}\frac{\partial \phi}{\partial \zeta}\right] + \frac{\partial}{\partial \xi}\left[\Gamma J g^{13}\frac{\partial \phi}{\partial \zeta}\right] + \frac{\partial}{\partial \zeta}\left[\Gamma J g^{13}\frac{\partial \phi}{\partial \xi}\right] + JS_\phi \end{aligned} \tag{2.6}$$

The terms g^{11}, g^{13} and g^{33} are the contravariant metric relations, given by:

$$g^{11} = \xi_x^2 + \xi_z^2 \quad ; \quad g^{33} = \zeta_x^2 + \zeta_z^2 \quad ; \quad g^{13} = \xi_x\zeta_x + \xi_z\zeta_z \tag{2.7}$$

The non-orthogonal term g^{13} is null if the mesh is locally orthogonal.

U and *W*, in equation (2.6), are the contravariant velocities. The terms $J\rho U$ and $J\rho W$ represent mass fluxes through the control volume faces, along the computational directions ξ and ζ, respectively, and are computed as follows:

$$F_\xi = J\rho U = \rho\left(z_\zeta u - x_\zeta w\right) \tag{2.8}$$

$$F_\zeta = J\rho W = \rho\left(x_\xi w - z_\xi u\right) \tag{2.9}$$

Note that, in this case, the sub-index for the fluxes (such as in F_ξ) represents the flux direction (and not a derivative, such as for the metrics).

Equation (2.6) may be rewritten as:

$$J\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial}{\partial\xi}\left(F_\xi\phi\right) + \frac{\partial}{\partial\zeta}\left(F_\zeta\phi\right) = \frac{\partial}{\partial\xi}\left[\Gamma Jg^{11}\frac{\partial\phi}{\partial\xi}\right] + \frac{\partial}{\partial\zeta}\left[\Gamma Jg^{33}\frac{\partial\phi}{\partial\zeta}\right] + J\left(S_{cross} + S_\phi\right) \tag{2.10}$$

where the cross derivatives were incorporated into the source term S_{cross}.

Navier-Stokes equations:

A similar procedure is applied to obtain the generalized form of the Navier-Stokes equations.

u component:

$$\begin{aligned}
&J\frac{\partial(\rho u)}{\partial t} + \frac{\partial}{\partial\xi}\left(J\rho U u\right) + \frac{\partial}{\partial\zeta}\left(J\rho W u\right) = \frac{\partial}{\partial\xi}\left[\Gamma Jg^{11}\frac{\partial u}{\partial\xi}\right] + \frac{\partial}{\partial\zeta}\left[\Gamma Jg^{33}\frac{\partial u}{\partial\zeta}\right] + \frac{\partial}{\partial\xi}\left[\Gamma Jg^{13}\frac{\partial u}{\partial\zeta}\right] + \frac{\partial}{\partial\zeta}\left[\Gamma Jg^{13}\frac{\partial u}{\partial\xi}\right] + \\
&\frac{\partial}{\partial\xi}\left[\Gamma J\left(\xi_x^2\frac{\partial u}{\partial\xi} + \xi_x\zeta_x\frac{\partial u}{\partial\zeta}\right)\right] + \frac{\partial}{\partial\zeta}\left[\Gamma J\left(\xi_x\zeta_x\frac{\partial u}{\partial\xi} + \zeta_x^2\frac{\partial u}{\partial\zeta}\right)\right] + \\
&\frac{\partial}{\partial\xi}\left[\Gamma J\left(\xi_x\xi_z\frac{\partial w}{\partial\xi} + \xi_x\zeta_z\frac{\partial w}{\partial\zeta}\right)\right] + \frac{\partial}{\partial\zeta}\left[\Gamma J\left(\zeta_x\xi_z\frac{\partial w}{\partial\xi} + \zeta_x\zeta_z\frac{\partial w}{\partial\zeta}\right)\right] - \\
&\frac{2}{3}\frac{\partial}{\partial\xi}\left[\Gamma J\left(\xi_x^2\frac{\partial u}{\partial\xi} + \xi_x\zeta_x\frac{\partial u}{\partial\zeta}\right)\right] - \frac{2}{3}\frac{\partial}{\partial\zeta}\left[\Gamma J\left(\xi_x\zeta_x\frac{\partial u}{\partial\xi} + \zeta_x^2\frac{\partial u}{\partial\zeta}\right)\right] - \\
&\frac{2}{3}\frac{\partial}{\partial\xi}J\xi_x\frac{\partial p}{\partial\xi} - \left[\Gamma J\left(\xi_x\xi_z\frac{\partial w}{\partial\xi} + \xi_x\zeta_z\frac{\partial w}{\partial\zeta}\right)\right] - \frac{2}{3}\frac{\partial}{\partial\zeta}\left[\Gamma J\left(\zeta_x\xi_z\frac{\partial w}{\partial\xi} + \zeta_x\zeta_z\frac{\partial w}{\partial\zeta}\right)\right] - \\
&J\left(\xi_x\frac{\partial p}{\partial\xi} + \zeta_x\frac{\partial p}{\partial\zeta}\right)
\end{aligned} \tag{2.11}$$

Casting the cross derivatives into a single source term, one may write:

$$J\frac{\partial(\rho u)}{\partial t} + \frac{\partial}{\partial\xi}\left(F_\xi u\right) + \frac{\partial}{\partial\zeta}\left(F_\zeta u\right) = \frac{\partial}{\partial\xi}\left[\Gamma Jg^{11}\frac{\partial u}{\partial\xi}\right] + \frac{\partial}{\partial\zeta}\left[\Gamma Jg^{33}\frac{\partial u}{\partial\zeta}\right] - J\left(\xi_x\frac{\partial p}{\partial\xi} + \zeta_x\frac{\partial p}{\partial\zeta}\right) + S_{crossu} \tag{2.12}$$

or

$$J\frac{\partial(\rho u)}{\partial t}+\frac{\partial}{\partial\xi}\left(F_\xi u\right)+\frac{\partial}{\partial\zeta}\left(F_\zeta u\right)=\frac{\partial}{\partial\xi}\left[\Gamma Jg^{11}\frac{\partial u}{\partial\xi}\right]+\frac{\partial}{\partial\zeta}\left[\Gamma Jg^{33}\frac{\partial u}{\partial\zeta}\right]-z_\zeta\frac{\partial p}{\partial\xi}+z_\xi\frac{\partial p}{\partial\zeta}+S_{crossu} \tag{2.13}$$

w component:

$$\begin{aligned}&J\frac{\partial(\rho w)}{\partial t}+\frac{\partial}{\partial\xi}\left(J\rho Uw\right)+\frac{\partial}{\partial\zeta}\left(J\rho Ww\right)=\frac{\partial}{\partial\xi}\left[\Gamma Jg^{11}\frac{\partial w}{\partial\xi}\right]+\frac{\partial}{\partial\zeta}\left[\Gamma Jg^{33}\frac{\partial w}{\partial\zeta}\right]+\frac{\partial}{\partial\xi}\left[\Gamma Jg^{13}\frac{\partial u}{\partial\zeta}\right]+\frac{\partial}{\partial\zeta}\left[\Gamma Jg^{13}\frac{\partial w}{\partial\xi}\right]+\\&\frac{\partial}{\partial\xi}\left[\Gamma J\left(\xi_z^2\frac{\partial w}{\partial\xi}+\xi_z\zeta_z\frac{\partial w}{\partial\zeta}\right)\right]+\frac{\partial}{\partial\zeta}\left[\Gamma J\left(\xi_z\zeta_z\frac{\partial w}{\partial\xi}+\zeta_z^2\frac{\partial w}{\partial\zeta}\right)\right]+\\&\frac{\partial}{\partial\xi}\left[\Gamma J\left(\xi_x\xi_z\frac{\partial u}{\partial\xi}+\xi_x\zeta_z\frac{\partial u}{\partial\zeta}\right)\right]+\frac{\partial}{\partial\zeta}\left[\Gamma J\left(\zeta_x\xi_z\frac{\partial u}{\partial\xi}+\zeta_x\zeta_z\frac{\partial u}{\partial\zeta}\right)\right]-\\&\frac{2}{3}\frac{\partial}{\partial\xi}\left[\Gamma J\left(\xi_z^2\frac{\partial w}{\partial\xi}+\xi_z\zeta_z\frac{\partial w}{\partial\zeta}\right)\right]-\frac{2}{3}\frac{\partial}{\partial\zeta}\left[\Gamma J\left(\xi_z\zeta_z\frac{\partial w}{\partial\xi}+\zeta_z^2\frac{\partial w}{\partial\zeta}\right)\right]-\\&\frac{2}{3}\frac{\partial}{\partial\xi}\left[\Gamma J\left(\xi_x\xi_z\frac{\partial u}{\partial\xi}+\xi_x\zeta_z\frac{\partial u}{\partial\zeta}\right)\right]-\frac{2}{3}\frac{\partial}{\partial\zeta}\left[\Gamma J\left(\zeta_x\xi_z\frac{\partial u}{\partial\xi}+\zeta_x\zeta_z\frac{\partial u}{\partial\zeta}\right)\right]-\\&J\left(\xi_z\frac{\partial p}{\partial\xi}+\zeta_z\frac{\partial p}{\partial\zeta}\right)+I\end{aligned} \tag{2.14}$$

where *I* represents buoyancy forces. Casting the cross derivatives into a single source term, results:

$$J\frac{\partial(\rho w)}{\partial t}+\frac{\partial}{\partial\xi}\left(F_\xi w\right)+\frac{\partial}{\partial\zeta}\left(F_\zeta w\right)=\frac{\partial}{\partial\xi}\left[\Gamma Jg^{11}\frac{\partial w}{\partial\xi}\right]+\frac{\partial}{\partial\zeta}\left[\Gamma Jg^{33}\frac{\partial w}{\partial\zeta}\right]-J\left(\xi_z\frac{\partial p}{\partial\xi}+\zeta_z\frac{\partial p}{\partial\zeta}\right)+S_{crossw}+I \tag{2.15}$$

or

$$J\frac{\partial(\rho w)}{\partial t}+\frac{\partial}{\partial\xi}\left(F_\xi w\right)+\frac{\partial}{\partial\zeta}\left(F_\zeta w\right)=\frac{\partial}{\partial\xi}\left[\Gamma Jg^{11}\frac{\partial w}{\partial\xi}\right]+\frac{\partial}{\partial\zeta}\left[\Gamma Jg^{33}\frac{\partial w}{\partial\zeta}\right]+x_\zeta\frac{\partial p}{\partial\xi}-x_\xi\frac{\partial p}{\partial\zeta}+S_{crossw}+I \tag{2.16}$$

Continuity equation:

The generalized form of the continuity equation is:

$$J\frac{\partial\rho}{\partial t}+\frac{\partial}{\partial\xi}\left(J\rho U\right)+\frac{\partial}{\partial\zeta}\left(J\rho W\right)=0 \tag{2.17}$$

or:

$$J\frac{\partial\rho}{\partial t}+\frac{\partial}{\partial\xi}\left[\rho\left(z_\zeta u-x_\zeta w\right)\right]+\frac{\partial}{\partial\zeta}\left[\rho\left(x_\xi w-z_\xi u\right)\right]=0 \tag{2.18}$$

A2.2 – Integration

The integration and solution method for the transport equations are entirely based on the methodology by Patankar (1980), with some of the suggestions described in Van Doormaal and Raithby (1984).

A2.2.1 – Space integration

Let us, now, focus on the integration of the transport equation for the variable ϕ, taking as reference the control volume shown in Figure A.2.

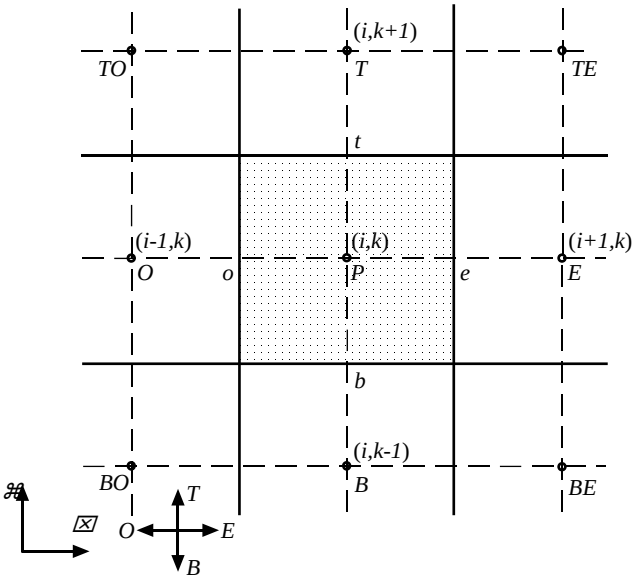


Figure A.2 – Control volume in the computational domain.

The general equation (2.10) may be written as:

$$J\frac{\partial(\rho\phi)}{\partial t}+\frac{\partial}{\partial\xi}\left(F_\xi\phi-\Gamma Jg^{11}\frac{\partial\phi}{\partial\xi}\right)+\frac{\partial}{\partial\zeta}\left(F_\zeta\phi-\Gamma Jg^{11}\frac{\partial\phi}{\partial\zeta}\right)=J\left(S_{cross}+S_\phi\right) \tag{2.19}$$

The integration of the previous equation in its *CV* leads to:

$$\begin{aligned}&J\frac{\partial(\rho\phi)}{\partial t}+\left[F_{\xi e}\phi_e-\left(\Gamma Jg^{11}\right)_e\left(\phi_E-\phi_P\right)\right]-\left[F_{\xi o}\phi_o-\left(\Gamma Jg^{11}\right)_o\left(\phi_P-\phi_O\right)\right]+\\&+\left[F_{\zeta t}\phi_t-\left(\Gamma Jg^{33}\right)_t\left(\phi_T-\phi_P\right)\right]-\left[F_{\zeta b}\phi_b-\left(\Gamma Jg^{33}\right)_b\left(\phi_P-\phi_B\right)\right]=J\left(S_{1_P}+S_{2_P}\phi_P\right)\end{aligned} \tag{2.20}$$

or

$$\begin{aligned}&J\frac{\partial(\rho\phi)}{\partial t}+\left[F_e\phi_e-D_e\left(\phi_E-\phi_P\right)\right]-\left[F_o\phi_o-D_o\left(\phi_P-\phi_O\right)\right]+\\&\left[F_t\phi_t-D_t\left(\phi_T-\phi_P\right)\right]-\left[F_b\phi_b-D_b\left(\phi_P-\phi_B\right)\right]=J\left(S_{1_P}+S_{2_P}\phi_P\right)\end{aligned} \tag{2.21}$$

where the source term has been explicited as a linear combination involving ϕ_P. D represents the diffusive coefficients, respectively:

$$F_e = F_{\xi e} = \left[\rho\left(z_\zeta\, u - x_\zeta w\right)\right]_e \;\; ; \;\; F_o = F_{\xi o} = \left[\rho\left(z_\zeta\, u - x_\zeta w\right)\right]_o$$

$$F_t = F_{\zeta t} = \left[\rho\left(x_\xi\, w - z_\xi u\right)\right]_t \;\; ; \;\; F_b = F_{\zeta b} = \left[\rho\left(x_\xi\, w - z_\xi u\right)\right]_b \tag{2.22}$$

$$D_e = \left(\Gamma J g^{11}\right)_e \; ; \; D_e = \left(\Gamma J g^{11}\right)_o \; ; \; D_t = \left(\Gamma J g^{33}\right)_t \; ; \; D_b = \left(\Gamma J g^{33}\right)_b \tag{2.23}$$

In the previous expressions, the subscripts indicate the location relative to the *CV* centre, in the computational domain, with the uppercase denoting neighbour nodes and the lowercase denoting neighbour faces (cf. Figure A2.2):

E,*e*: - East; *O*,*o*: - West; *T*,*t*: - Top *B*,*b*: - Bottom

For simplicity, in equation (2.21), the subscript ξ and ζ was dropped from the fluxes F (in fact, fluxes across the eastern and western faces are always along the ξ direction and fluxes across the top and bottom faces are always along the ζ direction).

A2.2.2 – Time integration

Time integration is done using, either the First Order Backward Euler scheme:

$$\frac{\partial(\rho\phi)}{\partial t} = \frac{1}{\Delta t}\left(\rho_P\phi_P - \rho_P^0\phi_P^0\right) \tag{2.24}$$

or the Second Order Backward Euler scheme:

$$\frac{\partial(\rho\phi)}{\partial t} = \frac{1}{\Delta t}\left(\frac{3}{2}\rho_P\phi_P - 2\rho_P^0\phi_P^0 + \frac{1}{2}\rho_P^{00}\phi_P^{00}\right) \tag{2.25}$$

The exponent 0 (like in ϕ_P^0) refers to the previous time instant (one time instant back), while the exponent 00 refers to 2 time instants back .

Time integration is performed adopting a fully implicit approach, *i.e.*, it is assumed that the values corresponding to the present time instant prevail during almost the totality of the time interval. In the mathematical counterpart, this is equivalent to take as unit the weighting factor *f* in the following equation:

$$\int_t^{t+\Delta t} \phi\, dt = \left[\, f\phi + (1-f)\phi^0 \,\right]\Delta t \quad ; \quad f = 1 \;\Rightarrow\; \int_t^{t+\Delta t} \phi\, dt = \phi\,\Delta t \tag{2.26}$$

For the solution of the equations, it is necessary to evaluate the values of ϕ in the *CV* faces (*i.e.*, $\phi_e, \phi_o, \phi_t, \phi_b$). These values are computed as function of both ϕ_P and the values in the neighbour nodes $\phi_E, \phi_O, \phi_T, \phi_B$, according to the adopted advection scheme (to be described later on). Equation (2.21) may, then, be written in the following standard form:

$$a_P\phi_P = a_E\phi_E + a_O\phi_O + a_T\phi_T + a_B\phi_B + b \tag{2.27}$$

or, in a more compact manner:

$$a_P\phi_P = \sum_{nb} a_{nb}u_{nb} + b \tag{2.27b}$$

with "*nb*" indicating that the sum is to be performed for all the neighbouring locations.

For the First Order Backward Euler time integration:

$$b = J\left(S_{1_P} + \frac{\rho_P^0\phi_P^0}{\Delta t}\right) \tag{2.28}$$

$$a_P = a_E + a_O + a_T + a_B + J\left(\frac{\rho_P\phi_P}{\Delta t} - S_{2_P}\right) \quad \Leftrightarrow \quad a_P = \sum_{viz} a_{viz} + J\left(\frac{\rho_P\phi_P}{\Delta t} - S_{2_P}\right) \tag{2.29}$$

For the Second Order Backward Euler time integration:

$$b = J\left[S_{1_P} + \frac{1}{\Delta t}\left(2\rho_P^0\phi_P^0 - \frac{1}{2}\rho_P^{00}\phi_P^{00}\right)\right] \tag{2.30}$$

$$a_P = a_E + a_O + a_T + a_B + J\left(\frac{3}{2\Delta t}\rho_P\phi_P - S_{2_P}\right) \quad \Leftrightarrow \quad a_P = \sum_{viz} a_{viz} + J\left(\frac{3}{2\Delta t}\rho_P\phi_P - S_{2_P}\right) \tag{2.31}$$

A2.2.3 – Advection schemes

The problem, now, is how to compute $\phi_e, \phi_o, \phi_t, \phi_b$ as function of $\phi_P, \phi_E, \phi_O, \phi_T, \phi_B$, in order to obtain the coefficients a_E, a_O, a_T, a_B of equation (2.27). Let us take, for example, ϕ_e, which should be a function of ϕ_P and of ϕ_E. A simple arithmetic average is not a physically plausible solution since, due to the presence of a flow, the property ϕ, being advected, tends to assume a value closer to the upwind value. Several advection schemes may be adopted, being the simplest one the *upwind* scheme. According to this scheme, the property ϕ in the *CV* face takes the upwind value, *i.e.*, for example:

$$F_e > 0 \quad \Rightarrow \quad \phi_e = \phi_P \tag{2.32}$$

$$F_e < 0 \quad \Rightarrow \quad \phi_e = \phi_E \tag{A2.10b}$$

Upwind scheme:

$$\begin{aligned} a_E &= D_e + \ -F_e, 0 \quad ; \quad a_O = D_o + \ F_o, 0 \\ a_T &= D_t + \ -F_t, 0 \quad ; \quad a_B = D_b + \ F_b, 0 \end{aligned} \tag{2.33}$$

where the operator takes the largest of its arguments. Other schemes may be considered, even involving non-neighbour nodes of *P*. Besides *Upwind*, the present code adopts the advection schemes described in Patankar (1980), all first order accurate, and the *Quick* scheme, which is third-order accurate.

Hybrid scheme:

$$a_E = \max\left[0.5|F_e|, D_e\right] - 0.5F_e \quad ; \quad a_O = \max\left[0.5|F_o|, D_o\right] + 0.5F_o$$
$$a_T = \max\left[0.5|F_t|, D_t\right] - 0.5F_t \quad ; \quad a_B = \max\left[0.5|F_b|, D_b\right] + 0.5F_b \tag{2.34}$$

Power-Law scheme:

$$a_E = D_e \max\left[0, \left(1 - \frac{0.1|F_e|}{D_e}\right)^5\right] + \max\left[0, -F_e\right] \quad ; \quad a_O = D_o \max\left[0, \left(1 - \frac{0.1|F_o|}{D_o}\right)^5\right] + \max\left[0, F_o\right]$$
$$a_T = D_t \max\left[0, \left(1 - \frac{0.1|F_t|}{D_t}\right)^5\right] + \max\left[0, -F_t\right] \quad ; \quad a_B = D_b \max\left[0, \left(1 - \frac{0.1|F_b|}{D_b}\right)^5\right] + \max\left[0, F_b\right] \tag{2.35}$$

Quick scheme:

The Quick scheme is third-order accurate. **EasyCFD** implements the deferred correction scheme version of Hayase (1992), which combines the first order upwind scheme (cf. Equations (2.33)) with a third order correction, b_{quick}, added to the source term. The corresponding mathematical formulation is as follows:

$$\begin{aligned} b_{quick} = & \\ & \frac{1}{8}\max\left[F_e, 0\right]\left(-\phi_O - 2\phi_P + 3\phi_E\right) - \frac{1}{8}\max\left[-F_e, 0\right]\left(3\phi_P - 2\phi_E - 3\phi_{EE}\right) + \\ & \frac{1}{8}\max\left[F_o, 0\right]\left(3\phi_O - 2\phi_P - \phi_E\right) - \frac{1}{8}\max\left[-F_o, 0\right]\left(-\phi_{OO} - 2\phi_O + 3\phi_P\right) + \\ & \frac{1}{8}\max\left[F_t, 0\right]\left(-\phi_B - 2\phi_P + 3\phi_T\right) - \frac{1}{8}\max\left[-F_t, 0\right]\left(3\phi_P - 2\phi_T - 3\phi_{TT}\right) + \\ & \frac{1}{8}\max\left[F_b, 0\right]\left(3\phi_B - 2\phi_P - \phi_T\right) - \frac{1}{8}\max\left[-F_b, 0\right]\left(-\phi_{BB} - 2\phi_B + 3\phi_P\right) + \end{aligned} \tag{2.36}$$

The Quick scheme has better accuracy than the previous schemes, resulting in low values of false diffusion. To illustrate, consider the case of the transport for scalar ϕ subject to a steep variation in the boundary conditions, represented in Figure A3. The flow is not aligned with the mesh and physical diffusion is zero. The solution for the scalar plotted along the main diagonal perpendicular to the flow direction of represented in Figure A.3. It may be observed that, when compared to the exact solution, the Quick scheme presents better accuracy, although at the expenses of minor overshoots and undershoots, corresponding to unbounded results. This may lead to unrealistic solutions. The same happens with the second order upwind scheme, which formulation is presented below.

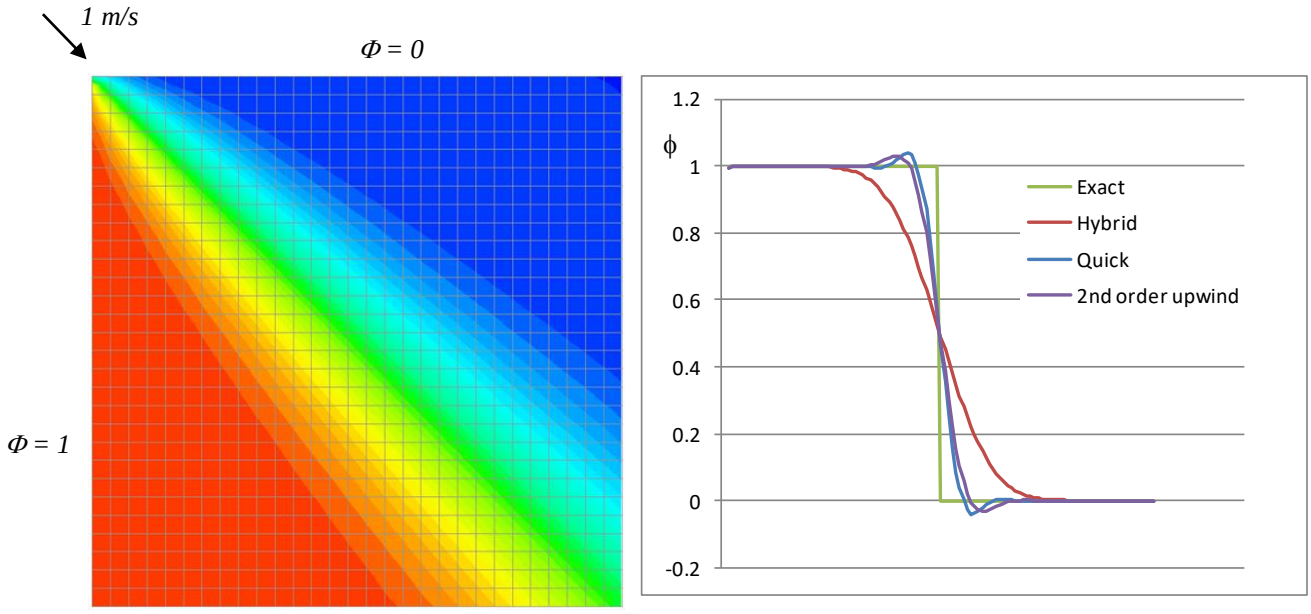
Total variation diminishing (TVD) schemes were developed to provide second-order accurate solutions that are free or nearly-free from oscillations (cf. Vertseeg and Malalasekera, 2007, for further information on the subject). For their formulation, the property at face f is computed assembling a diffusive upwind part and an anti-diffusive contribution, written in the following form (cf. Figure A.4)

$$\phi_f = \phi_C + 0.5\psi(r)(\phi_D - \phi_C) \tag{2.37}$$

where $\psi(r)$ if a flux limiter and r quantifies the variation of ϕ.

$$r = \frac{(\phi_C - \phi_U)}{(\phi_D - \phi_C)} \tag{2.38}$$

The value of the upstream node can be either taken at the upstream control volume, or computed by interpolation at a location obtained by projecting the line *D*-*C* (cf. Figure A.4). The latest option is computationally much more expensive and is optional, being carried out only near convergence. Different formulations for the function $\psi(r)$ correspond to different schemes. The second-order upwind scheme is obtained for $\psi(r) = r$, while the previously presented Quick scheme is obtained for $\psi(r) = (3+r)/4$. As previously referred, the schemes are not TVD. The representation of the function $\psi(r)$ allows the identification of the TVD region, which is the region including the shaded area and below, as depicted in Figure A.5. Second-order accurate schemes should, also, pass through the location (1,1) in the Sweby diagram. Some of the schemes presented in Vertseeg and Malalasekera, 2007 are available in EasyCFD, as listed in Table A1. The results obtained with these schemes are displayed in Figure A.6. It is noticeable that, when compared with the hybrid scheme, the TVD schemes show much less false diffusion, without the problem of undershoots and overshoots of non-TVD schemes like Quick and second order upwind.



(a) (b)

Figure A.3 – Domain and boundary conditions for false diffusion problem. (a) Solution obtained with the hybrid advection scheme; (b) comparison between the hybrid, the Quick and the second order upwind schemes.

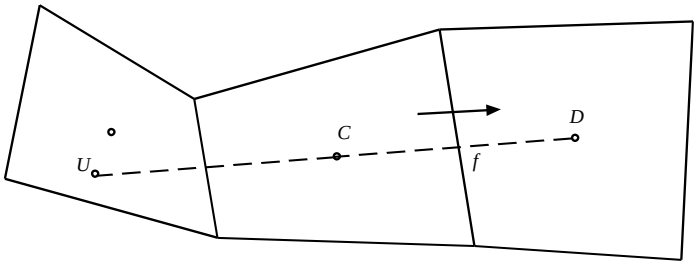


Figure A.4 – Location of control volumes for computation of property at face f.

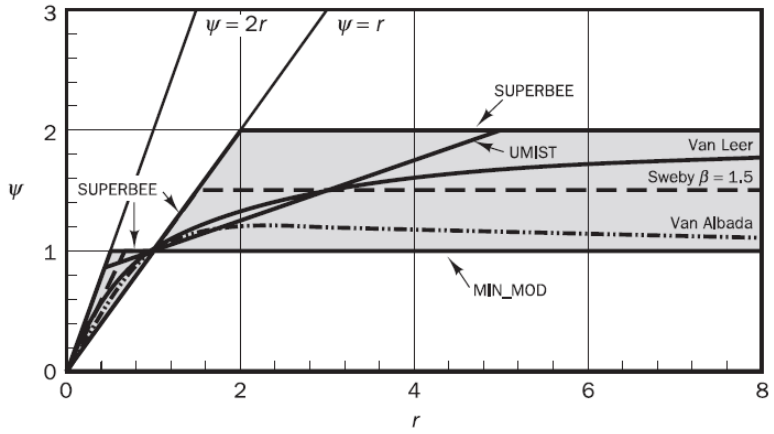
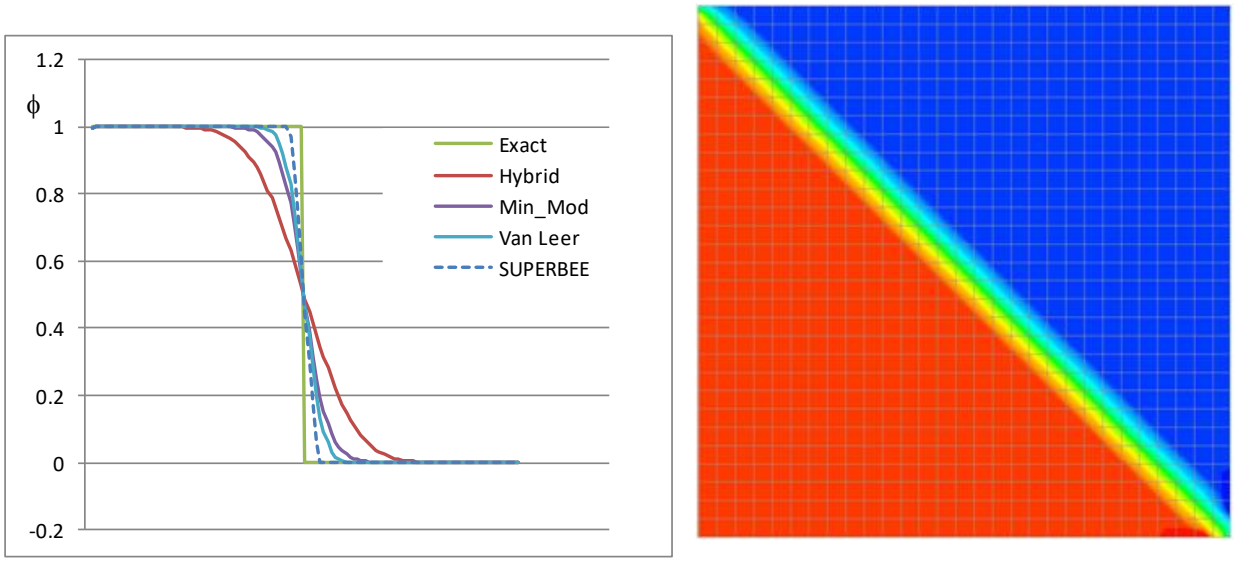


Figure A.5 – The Sweby diagram (from Vertseeg and Malalasekera, 2007)

Table A1 – TVD advection schemes implemented in EasyCFD

Scheme	$\psi(r)$
Min-Mod	$\psi(r) = max\left[0, min(1, r)\right]$
Van Leer	$\psi(r) = \frac{r + \lvert r \rvert}{1 + r}$
SUPERBEE	$\psi(r) = max\left[0, min(2r, 1), min(r, 2)\right]$



(a) (b)

Figure A.6 – (a) Comparison between the different schemes Solution obtained with the hybrid advection scheme; (b) comparison between the hybrid and the Quick scheme.

The values for density and diffusion coefficients at the *CV* faces, necessary for the evaluation of fluxes in equation (2.20), are obtained through linear interpolation. The weighting factor is the inverse of distance. Taking, as example, the calculation of density for evaluating the advective flux at the *East* boundary of the control volume depicted in Figure A.7, we get:

$$\rho_e = f_e \rho_P + (1 - f_e) \rho_E \tag{2.39}$$

with

$$f_e = \frac{\Delta L_{\xi e+}}{\Delta L_{\xi e}} \tag{2.40}$$

The previous equation leads to the arithmetic average when the control volume face is placed exactly midway between the neighbour nodes (*i.e.*, when $\Delta L_{\xi e+} = \Delta L_{\xi e-}$).

For the calculation of diffusive fluxes, it is more correct to employ the following formula for the calculation of the diffusion coefficient (Patankar 1980):

$$\Gamma_e = \frac{1}{\left(\frac{1 - f_e}{\Gamma_P} + \frac{f_e}{\Gamma_E} \right)} \tag{2.41}$$

In this case, if $\Delta L_{\xi e+} = \Delta L_{\xi e-}$, the harmonic average is obtained. In the present code, both types of interpolation for the computation of the diffusion coefficients are available.

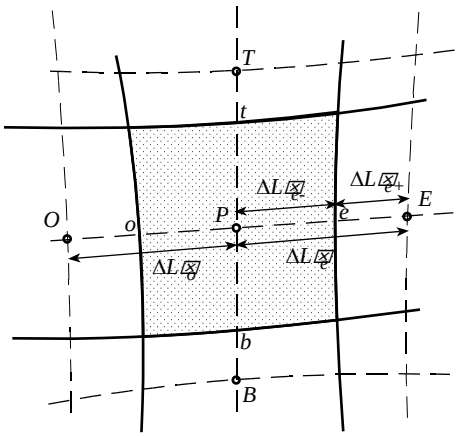


Figure A.7 – Control volume in the physical domain.

A2.3 – Pressure-Velocity coupling

The present code adopts the *SIMPLEC* algorithm (Semi-Implicit Method for Pressure-Linked Equations - Consistent), proposed by Van Doormaal and Raithby (1984), which is based on the original formulation *SIMPLE* by Patankar (1980). Due to the non-staggered mesh arrangement (collocated mesh), The Rie-Chow interpolation procedure (Rhie and Chow, 1983), with the modifications proposed by Majumdar (1988), also described by Shen et. al. (2003), is implemented in the present code.

Let us consider the u momentum conservation equation (2.13). After its integration, the evaluation of this equation leads to:

$$a_P u_P = \sum_{nb} a_{nb} u_{nb} - z_\zeta \frac{\partial p}{\partial \xi} + z_\xi \frac{\partial p}{\partial \zeta} + S_{crossu} \tag{2.42}$$

During the iterative process, velocities u^* are computed from the available velocity field u^m and pressure field p^* obtained at the previous step, as follows:

$$a_P \left(1 + \frac{1}{E}\right) u_P^* = \sum_{nb} a_{nb} u_{nb}^* + \frac{a_P}{E} u_P^m - z_\zeta \frac{\partial p^*}{\partial \xi} + z_\xi \frac{\partial p^*}{\partial \zeta} + S_{crossu} \tag{2.43}$$

where E is the under-relaxation factor proposed by Van Doormaal and Raithby (1984).

A2.4 – Velocity Correction

During the iterative process, unless convergence is reached, the velocity field u^* obtained from the solution of the momentum equations does not satisfy the continuity requirement, unless the correct pressure field p was employed (instead of an incorrect pressure field p^*). This means that, if the correct pressure field was employed, a mass-conservative velocity field would be obtained:

$$a_P \left(1 + \frac{1}{E}\right) u_P = \sum_{nb} a_{nb} u_{nb} + \frac{a_P}{E} u_P^m - z_\zeta \frac{\partial p}{\partial \xi} + z_\xi \frac{\partial p}{\partial \zeta} + b \tag{2.44}$$

Thus, the pressure field $p*$ should be corrected by a certain amount p':

$$p = p* + p' \tag{2.45}$$

and, similarly, for the velocity field:

$$u_P = u_P^* + u_P^{'} \tag{2.46}$$

Subtracting equation (2.44) from equation (2.43) and taking into account equations (2.45) and (2.46), one obtains:

$$a_P\left(1+\frac{1}{E}\right)u_P^{'} = \sum_{nb} a_{nb}u_{nb}^{'} - z_\zeta \frac{\partial p^{'}}{\partial \xi} + z_\xi \frac{\partial p^{'}}{\partial \zeta} \tag{2.47}$$

The combination of equations (2.46) and (2.47) with the continuity equation would allow us to get an equation for the pressure correction p'. In this case, however, each p' value would remain connected to the p' values in the entire domain, due to the term $\sum_{nb} a_{nb}u_{nb}^{'}$ in the previous equation. As a workaround, the *SIMPLE* algorithm proposes to drop this term in equation (2.47):

$$a_P\left(1+\frac{1}{E}\right)u_P^{'} = \sum_{nb} a_{nb}u_{nb}^{'} - z_\zeta \frac{\partial p^{'}}{\partial \xi} + z_\xi \frac{\partial p^{'}}{\partial \zeta} \tag{2.48}$$

A better proposal, which constitutes the keystone of the *SIMPLEC* algorithm, consists on the subtraction of the term $\sum_{nb} a_{nb}u_P^{'}$ to both sides of the equation:

$$\left[a_P\left(1+\frac{1}{E}\right) - \sum_{nb} a_{nb}\right]u_P^{'} = \cancel{\sum_{nb} a_{nb}\left(u_{nb}^{'} - u_P^{'}\right)} - z_\zeta \frac{\partial p^{'}}{\partial \xi} + z_\xi \frac{\partial p^{'}}{\partial \zeta} \tag{2.49}$$

or, for simplicity:

$$\tilde{a}_P u_P^{'} = -z_\zeta \frac{\partial p^{'}}{\partial \xi} + z_\xi \frac{\partial p^{'}}{\partial \zeta} \tag{2.50}$$

where:

$$\tilde{a}_P = a_P\left(1+\frac{1}{E}\right) - \sum_{nb} a_{nb} \tag{2.51}$$

being, in this case, dropped the term indicated in equation (2.49). This is a more **C**onsistent approximation that the SIMPLE proposal, as $u_P^{'}$ will be close to its neighbours. The equations for the velocity correction are obtained through the pressure correction field, recurring to the previous equation:

$$u_P^{'} = -\frac{z_\zeta}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi} + \frac{z_\xi}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta} \tag{2.52}$$

leading to:

$$u_P = u_P^* - \frac{z_\zeta}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi} + \frac{z_\xi}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta} \tag{2.53}$$

and, for the *w* component:

$$w_P^{'} = -\frac{x_\xi}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta} + \frac{x_\zeta}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi} \tag{2.54}$$

leading to

$$w_P = w_P^* + \frac{x_\zeta}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi} - \frac{x_\xi}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta} \tag{2.55}$$

A2.5 – The Pressure Correction Equation

As previously stated, the objective of the pressure correction is to produce a pressure field such that the solution of the momentum equations is a mass-conservative velocity field. Consequently, the equations for solving the *p'* field must be obtained from the continuity equation. The discretized form of this equation is obtained directly from the integration of equation (1.5):

$$J\frac{\rho_P - \rho_P^0}{\Delta t} + \left[\rho\left(z_\zeta u - x_\zeta w\right)\right]_e - \left[\rho\left(z_\zeta u - x_\zeta w\right)\right]_o + \left[\rho\left(x_\xi w - z_\xi u\right)\right]_t - \left[\rho\left(x_\xi w - z_\xi u\right)\right]_b = 0 \tag{2.56}$$

As one may see, velocities are, now, needed at the control volume faces. Taking equation (2.53), at the "*e* " and "*o*" faces and (2.55) at the "*t*" and "*b*" faces, substituting into equation (2.56) and rearranging the terms leads to:

$$\begin{aligned} &J\frac{\rho_P - \rho_P^0}{\Delta t} + F_e^* - F_o^* + F_t^* - F_b^* - \left(\frac{\rho g_{33}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi}\right)_e + \left(\frac{\rho g_{33}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi}\right)_o - \left(\frac{\rho g_{11}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta}\right)_t + \left(\frac{\rho g_{11}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta}\right)_b + \\ &\left(\frac{\rho g_{13}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta}\right)_e - \left(\frac{\rho g_{13}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \zeta}\right)_o + \left(\frac{\rho g_{13}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi}\right)_t - \left(\frac{\rho g_{13}}{\tilde{a}_P}\frac{\partial p^{'}}{\partial \xi}\right)_b = 0 \end{aligned} \tag{2.57}$$

The terms g_{ij} are the covariant metric relations, defined as:

$$g_{11} = x_\xi{}^2 + z_\xi{}^2 = g^{33}J^2 \tag{2.58}$$

$$g_{33} = x_\zeta{}^2 + z_\zeta{}^2 = g^{11}J^2 \tag{2.59}$$

$$g_{13} = x_\xi x_\zeta + z_\xi z_\zeta = -g^{13}J^2 \tag{2.60}$$

The derivatives are evaluated as (cf. Figure A2.2):

$$\left(\frac{\partial p'}{\partial \xi}\right)_e = p'_E - p'_P \;\; ; \;\; \left(\frac{\partial p'}{\partial \xi}\right)_o = p'_P - p'_O \;\; ; \;\; \left(\frac{\partial p'}{\partial \zeta}\right)_t = p'_T - p'_P \;\; ; \;\; \left(\frac{\partial p'}{\partial \zeta}\right)_b = p'_P - p'_B \tag{2.61}$$

and

$$\left(\frac{\partial p'}{\partial \xi}\right)_t = 0.25\left(p'_E + p'_{TE} - p'_O - p'_{TO}\right) \;\; ; \;\; \left(\frac{\partial p'}{\partial \xi}\right)_b = 0.25\left(p'_E + p'_{BE} - p'_O - p'_{BO}\right)$$

$$\left(\frac{\partial p'}{\partial \zeta}\right)_e = 0.25\left(p'_T + p'_{TE} - p'_B - p'_{BE}\right) \;\; ; \;\; \left(\frac{\partial p'}{\partial \zeta}\right)_o = 0.25\left(p'_T + p'_{TO} - p'_B - p'_{BO}\right) \tag{2.62}$$

Introducing the discretisation expressed by equations (2.62) and (2.61) into equation (2.57), allows us to obtain the pressure correction equation:

$$a_P p'_P = a_E p'_E + a_O p'_O + a_T p'_T + a_B p'_B + b \tag{2.63}$$

where

$$a_E = \left(\frac{g_{33}}{\langle \tilde{a}_P / \rho \rangle}\right)_e \;\; ; \;\; a_O = \left(\frac{g_{33}}{\langle \tilde{a}_P / \rho \rangle}\right)_o \;\; ; \;\; a_T = \left(\frac{g_{11}}{\langle \tilde{a}_P / \rho \rangle}\right)_t \;\; ; \;\; a_B = \left(\frac{g_{11}}{\langle \tilde{a}_P / \rho \rangle}\right)_b \tag{2.64}$$

$$a_P = a_E + a_O + a_T + a_B \tag{2.65}$$

$$\begin{aligned} b = {} & J\frac{\rho_P^0 - \rho_P}{\Delta t} - F_e^* + F_o^* - F_t^* + F_b^* - \\ & \left(\frac{g_{13}}{\langle \tilde{a}_P / \rho \rangle}\frac{\partial p'}{\partial \zeta}\right)_e + \left(\frac{g_{13}}{\langle \tilde{a}_P / \rho \rangle}\frac{\partial p'}{\partial \zeta}\right)_o - \left(\frac{g_{13}}{\langle \tilde{a}_P / \rho \rangle}\frac{\partial p'}{\partial \xi}\right)_t + \left(\frac{g_{13}}{\langle \tilde{a}_P / \rho \rangle}\frac{\partial p'}{\partial \xi}\right)_b \end{aligned} \tag{2.66}$$

The $\langle \; \rangle$ symbol denotes a linear interpolation from the control volume centre (where the momentum equations are defined) to the control volume faces (where the fluxes for the continuity equations are needed).

The pressure correction field *p'* obtained from the solution of equation (2.63) is employed for correcting the pressure through equation (2.45) and velocity through equations (2.53) and (2.55). Note that, since these equations are defined at the control volume centre, *p'* derivatives are evaluated as:

$$\left(\frac{\partial p'}{\partial \xi}\right)_P = 0.5\left(p'_E - p'_O\right) \;\; ; \;\; \left(\frac{\partial p'}{\partial \zeta}\right)_P = 0.5\left(p'_T - p'_B\right) \tag{2.67}$$

One may note that pressure values at the control volume centre are not included in the evaluation of these derivatives (the same applies for the pressure derivatives in the momentum equations (2.15) and (2.16)). This may lead to the well know checkerboard pattern for the pressure field. To avoid this

effect, the Chie-Row interpolation method proposes that the mass fluxes, to be evaluated at the control volume interfaces for all the transport equations (F_e, F_o, F_t and F_b, in (2.21)), be corrected using the pressure correction p' field (instead of being computed from the corrected control volume centre velocities). The correction equations for fluxes are obtained for the correction equations for velocities:

$$F_{e,o} = F_{e,o}^{\ *} - \left[\left\langle \frac{\rho}{\tilde{a}_P} \right\rangle \left(g_{33} \frac{\partial p'}{\partial \xi} - g_{13} \frac{\partial p'}{\partial \zeta} \right) \right]_{e,o} \tag{2.68}$$

$$F_{t,b} = F_{t,b}^{\ *} - \left[\left\langle \frac{\rho}{\tilde{a}_P} \right\rangle \left(g_{11} \frac{\partial p'}{\partial \zeta} - g_{13} \frac{\partial p'}{\partial \xi} \right) \right]_{t,b} \tag{2.69}$$

The "starred" fluxes F^* at the control volume interfaces are obtained by interpolating the momentum equation. The keystone of the method is that the pressure gradient is not interpolated, but, instead, it is obtained directly from the pressure at contiguous control volume centres. Taking equation (2.43), evaluated in terms of fluxes, one may write:

$$F_{e,o}^{\ *} = (1+E) F_{e,o}^{m} + \hat{F}_{e,o} + \left\langle \frac{\rho}{\tilde{a}_P} \right\rangle_{e,o} \left(g_{33} \frac{\partial p}{\partial \xi} - g_{13} \frac{\partial p}{\partial \zeta} \right)_{e,o} \tag{2.70}$$

$$F_{t,b}^{\ *} = (1+E) F_{t,b}^{m} + \hat{F}_{t,b} + \left\langle \frac{\rho}{\tilde{a}_P} \right\rangle_{t,b} \left(g_{11} \frac{\partial p}{\partial \zeta} - g_{13} \frac{\partial p}{\partial \xi} \right)_{t,b} \tag{2.71}$$

The terms $\hat{F}_{e,o}$ and $\hat{F}_{t,b}$ represent the fluxes:

$$\hat{F}_{e,o} = z_\zeta \langle \rho \hat{u} \rangle_{e,o} - x_\zeta \langle \rho \hat{w} \rangle_{e,o} \tag{2.72}$$

$$\hat{F}_{t,b} = x_\xi \langle \rho \hat{w} \rangle_{t,b} - z_\zeta \langle \rho \hat{u} \rangle_{t,b} \tag{2.73}$$

where the revised velocities $\hat{u}$ and $\hat{w}$ are obtained from the momentum equations as follows:

$$\tilde{a}_P \hat{u}_P = \sum_{nb} a_{nb} \hat{u}_{nb} + \cancel{\frac{(1-\alpha) a_P}{\alpha} u_P^m} - \cancel{z_\zeta \frac{\partial p^*}{\partial \xi}} + \cancel{z_\xi \frac{\partial p^*}{\partial \zeta}} + b \ \Rightarrow \hat{u}_P = \frac{\sum_{nb} a_{nb} \hat{u}_{nb} + b}{\tilde{a}_P} \tag{2.74}$$

$$\tilde{a}_P \hat{w}_P = \sum_{nb} a_{nb} \hat{w}_{nb} + \cancel{\frac{(1-\alpha) a_P}{\alpha} \hat{w}_P^m} + \cancel{x_\zeta \frac{\partial p^*}{\partial \xi}} - \cancel{x_\xi \frac{\partial p^*}{\partial \zeta}} + b \ \Rightarrow \hat{w}_P = \frac{\sum_{nb} a_{nb} \hat{w}_{nb} + b}{\tilde{a}_P} \tag{2.75}$$

Note that the source term b includes all the contributions (*e.g.* transient term, cross derivatives and buoyancy (for the w velocity component)), except the under-relaxation and pressure gradient.

A3 – Boundary Conditions

A3.1 – The *p'* Equation

When the local flow velocity is known, such as in the case of solid boundaries or inlets and outlets with imposed velocity, mass flow rate or volume flow rate, there should be no velocity correction with the *p'* values. Consequently, the corresponding coefficient in the *p'* equation should be zero, which is equivalent to impose a null gradient for this variable.

A3.2 – Pressure Values

The pressure field is initialized with null values. The pressure field in the final solution is established by the pressure corrections that took place during the iterative process. In fact, the velocity field is conditioned by pressure gradients and not by pressure itself. The pressure at a certain location has not, from this standpoint, a direct physical interpretation. Nevertheless, the overall pressure level can be imposed by properly specifying a reference pressure at a certain location (cf. B1.5.5).The velocity field at the outlet is computed based on a global mass balance. There is no need to correct the outlet velocities and, consequently, the corresponding coefficients in the pressure correction equation are zero. The pressure field excludes the hydrostatic gradient due to ρ_{ref} (cf. equation (1.51)). The total pressure p_{tot}, i.e., pressure that includes the hydrostatic term, is obtained as:

$$p_{tot} = p - \rho_{ref}\, g\, z \tag{3.1}$$

A3.3 – Momentum

A3.3.1 – Outlet boundaries and symmetry boundaries

A parabolic condition is assigned to the outlet boundaries, by imposing a null coefficient in the transport equations. A similar condition is applied at symmetry boundaries

A3.3.2 – Walls

Solid boundaries (walls) have a non-slip condition, *i.e.*, the fluid in contact with the wall acquires its velocity. Furthermore, a wall is always impervious to the flow and, consequently, the velocity component perpendicular to it is zero. Thus, the momentum flux into the wall, for the *CV* adjacent to it, is entirely composed by the diffusive part and is given, per unit area, by the expressions (1.29) and (1.30). Let us now take, for example, a *CV* adjacent to a wall, as depicted in Figure A.8. In terms of the discretized equation and using equations (2.21) and (2.27), it is not difficult to verify that the flux into the wall is given by $a_B\left(\phi_P - \phi_B\right)$. In this example, since there is no control volume for the velocity below the wall (there is no ϕ_B), the more efficient manner to impose the flux is through the source term (an alternative would be to use fictitious nodes). Following this approach, the corresponding coefficient is set to zero, *i.e.*:

$$a_B = 0 \tag{3.2}$$

and the source term $S_1 + S_2\phi_P$ is given by:

$$y^+ \le 11.63 \; or \; laminar \; flow \quad \Rightarrow \quad S_1 = \Delta L_1 \frac{\mu}{\Delta L_2} u_0 \;\; ; \;\; S_2 = -\Delta L_1 \frac{\mu}{\Delta L_2} \tag{3.3}$$

$$y^+ > 11.63 \quad \Rightarrow \quad S_1 = \Delta L_1 \frac{\rho C_\mu^{0.25} \chi \sqrt{k}}{ln\left(E^* y^+\right)} u_0 \;\; ; \;\; S_2 = -\Delta L_1 \frac{\rho C_\mu^{0.25} \chi \sqrt{k}}{ln\left(E^* y^+\right)} \tag{3.4}$$

where u_0 is the wall velocity.

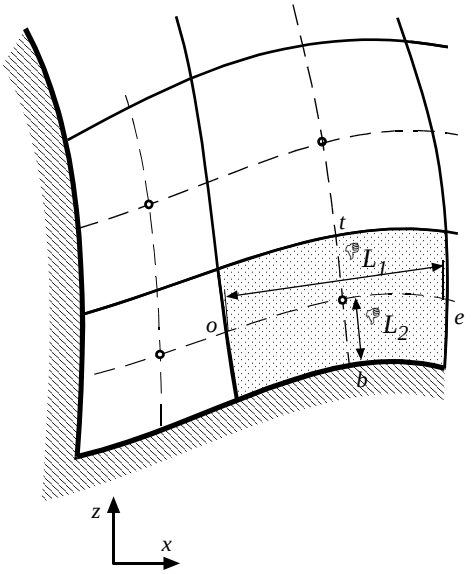


Figure A.8 – Control volume in the computational domain, at the neighbourhood of a wall.

Note 1: the previous equations illustrate contributions to the source term from the imposition of boundary conditions. Further to these, the source term may include other contributions such as, for example, those from subrelaxation.

Note 2: The shown example is for the *k*-ε model. Treatment for SST *k*-ε model follows a similar approach, adapted for the corresponding equations.

A3.4 – Energy

A3.4.1 – Outlet boundaries and symmetry boundaries

In these boundaries, a null gradient condition is imposed, by setting to zero the corresponding coefficients in the transport equations.

A3.4.2 – Walls

Imposed Flux

Is this case, the heat flux that enters the domain is known. The boundary condition is imposed through the source term. Taking as example the control volume depicted in Figure A.8, we have:

$$a_B = 0 \tag{3.5}$$

$$S_1 = \Delta L_1 \, H_b \tag{3.6}$$

where H_b is the heat flux [W/m^2] (positive if entering the domain).

Imposed Temperature

Generically, the imposition of a certain value ϕ_{imp} in a *CV* may be done with the source term, through the following formulation:

$$S_1 = 10^{20} \phi_{imp} \quad ; \quad S_2 = -10^{20} \tag{3.7}$$

If we consider the general equation:

$$\left(\sum_{nb} a_{nb} - S_2 \right) \phi_P = \sum_{nb} a_{nb} \phi_{nb} + S_1 \tag{3.8}$$

we can easily conclude that it leads to the desired value for ϕ_P:

$$\phi_P = \frac{\sum_{nb} a_{nb} \phi_{nb} + 10^{20} \phi_{imp}}{\sum_{nb} a_{nb} + 10^{20}} \approx \phi_{imp} \tag{3.9}$$

Denoting by T_0 the imposed wall temperature and referring to Figure A.8, the wall functions (1.35) and (1.36) lead to:

$$a_B = 0 \tag{3.10}$$

$$y^+ \le 12 \quad \Rightarrow \quad S_1 = \Delta L_1 \frac{\mu c_p}{Pr \, \Delta L_2} T_0 \quad ; \quad S_2 = \Delta L_1 \frac{\mu c_p}{Pr \, \Delta L_2} \tag{3.11}$$

$$y^+ > 12 \quad \Rightarrow \quad S_1 = \Delta L_1 \frac{\rho c_p C_\mu^{0.25} \sqrt{k}}{Pr_t \left[\frac{1}{\chi} ln\left(E^* \, y^+ \right) + P^* \right]} T_0 \quad ; \quad S_2 = \Delta L_1 \frac{\rho c_p C_\mu^{0.25} \sqrt{k}}{Pr_t \left[\frac{1}{\chi} ln\left(E^* \, y^+ \right) + P^* \right]} \tag{3.12}$$

A3.5 – Turbulence Kinetic Energy and Dissipation Rate (k-ε model)

Note: Treatment for SST *k-ε* model follows an approach similar to the one described next, adapted for the corresponding equations.

A3.5.1 – Outlet boundaries and symmetry boundaries

In these boundaries, a null gradient condition is imposed, by setting to zero the corresponding coefficients in the transport equation.

A3.5.2 – Walls

Turbulence kinetic energy

As stated in section A1.7.3, the turbulence kinetic energy flux is set to zero in the direction perpendicular to the wall. Thus, referring to the *CV* depicted in Figure A.8:

$$a_B = 0 \tag{3.13}$$

The source term for *k* also includes the terms P_1, $\rho\varepsilon$ and G_T (cf. equation (1.23)) and, in the *CV* neighbour to the wall, equations (1.31) to (1.33) are employed. The implementation in the source term is made as follows:

$$\text{If } G_T < 0 \;\Rightarrow\; S_1 = J\,P_1 \;\;;\;\; S_2 = J\,\frac{\left(G_T - \rho\varepsilon\right)}{k} \tag{3.14}$$

$$\text{If } G_T > 0 \;\Rightarrow\; S_1 = J\left(P_1 + G_T\right) \;\;;\;\; S_2 = -J\,\frac{\rho\varepsilon}{k} \tag{3.15}$$

Negative terms are no included in S_1, in order to prevent *k* from becoming negative.

Dissipation rate of turbulence kinetic energy

In the *CV* adjacent to the wall, the dissipation rate is evaluated by the equation (1.34), implemented in the source term:

$$S_1 = 10^{20}\,\varepsilon_0 \;\;;\;\; S_2 = -10^{20} \tag{3.16}$$

For the remaining control volumes (not adjacent to the wall), the different source terms in equation (1.24) are included in S_1 or in S_2 depending on whether they are positive or negative, respectively (for more details on the linearization of the source term, please cf. Patankar, 1980)

A3.6 – Other Scalars

Other scalars include passive scalars and the concentration of the secondary fluid in multi-component fluid flow.

A3.6.1 – Outlet boundaries and symmetry boundaries

In these boundaries, a null gradient condition is imposed, by setting to zero the corresponding coefficients in the transport equations.

A3.6.2 – Walls

Other scalars are treated similarly as for energy, with the Prandtl number *Pr* replaced by the Schmidt number, *Sc*.

A4 – Solution of the equations

The solution of the equations previously described (equations for *u*, *w*, *p'*, *T*, *k*, and ε) is carried out with an iterative procedure based on the *TDMA* (*TriDiagonal Matrix Algorithm*), which is described in detail in Patankar (1980). Basically, instead of sweeping the mesh node by node, this algorithm solves line by line. In the present code, alternated line sweeps are made for the two directions. For accelerating the converge rate, two relaxation factors (described next) are applied.

A4.1 – Relaxation of the Equations Solution

The solution of the equation is sub- or overrelaxed in the following manner:

$$\phi = f\phi_{computed} + (1-f)\phi_{previous} \tag{4.1}$$

Values of *f* lower than unit lead to subrelaxation, while values greater than unit overrelax the solution process. In the present code, the value $f = 1.1$ is employed for the transport equations, while, for the pressure correction equation, the Point Successive Over Relaxation method (PSOR; cf. Ehrlich (1979) is employed.

A4.2 – Criterion for Stopping the Solution of the *p´ Equations*

The transport equations for *u*, *w*, *T*, *k* and ε are easy to solve and do not need special care. For these equations, the user simply specifies a fixed number of cycles (iterations) for their solution. The *p'* equations, on the other hand, may be quite difficult to solve and require a number of cycles that cannot be predicted in advance. Furthermore, it should be ensured that the *p'* field is sufficiently converged before correcting the velocities, as the mass conservation for the velocity field depends on it. Van Doormaal and Raithby (1985) propose a stopping criterion which is based on the Euclidian norm of the residuals. This norm is defined as follows, for iteration *n*:

$$\|\gamma_n\| = \sqrt{\sum_{all}\left(\sum_{nb} a_{nb} p'_{nb} + b - a_P p'_P\right)^2} \tag{4.2}$$

where the summation of the residual squares is performed for the entire domain (*all*). The criterion establishes that the *p'* equations are sufficiently converged if the following condition is satisfied:

$$\gamma_r = \frac{\|\gamma_n\|}{\|\gamma_1\|} < \gamma_{r\,max} \tag{4.3}$$

where γ_r is the normalized Euclidian residual and $\gamma_{r\,max}$ is its maximum allowed value ($\gamma_{r\,max} = 0.1$, by default, in the present code). This criterion works rather well, though in some occasions it underpredicts the necessary number of cycles, especially at the beginning of the *SIMPLEC* iterative procedure. It is recommended the number of cycles to be no less than 20.

A4.3 – Convergence Criteria

The whole flow field calculation is considered to be converged when all the normalized residuals are lower that a predefined value R_{conv}.

$$R_u, R_w, R_m, R_T, R_k, R_e \; < R_{conv} \tag{4.4}$$

A4.3.1 – Normalized residual for the mass conservation equation

After solving the momentum equations (cf. *SIMPLEC* sequence in section A4.4), the total normalized residual of equation (1.29) is computed as follows:

$$R_m = \frac{\sum_{all} \left| J \frac{\rho_P - \rho_P^0}{\Delta t} - \left[\rho \left(z_\zeta u - x_\zeta w \right) \right]_e + \left[\rho \left(z_\zeta u - x_\zeta w \right) \right]_o - \left[\rho \left(x_\xi w - z_\xi u \right) \right]_t + \left[\rho \left(x_\xi w - z_\xi u \right) \right]_b \right|}{\sum_{all} F_{in}} \tag{4.5}$$

The summation of the absolute value of the mass residuals is thus normalized with the summation of all mass fluxes F_{in} entering the control volumes.

A4.3.2 – Normalized residual for the transport equations

The total normalized residual for the transport equations (cf. equation (2.27)) of ϕ ($\phi = u, w, T, k, \varepsilon$) is determined as follows:

$$R_\phi = \sum_{all} \left(\frac{\left| a_P \phi_P - \sum_{nb} a_{nb} u_{nb} + b \right| J}{a_P \left(\phi_{max} - \phi_{min} \right)} \right) / \sum_{all} J \tag{4.6}$$

where $\left(\phi_{max} - \phi_{min} \right)$ quantifies the amplitude of the variable ϕ in the calculation domain.

A4.4 – The *SIMPLEC* Algorithm

The main steps of the *SIMPLEC* algorithm are described next. Note that,

1 – Initialisation of variables.

2 – Calculation of coefficients and source terms for the momentum equations. Calculation of residuals.

3 – Solution of the momentum equations.

4 – Calculation of coefficients and source terms for the *p'* equations. Calculation of residuals.

5 – Solution of the *p'* equations.

6 – Correction of the velocity field.

7 – Correction of the pressure field.

8 – Calculation of coefficients and source terms for the energy (T), secondary fluid concentration (ϕ_{c2}) and turbulence (k and ε) equations. Calculation of residuals.

9 – Solution of energy, secondary fluid concentration and turbulence equations.

10 – Checking of convergence criteria and return to point *2* if convergence is not achieved.

11 –Calculation of coefficients and source terms for passive scalar transport

12 –Solution of passive scalars equations.

A5 – Mesh Generation

The quality of the flow solution is greatly dependent on the mesh characteristics. The major requirements for a good mesh are:

- Higher resolution (i.e., higher nodes concentration) in regions where the flow characteristics are expected present higher gradients: these regions are, typically, near solid walls (the flow velocity at the wall is equal to the wall velocity and, thus, a boundary layer is created), and, especially, near solid boundaries with curvature and where recirculation regions are expected to be present.
- Smoothness. The mesh spacing should change gradually, to reduce discretisation errors.
- Low degree of skewness. The higher the skewness, the larger the discretisation errors. Ideally, the angle of intersection of mesh lines should be close to *90º*, though in most cases this is not possible.

Two types of quadrilateral meshes are available within EasyCFD: structured meshes and unstructured meshes.

Structured meshes allow more control on the final result, but are more complex to setup and may be difficult to proper setup in complex geometries.

Unstructured meshes are easier to setup, but the user has less control. For instance, for a symmetric geometry, the final mesh will be, more likely, non-symmetric mesh.

Both types of mesh compute the position of inner nodes starting from the 1D domain boundary nodes, which are generated as described next.

A5.1 – 1D algebraic mesh generation on the domain boundaries

The calculation of the mesh nodes distribution on the boundaries (geometric elements defined in the drawing stage) is carried out using an algebraic method. Let´s consider, for simplicity, a line segment (for an arc geometric element, the method is analogous). Given the total number of mesh nodes in the element and the mesh spacings ΔL_1 and ΔL_2 (cf. Figure A.9) in the element edges, the problem is to find the mesh distribution along the whole element.

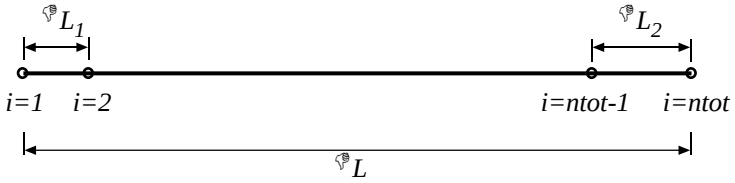


Figure A.9 – Mesh generated using the Laplace system (*i.e.*, without control functions).

Let´s suppose that the line segment length is $\Delta L = 2.5\,m$ and the total number of nodes is $ntot = 26$. If a uniform mesh spacing is considered, $\Delta L_1 = \Delta L_2 = 0.1\,m$. If the imposed value for ΔL_1 and/or ΔL_2 is not *0.1 m*, the mesh spacing distribution will be non-uniform. To exemplify, let´s consider $\Delta L_1 = 0.025\,m$ and $\Delta L_2 = 0.05\,m$. The solution method is as follows:

1. A first solution (solution *A*) for the mesh distribution is computed so as to satisfy the imposed ΔL_1 value, using a constant expansion factor (i.e., a constant value for the ratio between two consecutive mesh spacings). Figure A.10 shows the obtained solution. Note that the imposed ΔL_2 value is not satisfied in this solution.

2. A similar procedure is followed to compute a second solution (solution *B*) that satisfies the imposed ΔL_2 value (cf. Figure A.11). Note that the imposed ΔL_1 value is not satisfied.

3. The final solution, satisfying both ΔL_1 and ΔL_2 imposed values is obtained by properly combining both solutions, as follows:

$$x_i = (1-f)\,x_A + f\,x_B \quad ; \quad 3 \le i \le ntot-1 \tag{5.1}$$

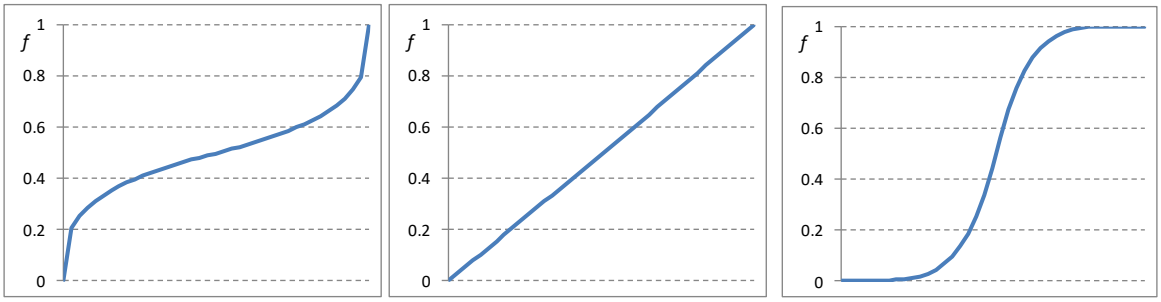
The weighting factor *f* is given by:

$$\begin{aligned} i \le \frac{ntot}{2} \quad &\Rightarrow \quad f_i = \left[2\frac{(i-2)}{ntot-3} \right]^c / 2 \\ i > \frac{ntot}{2} \quad &\Rightarrow \quad f_i = 1 - \left[2\frac{(ntot-1-i)}{ntot-3} \right]^c / 2 \end{aligned} \tag{5.2}$$

The exponent *c* in the previous equation (*Decay*, in section B1.3.2) controls how *f* changes across the element. Figure A.17 displays the weighting factor for different values of the exponent *c*.

Figure A.10 – Solution *A*: 1D mesh for a constant expansion factor, satisfying the imposed ΔL_1 value.

Figure A.11 – Solution *B*: 1D mesh for a constant expansion factor, satisfying the imposed ΔL_2 value.



(a) (b) (c)

Figure A.12 – Weighting factor *f* along the geometric element. (a) *c = 0.3*; (a) *c = 1*; (a) *c = 5*;

Figure A.13 and Figure A.14 show the final result for two different values of the exponent *c*.

Figure A.13 – Final 1D mesh distribution in a line segment. *A*: $c = 5$; $\Delta L_1 = 0.025\,m$; $\Delta L_2 = 0.05\,m$.

Figure A.14 – Final 1D mesh distribution in a line segment. *A*: $c = 1$; $\Delta L_1 = 0.025\,m$; $\Delta L_2 = 0.05\,m$.

In these two examples, the mesh size increases from both edges of the line segment into the middle region. Of course, the final result depends on the segment length, total number of mesh nodes and mesh spacings ΔL_1 and ΔL_2. Figure A.17 and Figure A.18 represent an example where the mesh spacing is contracting towards the segment middle.

Figure A.15 – Final 1D mesh distribution in a line segment. *A*: $c = 0.3$; $\Delta L_1 = 0.3\,m$; $\Delta L_2 = 0.3\,m$.

Figure A.16 – Final 1D mesh distribution in a line segment. *A*: $c = 0.8$; $\Delta L_1 = 0.3\,m$; $\Delta L_2 = 0.3\,m$.

A5.2 – 2D numerical generation of structured mesh in the inner domain

Numerical generation of structured meshes represents a good and practical method to achieve some of the characteristics described previously. The numerical procedure implemented in **EasyCFD** closely follows the method described in Steger and Sorenson (1979) and Thomas and Middlecoff (1979).

Mesh generation is a procedure for the calculation of the Cartesian coordinates (*x*, *z*) of each mesh node, *i.e.*, a process of finding a relationship between computational coordinates (ξ, ζ) and Cartesian coordinates (*x*, *z*). Numerical mesh generation using *Laplace* differential equations (and *Poisson* differential equations) is mostly used due to the associated smoothness of the meshes produced by these methods. The *Laplace* system has the following formulation:

$$\begin{aligned} \frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial z^2} &= 0 \\ \frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial z^2} &= 0 \end{aligned} \qquad (5.3).a$$

or

$$\begin{aligned} \xi_{xx} + \xi_{zz} &= 0 \\ \zeta_{xx} + \zeta_{zz} &= 0 \end{aligned} \qquad (9.3).b$$

This equation system states, in fact, that the *x* and *z* coordinates of a mesh node is given as the arithmetic average of the *x* and *z* coordinates of its neighbour nodes. Figure A.17 shows a mesh generated with the Laplace system. The boundary nodes distribution is given as input and the inner nodes distribution is computed with the differential equations (5.3). Due to its averaging characteristics, the Laplace system tends to produce an equally nodes distribution inside the domain. The non-uniform nodes distribution in regions *1* and *2* of Figure A.17 is lost inside the domain (which may be a disadvantage) and the nodes clustering at the lower boundary is non-uniform, i.e., the

distance of the first computed ξ line to the lower boundary is increased in convex regions (like region *3*) and decrease in concave regions (like region *4*).

The remedy to these problems is to use **control functions**. The Poisson system, which incorporates control functions, is:

$$\begin{aligned} \xi_{xx} + \xi_{zz} &= P_c + P_t \\ \zeta_{xx} + \zeta_{zz} &= Q_c + Q_t \end{aligned} \tag{5.4}$$

In this set of *Poisson* equations, the terms P_c, Q_c are forcing terms which allow the control of the mesh lines clustering near the boundaries (to avoid the effects shown in regions *3* and *4* of Figure A.17). In turn, P_t, Q_t are forcing terms to transmit the mesh spacing into the interior of the domain (to allow the nodes spacing of regions *1* and *2*, in Figure A6.1, to be felt in the inner domain)

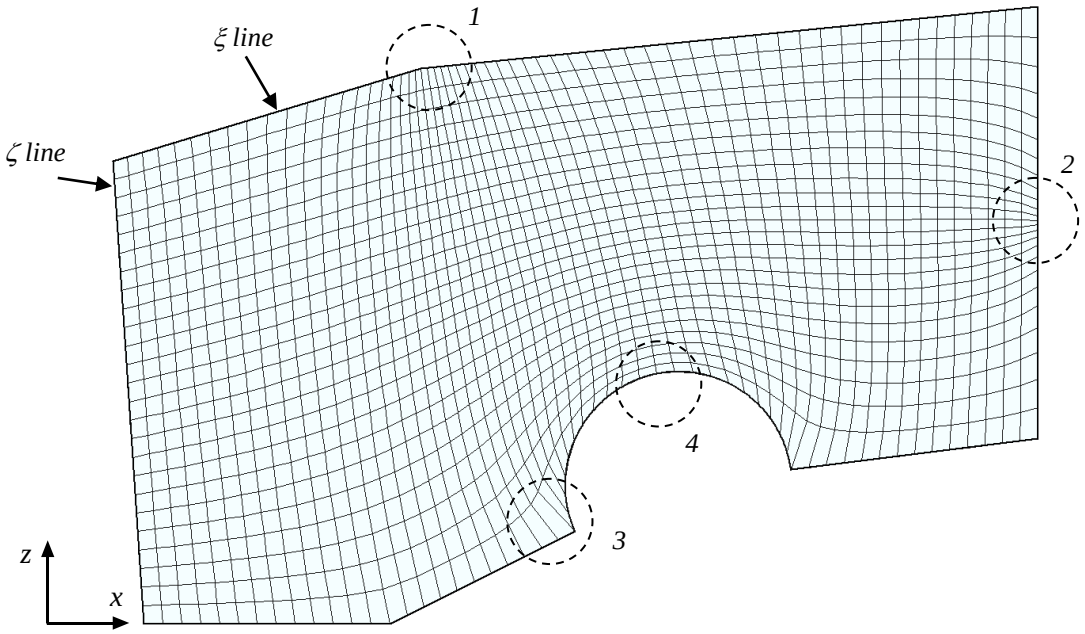


Figure A.17 – Mesh generated using the Laplace system (*i.e.*, without control functions).

Since the unknowns in equations (5.4) are the Cartesian coordinates, the solution of these equations must be carried out in the computational domain. For that purpose, the dependent and independent variables are interchanged, yielding:

$$\begin{aligned} g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\zeta} + g_{11}x_{\zeta\zeta} &= -J^2\left[\left(P_c + P_t\right)x_\xi + \left(Q_c + Q_t\right)x_\zeta\right] \\ g_{22}z_{\xi\xi} - 2g_{12}z_{\xi\zeta} + g_{11}z_{\zeta\zeta} &= -J^2\left[\left(P_c + P_t\right)z_\xi + \left(Q_c + Q_t\right)z_\zeta\right] \end{aligned} \tag{5.5}$$

where the covariant metric relations g_{ij} are as defined by equations (2.58) to (2.60) and the Jacobian J is as defined by equation (2.3).

A5.2.1 – Control functions: Clustering and orthogonality at boundaries

Steger and Sorenson (1979) proposed, for two dimensions, an efficient method for, simultaneously, achieve orthogonality and a pre-specified mesh clustering (normal distance to a boundary) near the boundaries.

Let's assume that clustering and orthogonality is to be controlled near the lower boundary defined by a ξ line[2], as depicted in Figure A.18. This boundary is characterized by $\zeta = \zeta_1$. The forcing terms (or control functions) P_c and Q_c assume values P_{c1} and Q_{c1} at this boundary and an exponential decay is considered into the inner domain, as given by the following equations:

$$P_c = P_{c1}\, e^{-a(\zeta-\zeta_1)} \tag{5.6}$$

$$Q_c = Q_{c1}\, e^{-b(\zeta-\zeta_1)} \tag{5.7}$$

where a, b are decay factors which control how far and with which intensity the control function effects are felt in the inner domain.

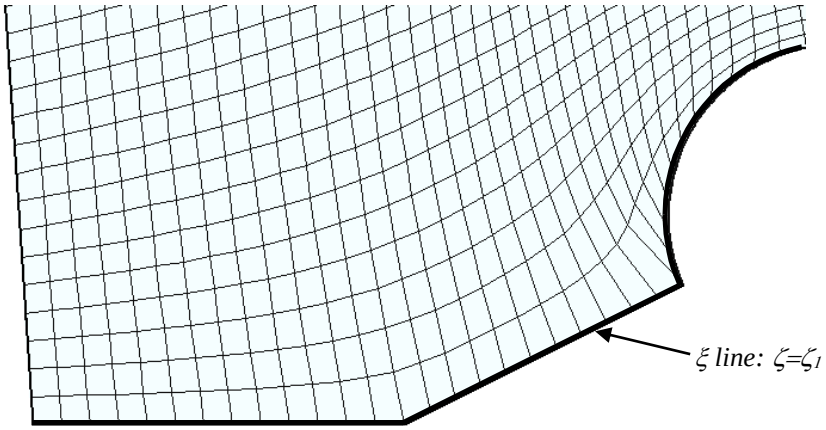


Figure A.18 – Definition of the lower boundary $\zeta = \zeta_1$.

To compute P_{c1} and Q_{c1}, equations(5.5) are evaluated at $\zeta = \zeta_1$, yielding the following set:

$$\begin{aligned} P_{c1} &= \left(\frac{z_\zeta R_1 - x_\zeta R_2}{J} \right)_{\zeta=\zeta_1} \\ Q_{c1} &= \left(\frac{-z_\xi R_1 + x_\xi R_2}{J} \right)_{\zeta=\zeta_1} \end{aligned} \tag{5.8}$$

with:

[2] ξ lines are defined as lines along which the computational coordinate ξ varies. They are characterized by a constant value of the ζ computational coordinate. A similar reasoning applies to ζ lines.

$$R_1 = \left(-\frac{g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\zeta} + g_{11}x_{\zeta\zeta}}{J^2} \right)_{\zeta=\zeta_1}$$
$$R_2 = \left(-\frac{g_{22}z_{\xi\xi} - 2g_{12}z_{\xi\zeta} + g_{11}z_{\zeta\zeta}}{J^2} \right)_{\zeta=\zeta_1} \tag{5.9}$$

Note that, in this derivation, control functions P_t and Q_t were set to zero.

In the previous equations (5.8) and (5.9), all the derivatives in ξ are known, since the location (x and z coordinates) of the mesh nodes on the boundary ζ_1 is specified as imposed condition.

The derivatives in ζ are computed as a function of the (imposed) distance Δs of the first mesh nodes $\zeta = \zeta_2$ to the surface $\zeta = \zeta_1$ (clustering distance). It may be proven that, based on the orthogonality condition, the following relations (which include derivatives solely with respect to ξ direction) apply:

$$\left(x_\zeta\right)_{\zeta=\zeta_1} = \Delta s \left(\frac{-z_\xi}{\sqrt{g_{11}}} \right)_{\zeta=\zeta_1} = \Delta s \left(\frac{-z_\xi}{\sqrt{x_\xi^2 + z_\xi^2}} \right)_{\zeta=\zeta_1}$$
$$\left(z_\zeta\right)_{\zeta=\zeta_1} = \Delta s \left(\frac{x_\xi}{\sqrt{g_{11}}} \right)_{\zeta=\zeta_1} = \Delta s \left(\frac{x_\xi}{\sqrt{x_\xi^2 + z_\xi^2}} \right)_{\zeta=\zeta_1} \quad s \tag{5.10}$$

Evaluation of the second derivatives with respect to ζ, at $\zeta = \zeta_1$, in equations (5.9), is made using a one-sided second order discretisation, as follows:

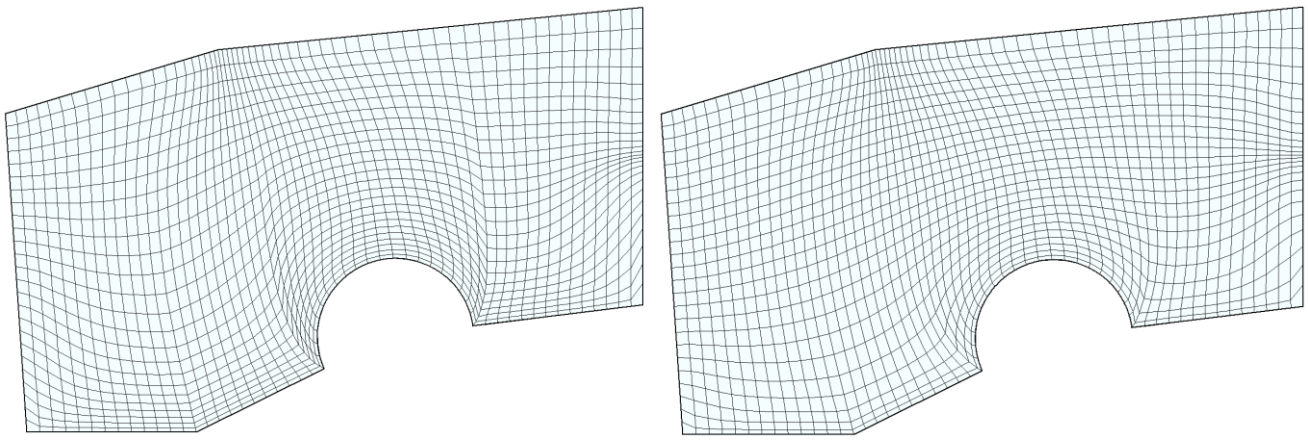
$$\left(x_{\zeta\zeta}\right)_{\zeta=\zeta_1} = \left(\frac{-7x_1 + 8x_2 - x_3}{2} \right) - 3\left(x_\zeta\right)_{\zeta=\zeta_1}$$
$$\left(z_{\zeta\zeta}\right)_{\zeta=\zeta_1} = \left(\frac{-7z_1 + 8z_2 - z_3}{2} \right) - 3\left(z_\zeta\right)_{\zeta=\zeta_1} \tag{5.11}$$

where the subscript 1, 2, 3 indicates the position of the mesh node relative to the boundary, along the ζ line.

The equations derived previously allow the determination of the forcing terms P_c, and Q_c as a function of the imposed clustering distance Δs.

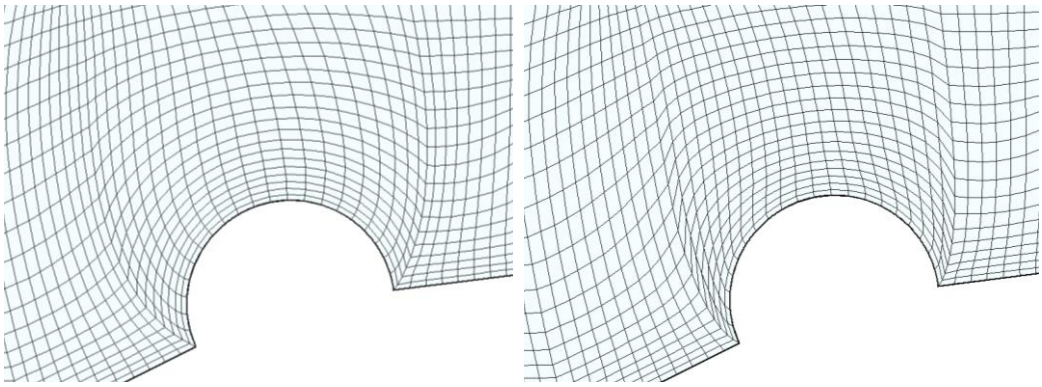
Coefficient *b*, in equation (5.7), controls the attraction of the ξ lines (clustering) into the boundary (cf. Figure A.19)

Coefficient *a*, in equation (5.6), determines how deep inside the computational domain the orthogonality condition is transmitted (cf. Figure A.20).



(a) (b)

Figure A.19 – Effect of the value for the *b* clustering decay factor. (a) *b = 0.2*; (b) *b = 0.8*.



(a) (b)

Figure A.20 – Effect of the value for the *a* orthogonality decay factor. (a) *a = 0.1*; (b) *a = 1.2*.

Similar equations are considered for imposition of the mesh clustering at other boundaries (top, east and west, in the computational domain).

A5.2.2 – Control functions: Transmission of the mesh spacing into the interior of the domain

Transmission of the boundary mesh spacing into the inner domain is accomplished with the control functions proposed by Thomas and Middlecoff (1979). The terms P_t and Q_t, in equations (5.4) and (5.5) have the following formulation:

$$P_t = \frac{g_{22}\,\phi}{J^2} \tag{5.12}$$

$$Q_t = \frac{g_{11}\,\omega}{J^2} \tag{5.13}$$

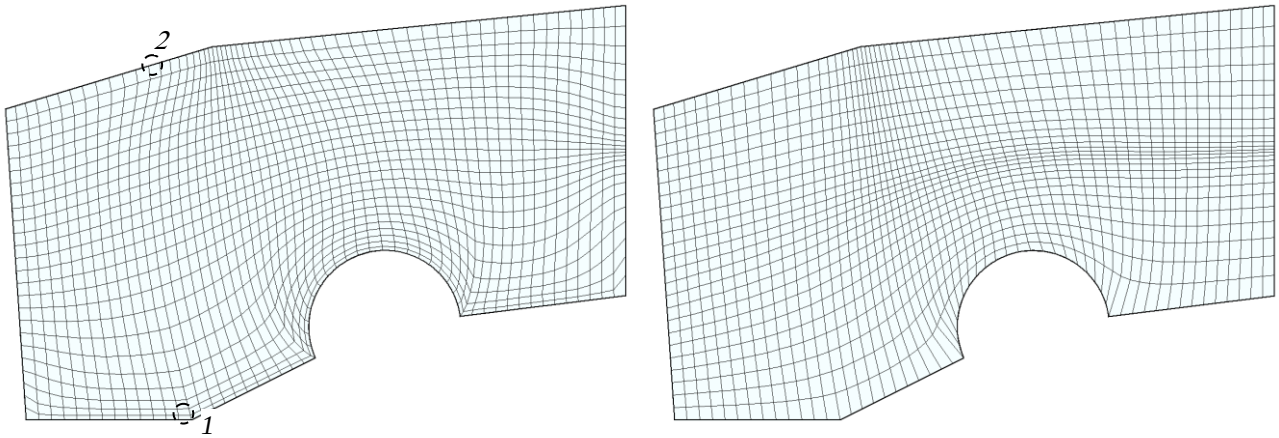
where

$$\phi = \frac{x_\xi x_{\xi\xi} + z_\xi z_{\xi\xi}}{x_\xi^2 + z_\xi^2} \tag{5.14}$$

$$\omega = \frac{x_\zeta x_{\zeta\zeta} + z_\zeta z_{\zeta\zeta}}{x_\zeta^2 + z_\zeta^2} \tag{5.15}$$

The term ϕ, responsible for the transmission of the ζ lines spacing, is directly evaluated at two opposite boundaries (location pairs like *1* and *2* in Figure A.21). The corresponding values inside the domain are obtained through a linear interpolation. A similar procedure is followed for the term ω, responsible for the transmission of the ξ lines spacing. The remaining terms, necessary for the computations of P_t and Q_t, are evaluated directly in each inner node.

Figure 6.4 shows the effect of these control functions. In these meshes, Steger and Sorenson control functions for mesh clustering and orthogonality were also employed.



(a) (b)

Figure A.21 – Effect of the transmission of the mesh spacing into the inner domain (orthogonality and mesh clustering is also activated). (a) without transmission; (b) with transmission.

A5.2.3 – Solution of the equations

For their solution, equations (5.5) are cast in the following form:

$$\begin{aligned} a_P x_P &= a_E x_E + a_O x_O + a_T x_T + a_B x_B + b_x \\ a_P z_P &= a_E z_E + a_O z_O + a_T z_T + a_B z_B + b_z \end{aligned} \tag{5.16}$$

where:

$$a_E = (g_{22})_E \;\; ; \;\; a_O = (g_{22})_O \;\; ; \;\; a_T = (g_{11})_T \;\; ; \;\; a_B = (g_{11})_B \tag{5.17}$$

$$\begin{aligned} b_x &= 2g_{12}x_{\xi\zeta} - J^2\left[(P_c + P_t)x_\xi + (Q_c + Q_t)x_\zeta\right] \\ b_z &= 2g_{12}z_{\xi\zeta} - J^2\left[(P_c + P_t)z_\xi + (Q_c + Q_t)z_\zeta\right] \end{aligned} \tag{5.18}$$

In these equations, second derivatives are evaluated using central differences. First derivatives are evaluated using forward or backward differences, depending on the sign of the corresponding coefficient. Solution of these equations is carried out using a Point Successive Over Relaxation method (PSOR), where optimum relaxation parameter is computed at every node in the mesh following the procedure proposed by Ehrlich (1979). Control functions are introduced only after a given number of

iterations defined by the user (cf. section B1.3.8). Steger and Sorenson control functions need heavy under-relaxation at the beginning of the iterative process. The under-relaxation factor starts at zero (total under-relaxation) and then is allowed to increase gradually (lesser under-relaxation) following an exponential-type law.

A5.3 – 2D numerical generation of unstructured mesh in the inner domain

The unstructured mesh generation method (called Paving Method) implemented in **EasyCFD** closely follows the proposal described in detail in Blaker and Stephenson (1991). Only a brief overview of the procedure is given here. Please refer to Blaker and Stephenson (1991) for details.

Nodes coordinates are computed starting from the boundary nodes distribution, already computed as described in A5.1. Thus, a paving row (row of nodes which is being computed) starts from a boundary row (row which lies in a geometric boundary), by adding nodes to the previous row (Figure A.22). The process continues until the whole domain is filled (Figure A.23).

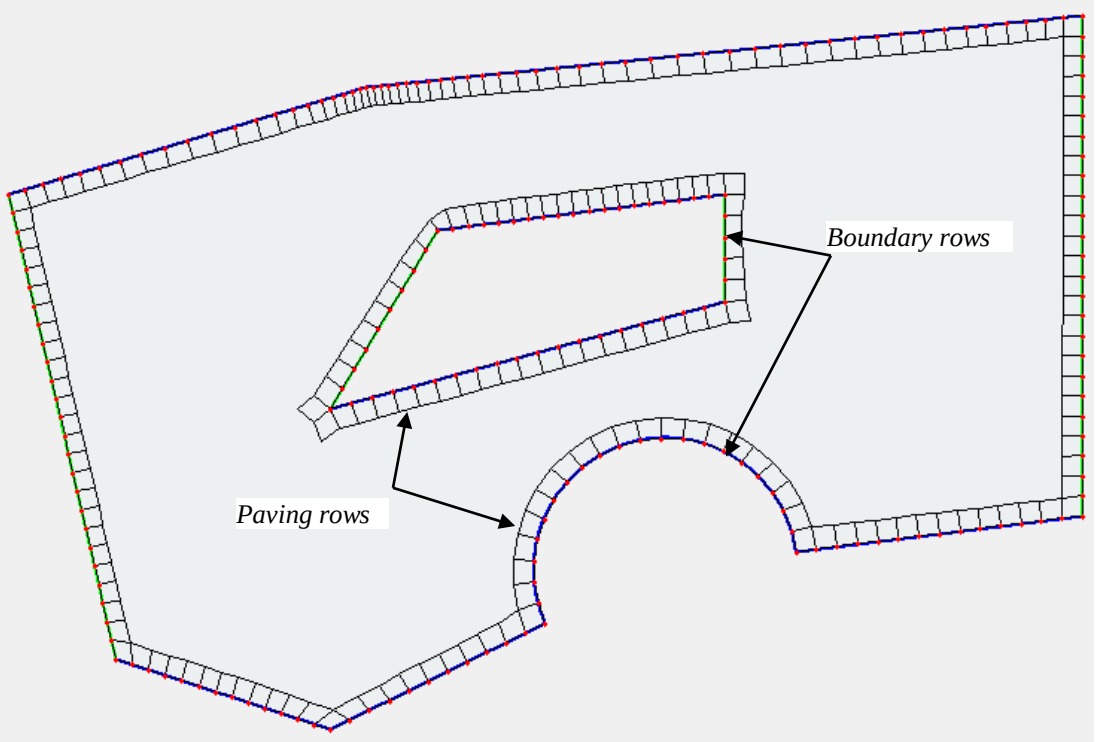


Figure A.22 – This geometry is composed by two boundary rows, corresponding to the envelope and an inner block.

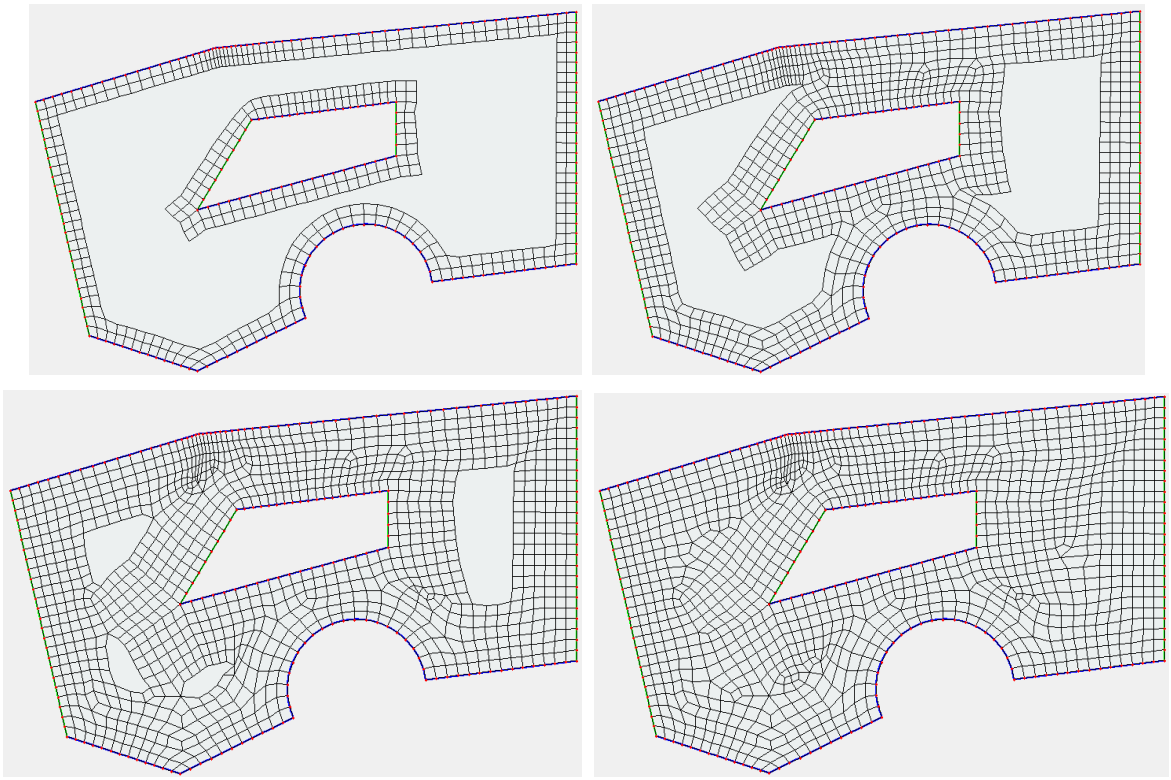


Figure A.23 – Paving process.

A5.3.1 – Row projection

Each row is generated by projecting each node from the previous row by an appropriate distance (projecting distance), in the appropriate direction. Depending on the local characteristic angle of the previous row, four types of projection may exist: end, side, corner and reversal.

Row end projection

In this case, each node in the previous row generates a single node in the new row. One new control volume (also called, in this context, a new element) is thus generated. Three nodes in the previous row are used for the new element. This type of projection is applied for values of angle θ (cf. Figure A.24) in the range:

$$\theta < 180^{\circ} - \sigma_1 \tag{5.19}$$

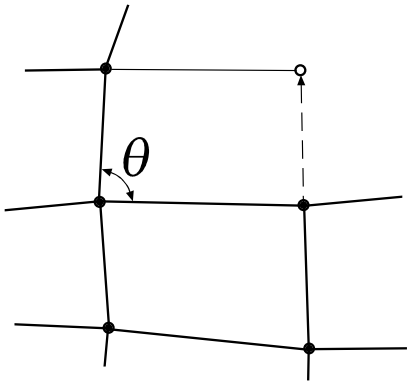


Figure A.24 – Row end projection.

Row side projection

In this case, each node in the previous row generates a single node in the new row, corresponding to the generation of one new element. Only two nodes in the previous row are used for the new element. This type of projection is used for values of angle θ (cf. Figure A.25) in the range:

$$180^\circ - \sigma_1 \leq \theta \leq 180^\circ + \sigma_2 \tag{5.20}$$

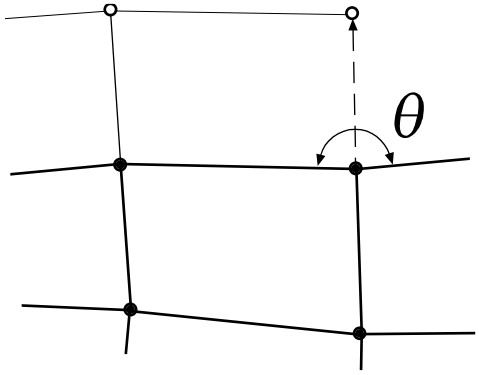


Figure A.25 – Row side projection.

Row corner projection

In this case, each node in the previous row generates three nodes in the new row, corresponding to two new elements. This type of projection is used for values of angle θ (cf. Figure A.26) in the range:

$$180^\circ + \sigma_2 < \theta \leq 360^\circ - \sigma_3 \tag{5.21}$$

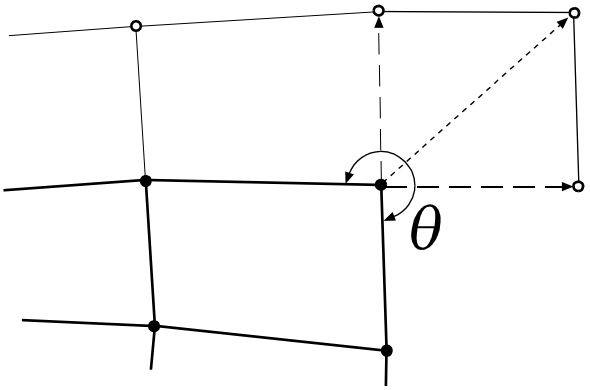


Figure A.26 – Row corner projection.

Row reversal projection

In this case, each node in the previous row generates four nodes in the new row, corresponding to three new elements. This type of projection is used for values of angle θ (cf. Figure A.27) in the range:

$$360^\circ - \sigma_3 \leq \theta < 360^\circ \tag{5.22}$$

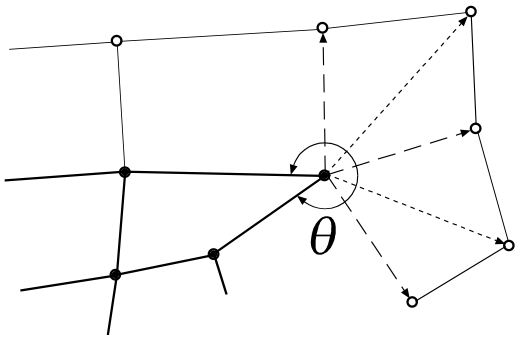


Figure A.27 – Row reversal projection.

Default values are $\sigma_1 = 55^\circ$; $\sigma_2 = 60^\circ$; $\sigma_3 = 50^\circ$. By changing these values (cf. section B1.4.3), different types of row projections can be achieved for the same feature. Some examples are given below.

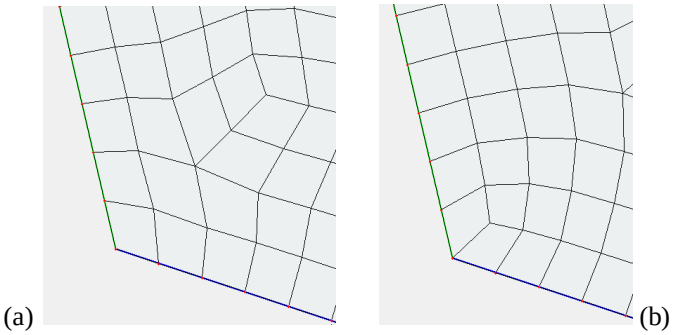


Figure A.28 – Feature mapped as: (a) row end; $\sigma_1 = 55^\circ$; (b) row side; $\sigma_1 = 60^\circ$.

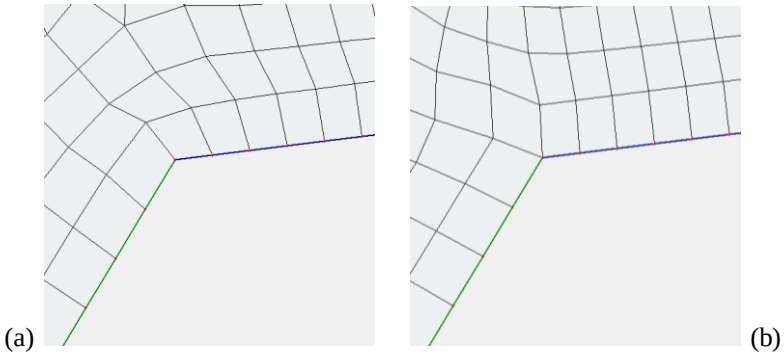


Figure A.29 – Feature mapped as: (a) row side; $\sigma_2 = 60^\circ$; (b) row corner; $\sigma_2 = 50^\circ$.

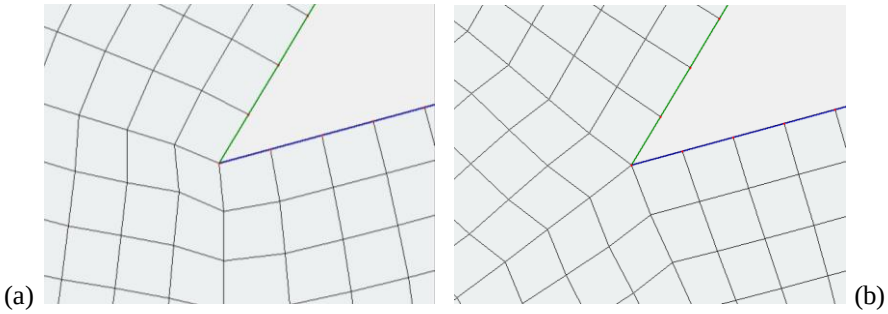


Figure A.30 – Feature mapped as: (a) row corner; $\sigma_3 = 60^\circ$; (b) row reversal; $\sigma_3 = 60^\circ$.

A5.3.2 – Seaming of rows

Seaming of a row is the process of closing a crack that may form in the paving row. A crack occurs when the paving row angle is very small (or even negative). During this process, two nodes are combined into a single one, resulting in an increased paving row angle (cf. Figure A.31).

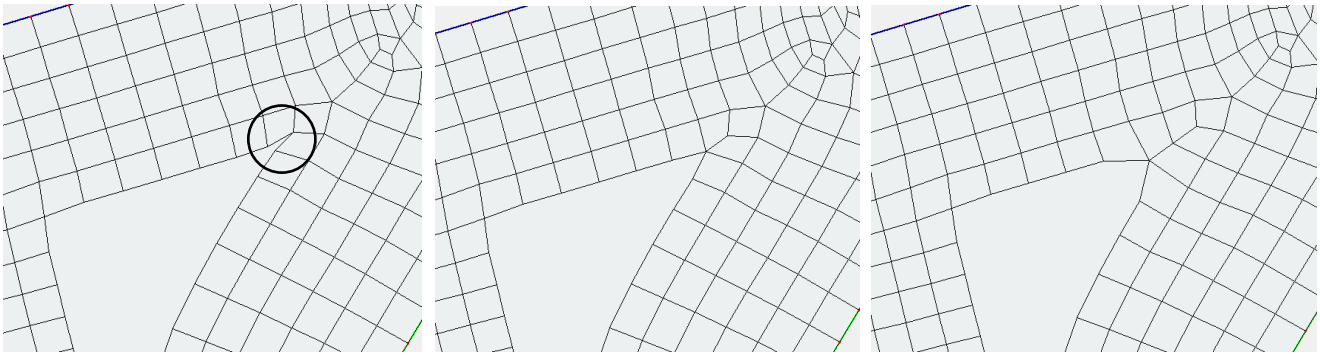


Figure A.31 – Seaming process for a small paving row angle.

A5.3.3 – Intersection of two rows

When two rows intersect, the intersection point is computed and the closest nodes in each row are combined to form a single node. The remaining row is seamed, leading to a single row. Figure A.32 and Figure A.33 illustrate the process.

A5.3.4 – Self intersection of a row

When a row self intersects, the intersection point is computed and the closest nodes are combined into a single one. The row is then seamed, generating two different rows. This is illustrated in Figure A.34.

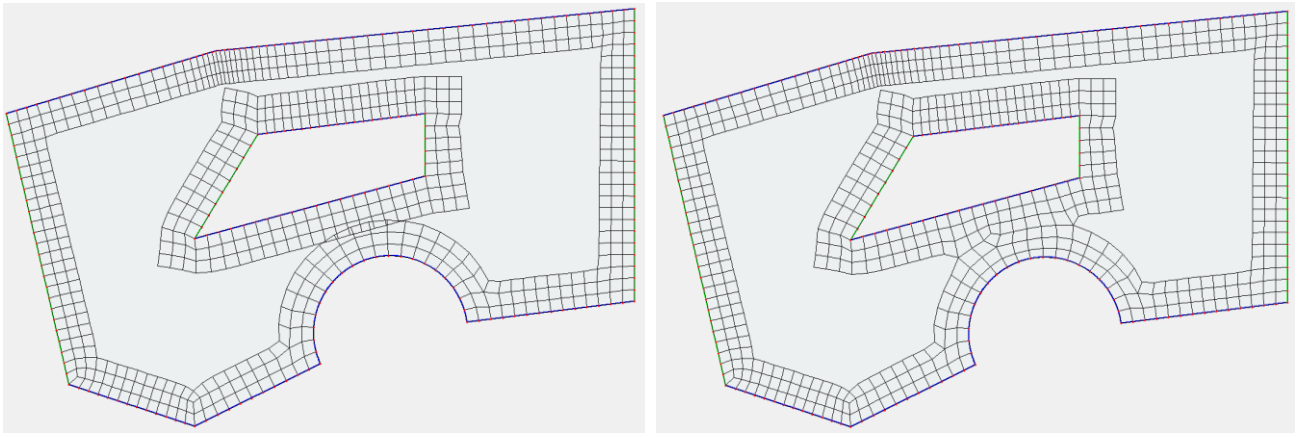


Figure A.32 – Intersection of two rows: general view.

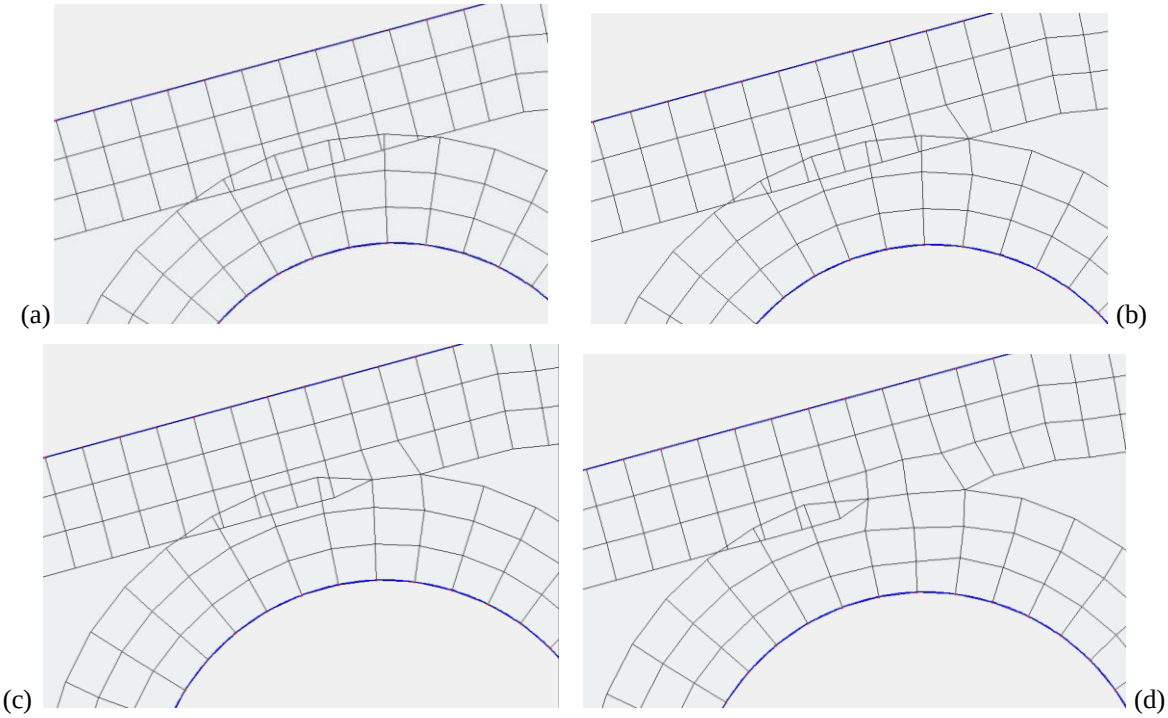


Figure A.33 – Intersection of two rows: some details of the process.

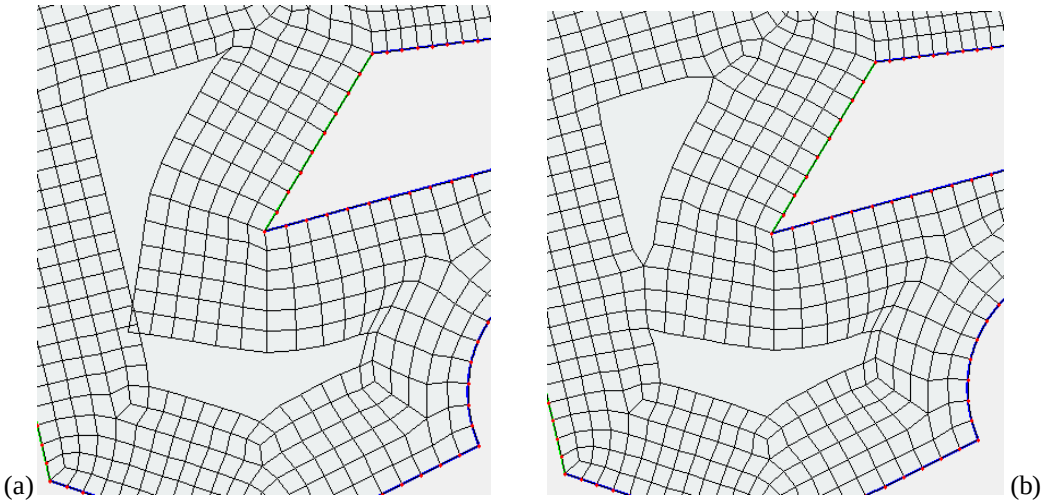


Figure A.34 – Self intersection of a row.

A5.3.5 – Control of mesh size

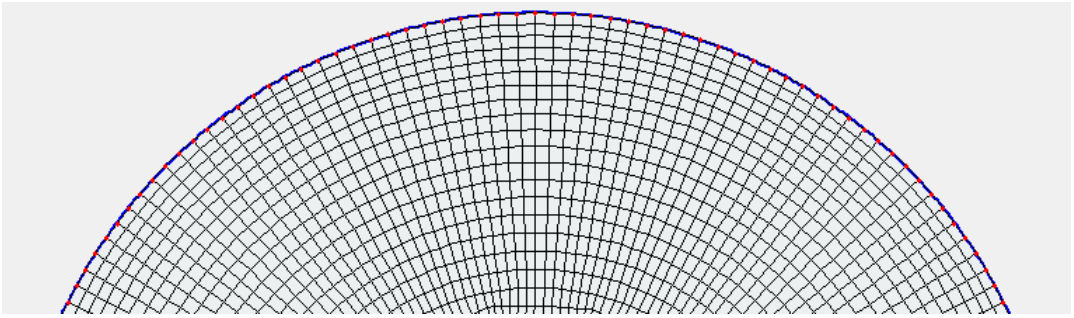
Mesh size should be kept between specified values. In what concerns projection distance, this is done directly through the assignment of the projection value. Projection value starts at the clustering distance (cf. B1.4.2) and is progressively increased as rows are build, through a constant expanding factor (cf. B1.4.2), up to the specified maximum for the inner mesh size (cf. B1.4.1).

Distances parallel to row direction are kept within certain limits through the insertion of tucks and wedges, as described next.

Minimum mesh spacing

Control of the minimum mesh spacing is especially important for concave boundaries, where mesh spacing tends to decrease as new rows are built. This is achieved through the insertion of tucks. Basically, inserting a tuck consists in combining two elements into a single one, as depicted in Figure A.35.

(a)



(b)

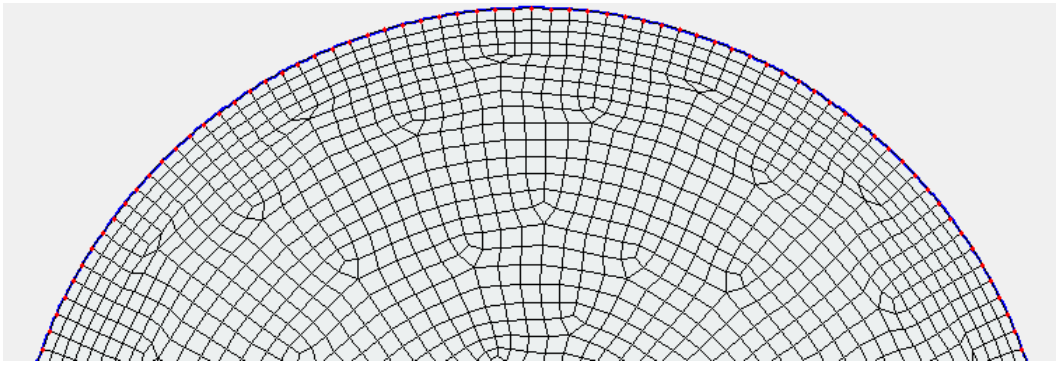
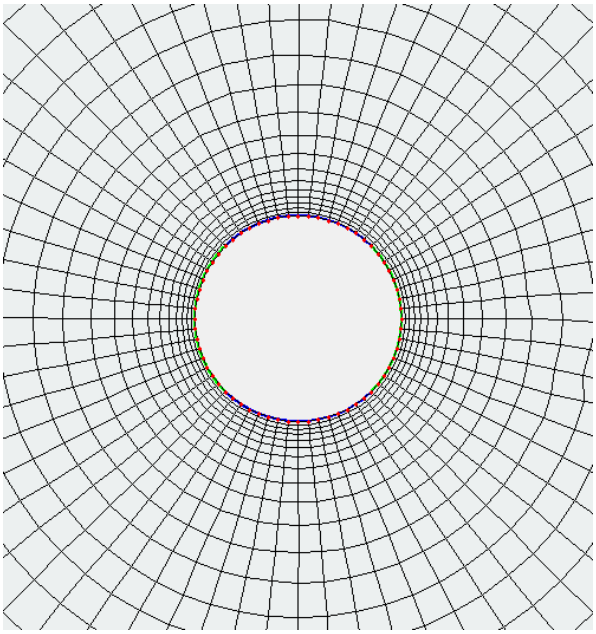


Figure A.35 – Controlling the minimum mesh size through the insertion of tucks; (a) without tucks; (a) with tucks.

Maximum mesh spacing

Control of the maximum mesh spacing is especially important for convex boundaries, where mesh spacing tends to increase as new rows are built. This is achieved through the insertion of wedges. Inserting a wedge consists in inserting a new element at a certain location, as depicted in Figure A.36.

(a)



(b)

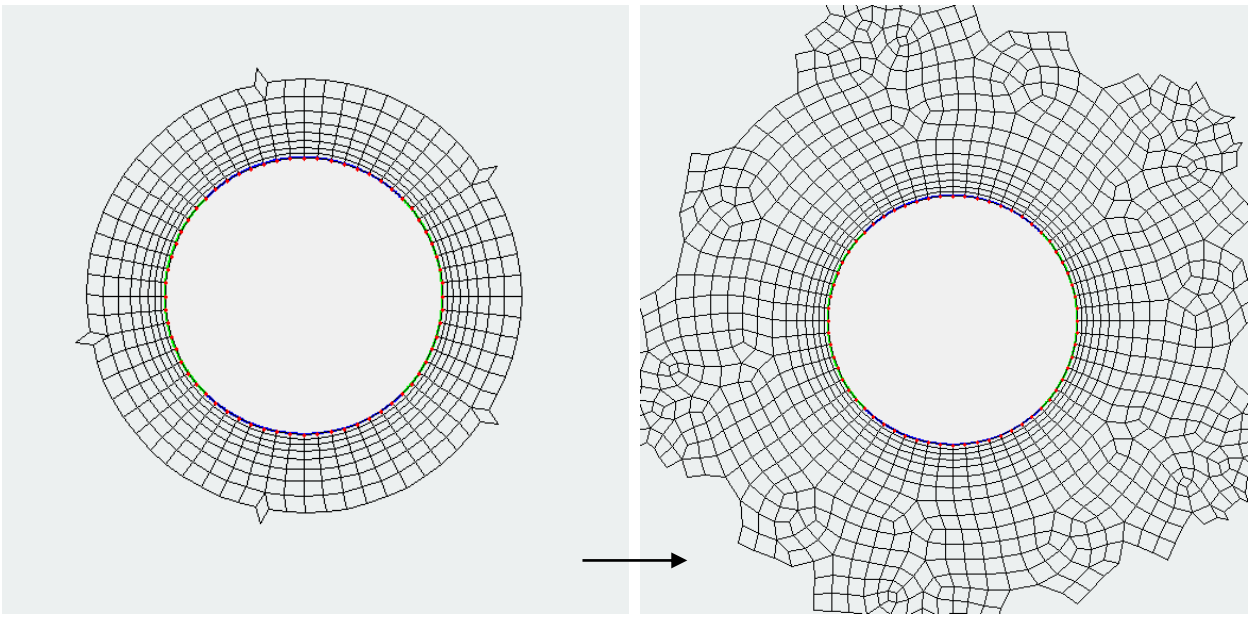


Figure A.36 – Controlling the maximum mesh size through the insertion of wedges; (a) without wedges; (b) with wedges.

B– Working with EasyCFD

Working with a problem goes through three major stages to be accomplished sequentially:

- *Problem Definition*
- *Solver*
- *Post-Processing*

Each of these stages is described next. In the following exposition, references to objects of the graphical interface are made using the *italic*.

B1 – Problem Definition

B1.1 – Brief Description of the Interface

Figure B.1 represents the graphical interface for the *Problem Definition*. Several spatial regions (*Spaces*) are identified, for easier reference in the remaining text. The working stage is selected in *Space 1*. Each of these stages can only be selected if the precedent stage was already completed.

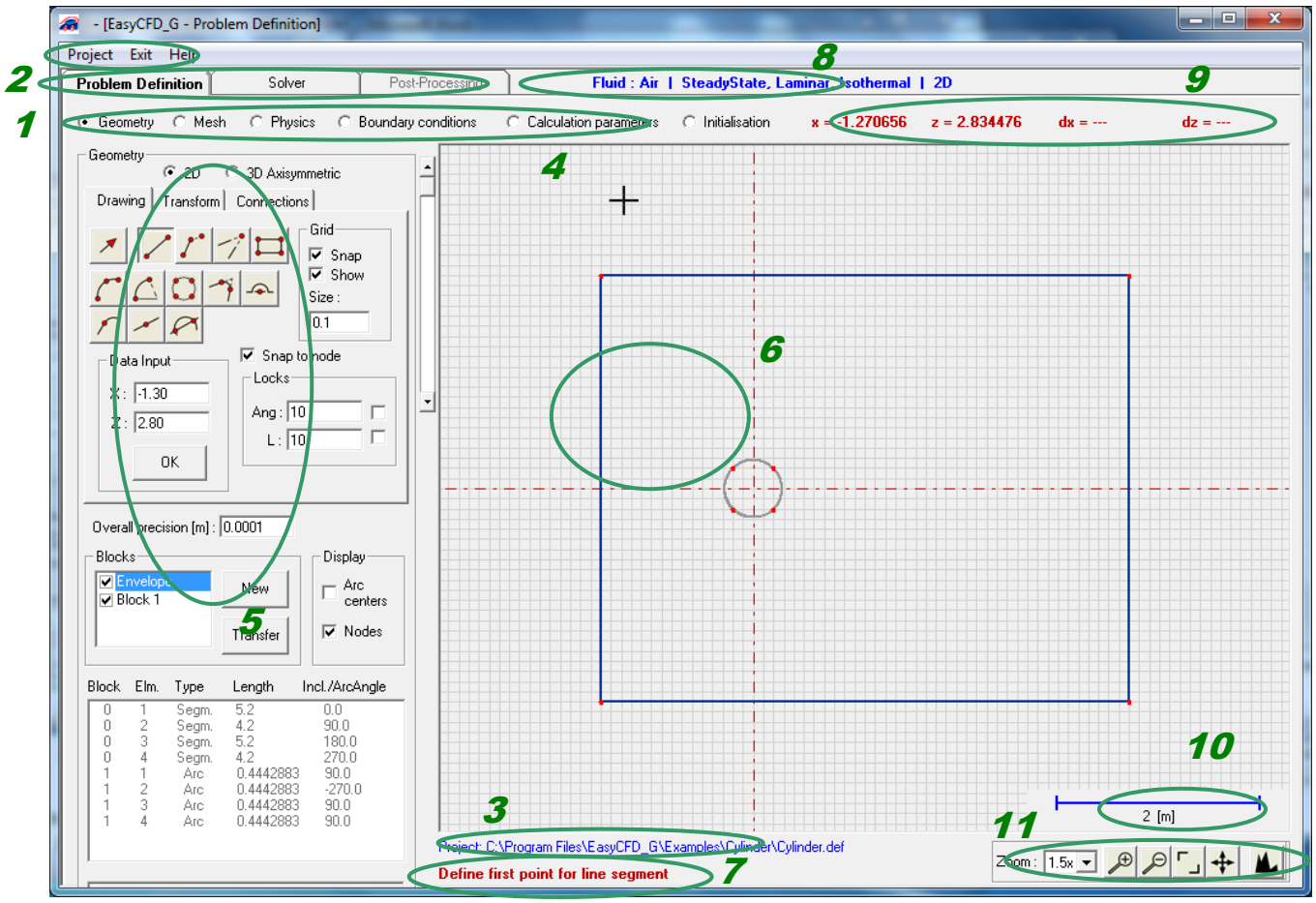
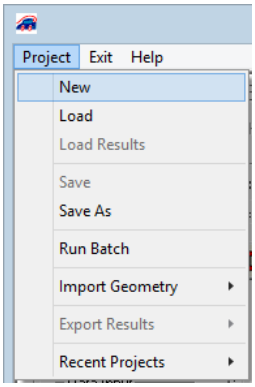
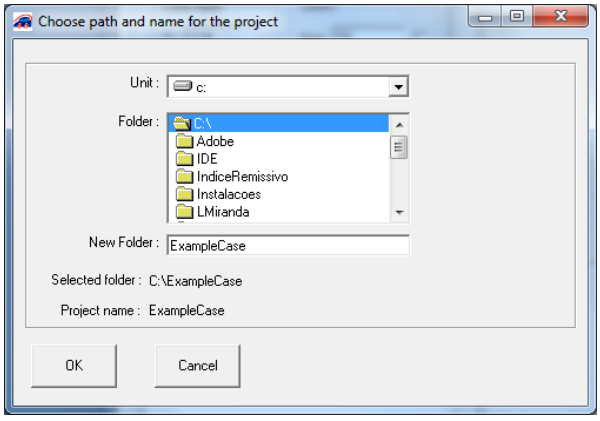


Figure B.1 – Graphical interface for the *Problem Definition*.

A new problem (Project) can be started by clicking *Project – New* (this will erase any loaded project). If there is no loaded Project, you can simply choose *Save As* to assign a name to the current Project





When creating a Project, it will be saved in a folder with a name similar to the project´s name. If the does not

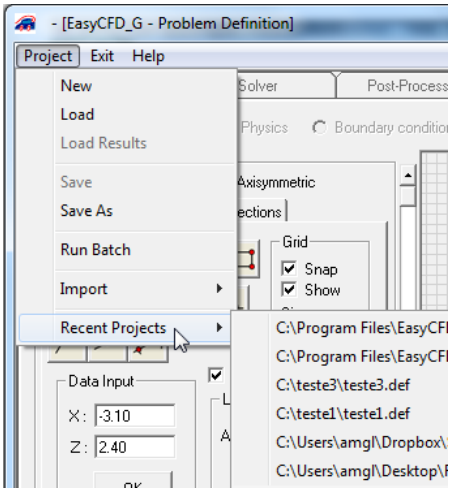
exist, a new one is automatically created. The following files are created in the project folder (taking, as example, the name *ExampleCase*):

- *ExampleCase.def*: file containing all the necessary information for the solver.
- *ExampleCaseST.res or ExampleCaseTR.Res:* file with the calculation results, for steady-state and for transient problems, respectively. This file, along with the *ExampleCase.def* file, contains all the necessary information for the post-processing of the results.

For transient problems, intermediate results are stored in the folder *Trans*, in files named *Tr1.res*, *Tr2.res*, etc.

Other files and folders created during the *Post-Processing* are described in sections B3.9 and B3.10.

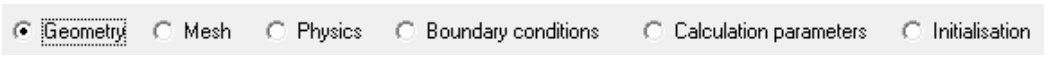
To load a previous Project, choose *Load*, or *Recent Projects* for a list of the 6 most recent projects.



The loaded project name is displayed in *Space 3* (cf. Figure B.1).

After loading a project, you may also load previously computed results, either to post-process them or, for example, to start or continue a transient simulation from the present results (this functionality is also available in the Initialization of Variables; cf. section 0).

The *Option-Buttons* shown below list the several phases for defining a problem (*Space 4*). *Space 5* contains controls specific for each phase.



Space 6 is the visualisation interface.

Space 7 displays information messages and tips as necessary.

Space 8 presents a brief summary of the problem physics.

Space 9 displays the coordinates at the cursor location and distances from the previous location.

Space 10 displays a rule with a reference length.

Space 11 contains several tools for visualisation control, as described next.

The tools for the visualisation control are:

When the user clicks on the visualisation area, a "zoom in" is made with the factor specified in *Zoom*. There are two ways of doing the zoom: a) if the *Shift* key is not pressed, the images will be centred in the click point; b) if the *Shift* key is pressed, the click point remains unchanged.

 A "zoom out" is made with the factor specified in *Zoom*, when the user clicks on the visualisation area. The *Shift* key works in a similar way as previously described.

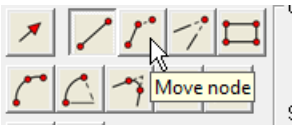
 "Zoom in" for the rectangular region selected by the user (usage: click on a diagonal side and then click on the opposite corner of the diagonal).

 Pan the visualisation area (usage: click on a location and then, keeping the mouse button pressed, drag the mouse). Note: the same functionality is achieved by pressing the mouse middle button, independently of the selected zooming tool).

 Fits the whole geometry to the visualisation area.

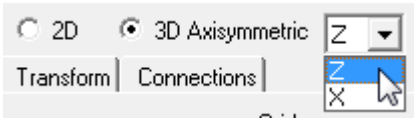
The visualisation control tools are deselected with the mouse right button.

You may obtain additional information about each control or command by letting the mouse pointer for some time over the object (a small text is displayed near the cursor; cf. at right).



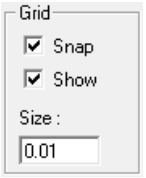
B1.2 – Geometry Definition

B1.2.1 – Choosing 2D or 3D axisymmetric problems



The type of problem may be chosen between 2D and 3D Axisymmetric. For the later option, the axisymmetry axis may be the x or the z axis and the geometry must be defined in the corresponding positive area (for example, if *z* is the axisymmetry axis, the geometry must be defined for x > 0).

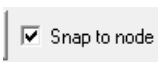
B1.2.2 – Drawing Aids



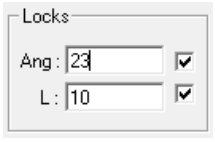
Dimension units are **meters**. Angles are given in **degrees**.

The geometry may be composed by two types of geometric elements: **Line Segments** and **Arcs**. Two contiguous geometric elements are connected by a geometric **Node**.

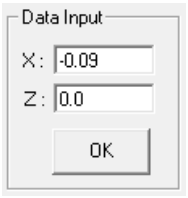
- To help drawing, the *Snap* feature may be activated. This enforces all selected locations to snap to the **drawing grid** (cf. Figure B.2). The drawing grid visibility may be turned on or off (*check-button Show*). Its *Size* is also adjustable.
- Drawing locations may be forced to snap to existing geometric nodes (*Snap to node*).



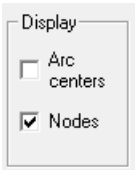
- Distances and/or angles may be locked to help drawing.



Instead of clicking on the drawing interface to define a location, the corresponding coordinates may be entered by keyboard (*Data Input*).



- By holding down the Shift key, angles are enforced to 45º increments.



- *Display* of *Arc centres* and geometric *Nodes* may be switched on or off.

- A white arrow cursor indicates that no tool is selected (**default state**). This state is recovered by clicking the mouse right-hand button.

- A black arrow cursor indicates that, based on the selected drawing tool, you should pick a geometric element.

- A cross cursor indicates that, based on the selected drawing tool, you should select a geometric location.

- To delete a geometric element, select it by clicking over the element (**picking the element**) and press the *Delete* key on the keyboard. Several geometric elements may be selected at once by dragging the cursor. Deleting is allowed only if the cursor is in the default state (white arrow).

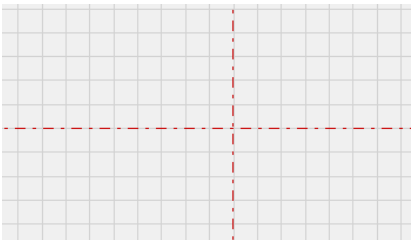
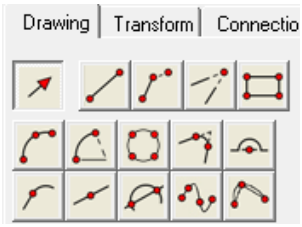


Figure B.2 – Drawing grid (gray lines) and coordinates axis (red lines).

B1.2.3 – Drawing Tools

A set of drawing tools is available to draw line segments and arcs, using several techniques

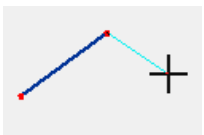


Tool: Draw Line

Draws a line between two locations.

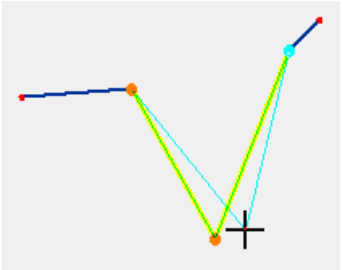
Usage: 1 – specify several locations to draw several consecutive line segments.

Mouse right click to finish.



Tool: Move node

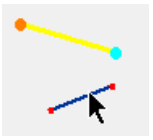
Moves the selected geometric node.



Usage: 1 – pick the desired node; 2 – specify its final location.

Nodes associated to arcs cannot be moved. Mouse right click to cancel at any of the stages.

Tool: Extend up to intersection



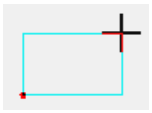
Extends two line segments up to their intersection.

Usage: 1 – pick one line segment; 2 – pick another line segment; 3 – pick, again, the second line segment to confirm.

Mouse right click to cancel at any of the stages.

Tool: Draw rectangle

Draws 4 line segments forming an horizontal rectangle.

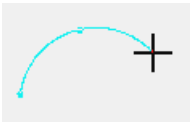


Usage: 1 – select one location for one side of the diagonal; 2 – select a second location for the opposite side of the diagonal.

Mouse right click to cancel at any of the stages.

Tool: Draw arc by 3 points

Draws an arc that passes through 3 locations.

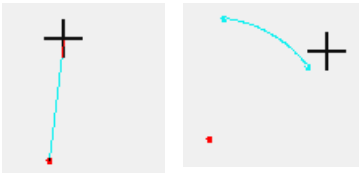


Usage: 1 – select one location for the arc beginning; 2 – select a second location; 3 – select a location for the arc end.

Mouse right click to cancel at any of the stages.

Tool: Draw arc by centre and angle

Draws an arc given its centre, one end and the arc angle.

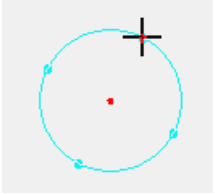


Usage: 1 – select the centre location; 2 – select the arc beginning; 3 – drag the mouse to define the arc angle and left click to draw.

Mouse right click to cancel at any of the stages.

Tool: Draw circle

Draws circle with a given radius. The circle is composed by four arcs.

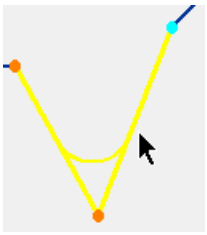


Usage: 1 – select the centre location; 2 – define the radius, dragging the mouse; 3 – click to finish.

Mouse right click to cancel at any of the stages.

Tool: Draw arc by tangent to 2 line segments

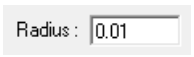
Draws an arc with a given radius, tangent to 2 consecutive line segments. The arc *Radius* must be specified in the corresponding *TextBox*.



Usage: 1 – pick one line segment; 2 – pick another line segment, consecutive to the first one; 3 – pick, again, the second line segment to confirm.

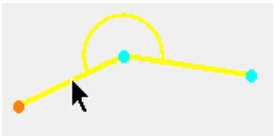
Mouse right click to cancel at any of the stages.

Tool: Draw arc between segments



Draws an arc with a given radius, centred in the intersection of two consecutive line segments. The arc *Radius* must be specified in the corresponding *TextBox*.

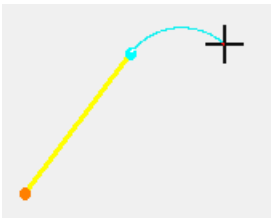
Usage: 1 – pick one line segment; 2 – pick another line segment, consecutive to the first one; 3 – pick, again, the second line segment to confirm.



Mouse right click to cancel at any of the stages. The arc is generated, counter-clockwise from the first to the second line segment. Thus, inverting the picking order generates the complementary arc (cf. at right).

Tool: Draw arc tangent to segment

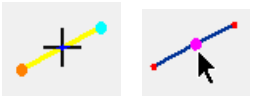
Draws an arc tangent to a segment end.



Usage: 1 – pick a free node at the end of a line segment; 2 – select the end location for the arc.

Mouse right click to cancel at any of the stages.

Tool: Break geometric element



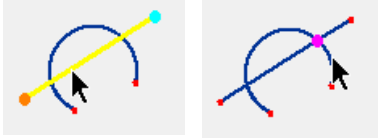
Breaks a geometric element (either a line segment, an arc or a spline), by inserting a node at the selected location.

Usage: 1 – pick a geometric element; 2 – select the location for breaking the element.

Mouse right click to cancel at any of the stages.

Tool: Break at intersection

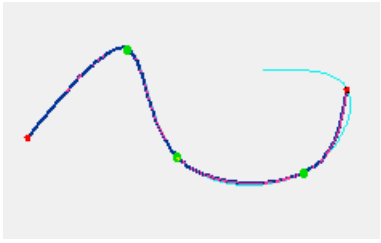
Breaks two geometric elements at their intersection. The geometric elements may be, either, two line segments, two arcs or an arc and a line segment. If the elements intersect at two locations, a second operation must be carried out for the second intersection.



Usage: 1 – pick a geometric element; 2 – pick another geometric element.

Mouse right click to cancel at any of the stages

 Tool: Draw spline



Draws a cubic spline through specified locations (fit points).

Usage: 1 – specify several locations for the fit points.

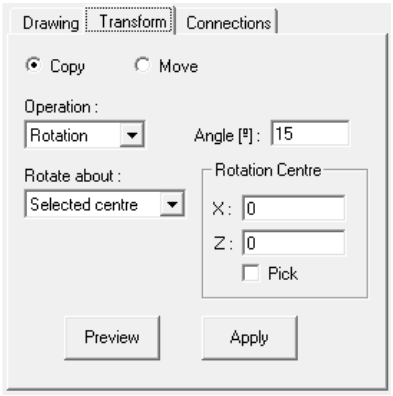
Mouse right click to finish

 Tool: Create spline from end points

Creates a spline taking as input the end point of selected elements.

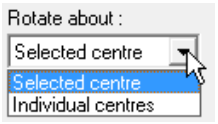
Usage: 1 – select elements; 2 – press the button and confirm to create the spline.

B1.2.4 – Transform Tools

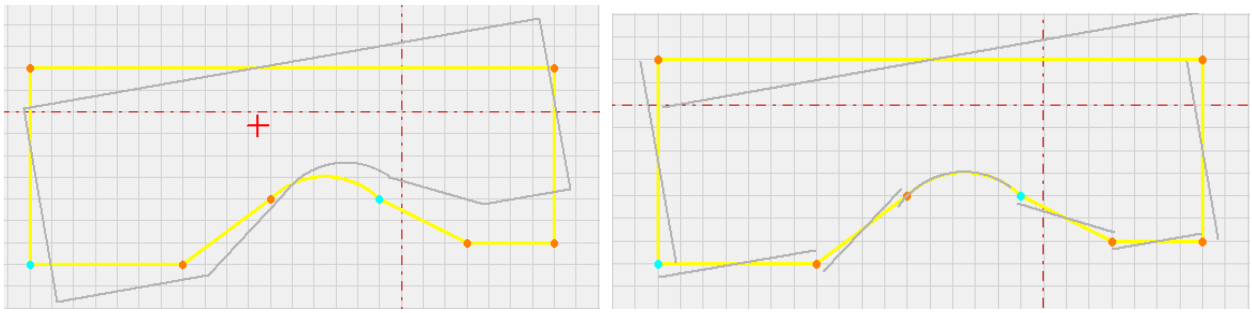


Transform tools allow either moving or creating a copy of existing geometric elements. The operation is applied (*Button Apply*) only to the selected elements. There is the possibility to *Preview* the results (in gray, in the examples of Figure B.3, Figure B.4 and Figure B.5) before carrying out the transformation.

Selectable operations are:



Rotation: a rotation is performed for the specified *Angle*[º] about a *Selected centre* or about the *Individual centres*. If *Selected centre* is chosen, all selected geometric elements are rotated about the same location (cf. Figure B.3(a)). The *Rotation Centre* location may be defined by entering the corresponding coordinates *X* and *Z*, or by picking with the mouse (enabled only if *Pick* is checked). If *Individual Centres* is chosen, all selected elements are rotated about each individual centre (cf. Figure B.3(b)).



(a) (b)

Figure B.3 – Rotation. (a) about a selected centre; (b) about individual centres.

Translation: needed data are the horizontal (*Dx* [*m*]) and vertical (*Dz* [*m*]) displacements (cf. **Figure B.4**).

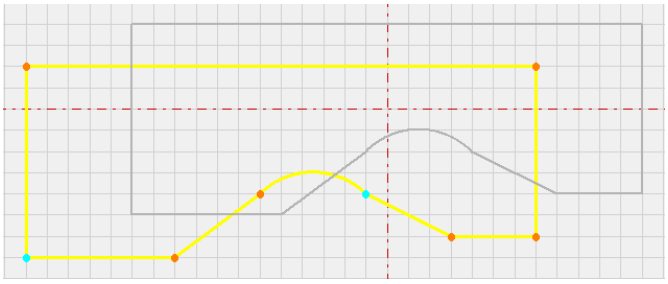
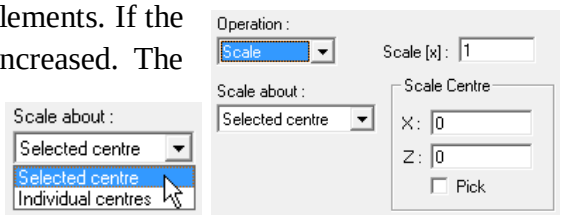
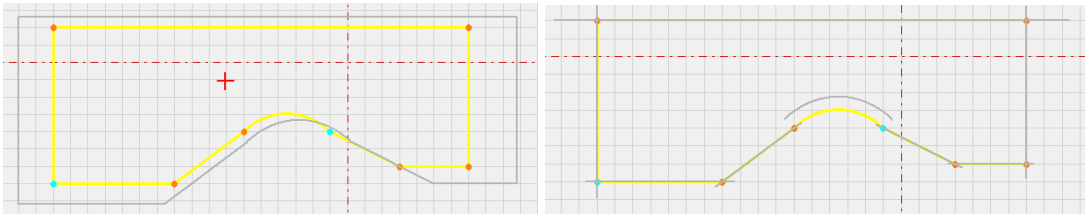


Figure B.4 – Translation.

Scale: this tool applies a *Scale* factor to the selected elements. If the scale factor is larger than *1*, the elements size will be increased. The selected elements remain centred in the *Selected Centre* (cf. Figure B.5(a)) or, alternatively, in the *Individual Centres* (cf. Figure B.5(b)).





(a) (b)

Figure B.5 – Scale. (a) about a selected centre; (b) about individual centres.

OffSet: this tool applies a perpendicular translation to the selected elements in both directions, with the chosen *Offset* distance. This is a very convenient tool for drawing pipes, for instance.

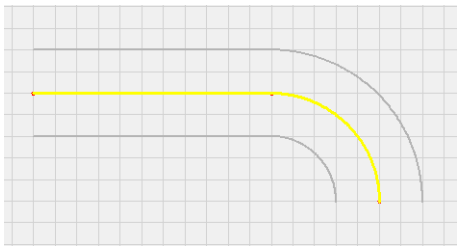
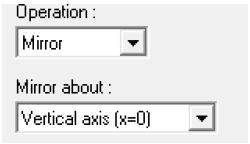


Figure B.6 – Offset.

Mirror: this tool applies a mirror operation to the selected elements, about the selected axis.



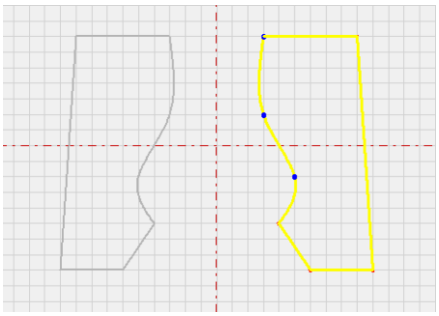
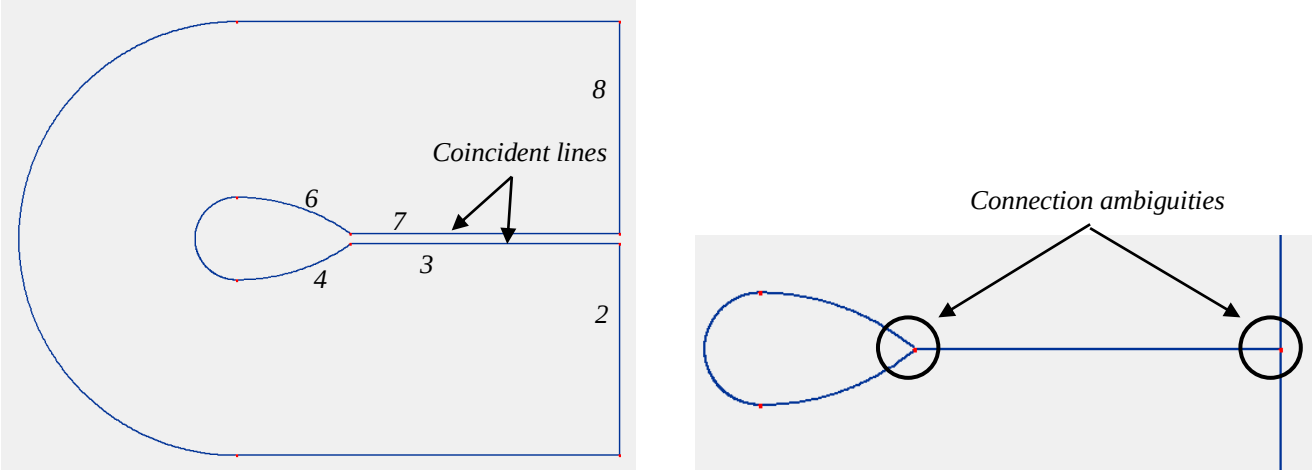


Figure B.7 – Mirror.

B1.2.5 – Connections for U-Type and C-Type meshes

Cf. example files *NACA2412_UMesh.def* and *C-TypeMesh.def*

For calculations around certain types of objects, such as airfoils, a good solution is to use C-Type or U-Type structured meshes. In these cases, the physical domain folds over itself, leading to a coincidence of some geometric elements (cf. Figure B.8(a)). Since there are four geometric elements that share a common punctual location (cf. Figure B.8(b)), the connection ambiguity must be solved by properly specifying the actual connections.



(a) (b)

Figure B.8 – Geometry for a U-Type or C-Type Mesh. (a): In this representation, coincident lines3 and 7 are show moved apart, for a better visualization; (b) Identification of connection ambiguities.

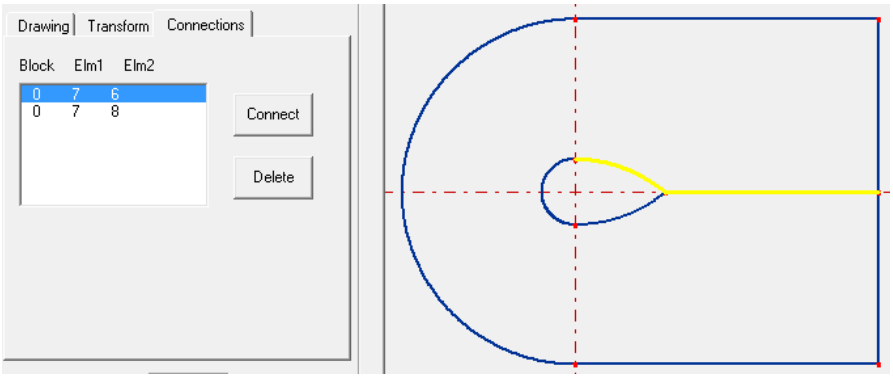


Figure B.9 – Defining connections.

Referring to Figure B.8(a), one possibility would be to specify the connection between elements 6 and 7, and another connection between elements 7 and 8. Another possibility would be to take the pair 4-3 and 3-2, or even 6-7 and 3-2. This is accomplished by selecting the corresponding elements (pressing down the shift key to be able to pick more than one element) and clicking on the button *Connect* (cf. Figure B.9). Every time a new connection is created, the elements constituting the pair are added to the *ListBox*. To delete a connection, select it on the *ListBox* and press *Delete*.

B1.2.6 – Closed- and open-blocks; other general drawing concepts

The calculation domain is externally delimited by the **envelope**. The envelope is a closed boundary (or closed body) constituted by one set of interconnected line segments and/or arcs. Inside the envelope, one or more **blocks** may exist.

There are two types of blocks:

- Closed-blocks: Similarly to the envelope, closed-blocks are defined by a closed set of geometric elements.
- Open-blocks: by default, an open-block constitutes a region pervious to the flow and heat transfer. These entities can be used to create thin plates (cf. section B1.6.10), to provide a better control for mesh characteristics or to impose velocity values inside the fluid domain (cf. section B1.6.6).

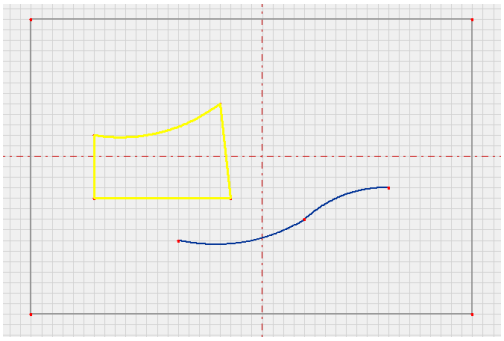
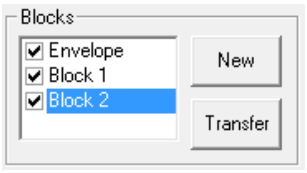


Figure B.10 – Envelope (closed block, in gray), closed inner block (in yellow) and an open-block (in blue).

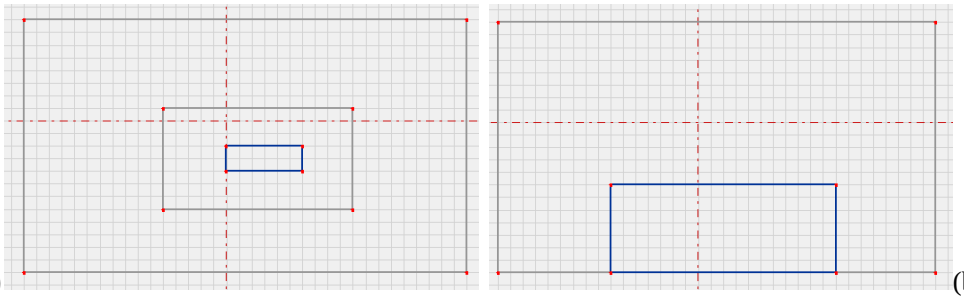
A new block is created simply by pressing the button *New*.



The *ListBox* showing the existing blocks allows you to choose which blocks are visible (checked items). The **active block** (i.e., the block that receives the drawing elements) is selected in blue and the corresponding geometric elements are shown in blue. The geometric elements associated with the remaining blocks are displayed in gray.

The *Transfer* button allows you to transfer the selected elements into the active block.

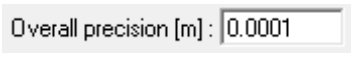
Different blocks cannot intersect but they can partially overlap. Figure B.11(a) shows 3 blocks inside each other. Figure B.11(b) shows 2 blocks that partially overlap. The coincident line segments in this case must share the same ending points. This means that, in the case shown, the bottom boundary of the larger block (the envelope) should be constituted by 3 line segments, so that the middle segment is coincident with the bottom boundary of the internal block. Is this is not done, **EasyCFD** will break the line segments at the necessary location, before changing to the *Meshing* menu (cf. section B1.2.9).



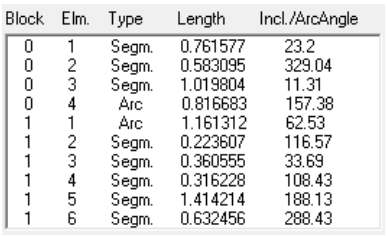
(a) (b)

Figure B.11 – Different arrangements for blocks.

The *Overall precision* represents the minimum distance below which two locations are considered as coincident.

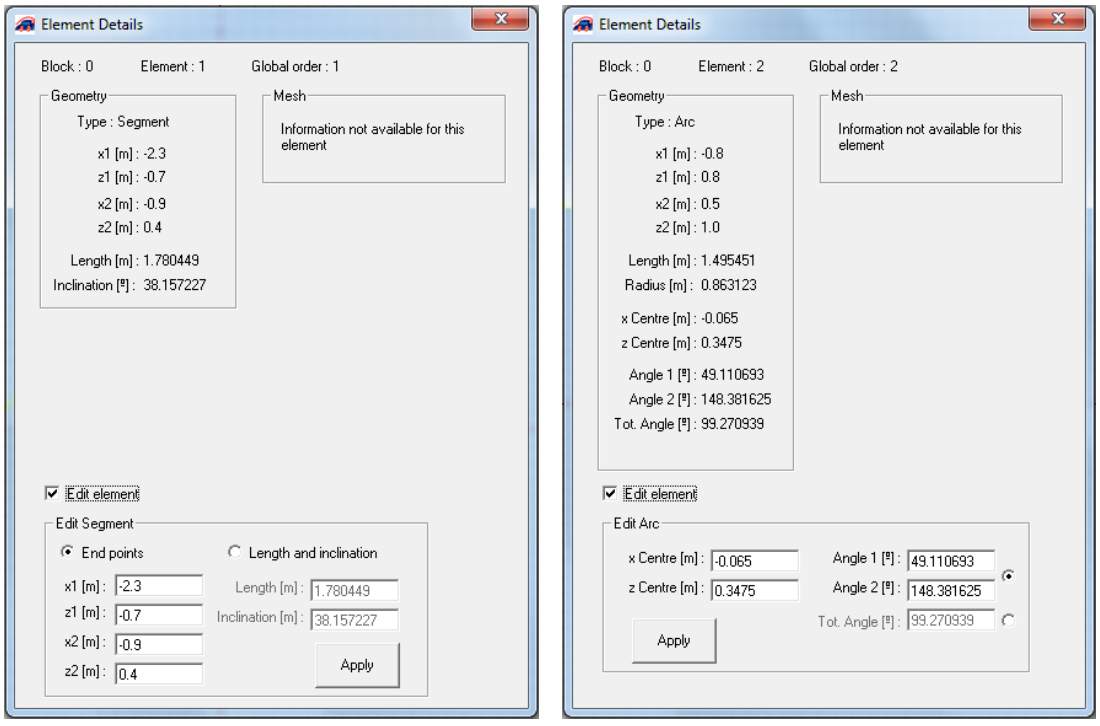


The *ListBox* shown at right displays all the geometric elements in the project and some of their characteristics. Double-clicking one item in this *ListBox* opens a *Dialog-Box* with detailed information about geometry and mesh of the selected element, as shown in Figure B.12 (mesh information is displayed only if meshing data is already available). The same functionality is achieved if you double-click over the element, on the drawing-space. By checking *Edit element*, the geometric characteristics of line segments and arcs can be modified.



Block	Elm.	Type	Length	Incl./ArcAngle
0	1	Segm.	0.761577	23.2
0	2	Segm.	0.583095	329.04
0	3	Segm.	1.019804	11.31
0	4	Arc	0.816683	157.38
1	1	Arc	1.161312	62.53
1	2	Segm.	0.223607	116.57
1	3	Segm.	0.360555	33.69
1	4	Segm.	0.316228	108.43
1	5	Segm.	1.414214	188.13
1	6	Segm.	0.632456	288.43

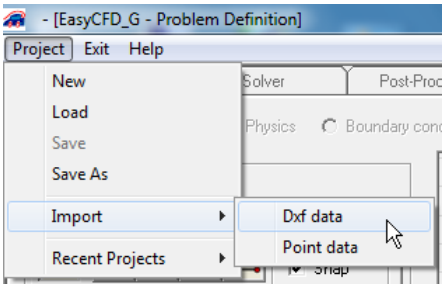
Note: the order of the elements in this listing reflects the sequence of their generation. When the *Mesh* mode is open, the elements are reordered according to the block to which they belong and to their connection sequence.



(a) (b)

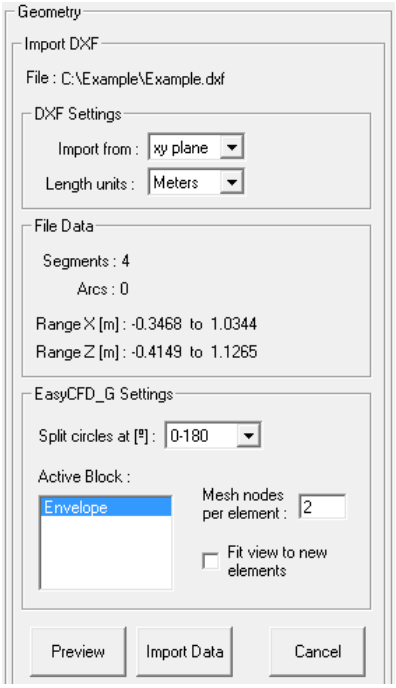
Figure B.12 – Detailed information for a geometric element (no mesh information is available) and editing parameters. (a) data for line segment; (b) data for arc.

B1.2.7 – Importing DXF data



EasyCFD may import geometric data from a DXF file (cf. menu at the right). Allowed elements for import are line segments, arcs and circles.

The menu show at right is displayed after specifying the file to import data from.



DXF Settings:

If the DXF file has 3D information, you must specify which coordinates are to be imported (*Import from*): *xy plane* or *xz plane*. The *Length units* in the DFX file must also be supplied.

File Data:

This provides information about the data: total number for each element type and *x* and *z* range.

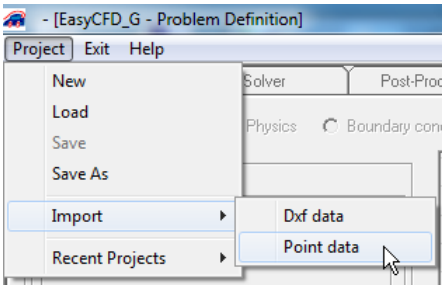
EasyCFD Settings:

Each circle must be split into 2 arcs. The angle for the split location must be specified.

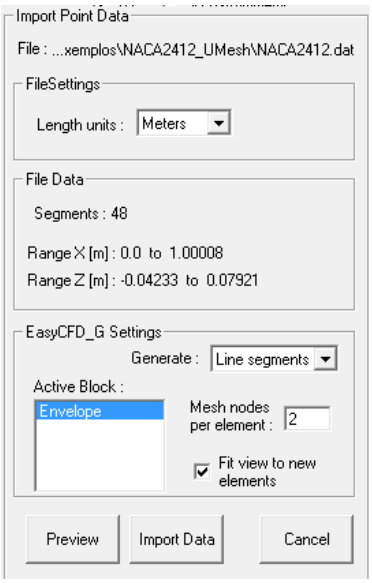
Mesh nodes per element represents the initial information for the meshing process.

If checked, the *Fit view to new elements* enforces the view to fit into the new elements after importing.

B1.2.8 – Importing data from Point Data File



EasyCFD may import geometric data from a Point Data File (cf. Menu at the right). This file is a text file (ASCII file; cf. Figure B.13) containing the coordinates of 2 or more spatial locations. Each line represents one point, with the *x* and *z* coordinate in the first and in the second column, respectively. Each pair of points will be connected by a line segment after importing into **EasyCFD**.



Settings and information in the Import menu are similar to the corresponding ones in the DXF file import described previously.

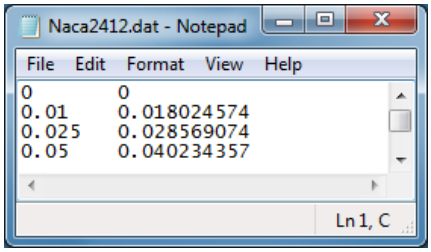
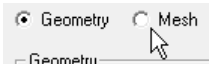


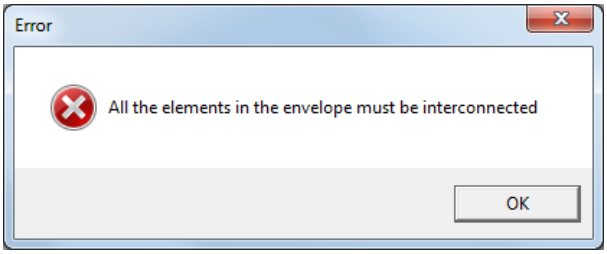
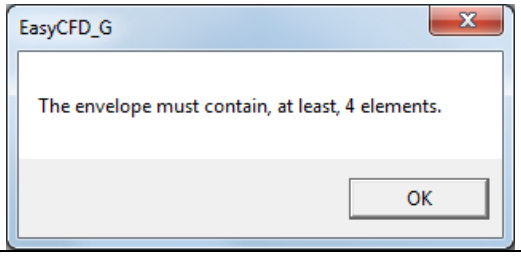
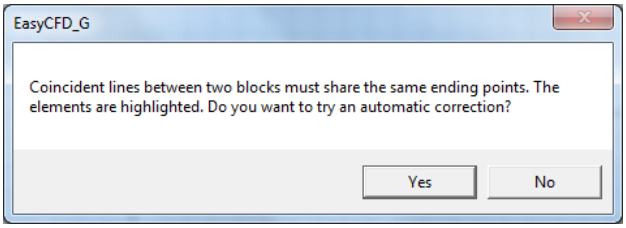
Figure B.13 – Point data file. This example would produce 3 line segments.

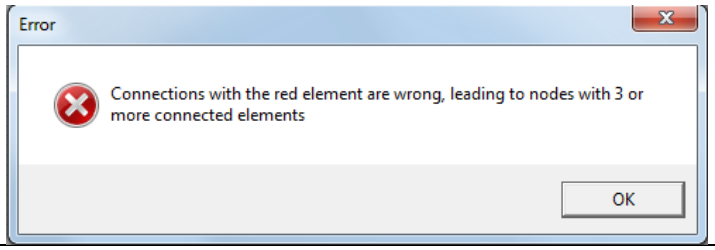
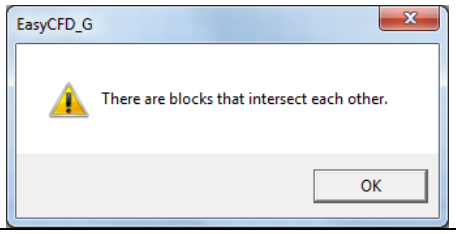
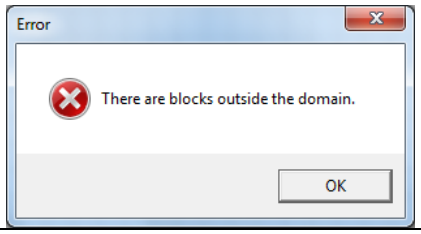
B1.2.9 – Final auto-verification before meshing



When the geometry is ready, the next step is to generate the mesh. When choosing the *Mesh* mode (by pressing *Mesh* at the option-button shown at right) a general checkup is performed to the geometry, to detect for errors. Additionally, the geometric elements are reordered. If any error is detected, **EasyCFD** jumps back the *Geometry* mode. Next table lists some of the errors that may be detected.

Table 1 – Error Messages

#	Example of Error Message	Description
1		The envelope must be formed by a closed set of geometric elements. If you define a new set and don´t assign it to a new block, some elements appear to be disconnected (cf. Figure B.14). A similar message is displayed if, within a block, one element is not connected to any other.
2		The message is self explanatory. Even if the envelope should represent, for instance, a closed circumference, it must be split it into four arcs (cf. Figure B.15).
3		If two blocks share one edge or part of one edge, the coincidence between elements must be total (*cf.* Figure B.16). If you didn´t break the necessary elements, **EasyCFD** can do this for you at this stage.

4		Three or more geometric elements within the same block share a common edge (cf. Figure B.17).
5		Some blocks intersect each other (cf. Figure B.18).
6		Some blocks are outside the envelope (*cf.* Figure B.19).

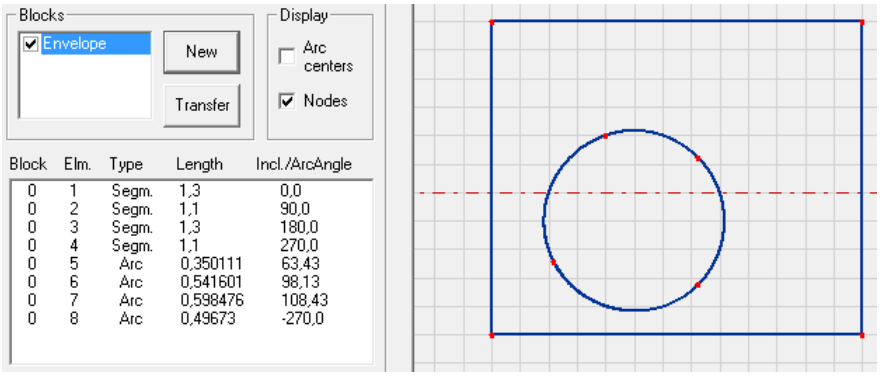


Figure B.14 – Example of situation for Error Message #1 (cf. Table 1). There are two different closed bodies sharing the same block (block 0: the envelope). Solution: Transfer all the circle elements into Block 1.

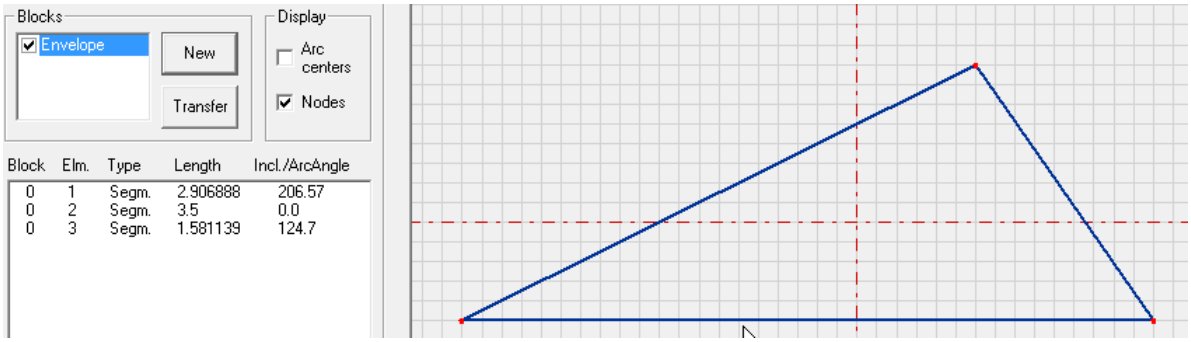
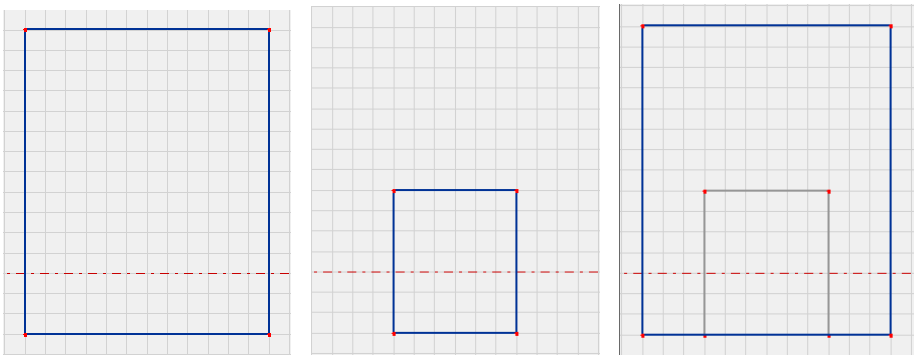


Figure B.15 – Example of situation for Error Message #2 (cf. Table 1). The envelope is composed by only 3 elements. Solution: break at least one of the line segments of the triangle.



(a) The envelope (b) Block 1 (c) The whole geometry

Figure B.16 – Example of situation for Error Message #3 (cf. Table 1). The lower edge of the envelope must be broken at two points to obtain a total overlapping between the coincident elements.

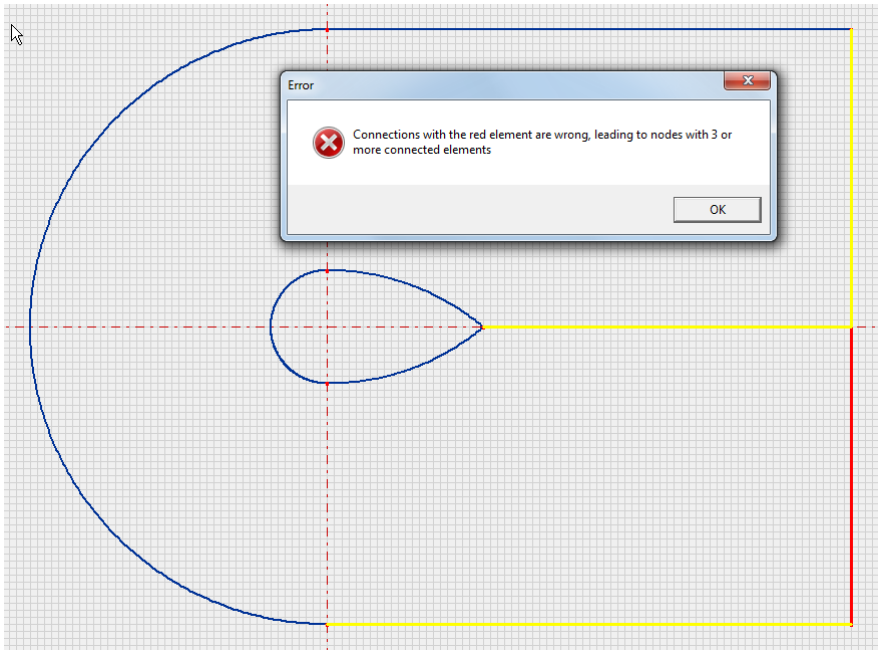


Figure B.17 – Example of situation for Error Message #4 (cf. Table 1). Three or more line segments share a common edge. In this case, this should be corrected by specifying the proper connections; cf. section B1.2.5 .

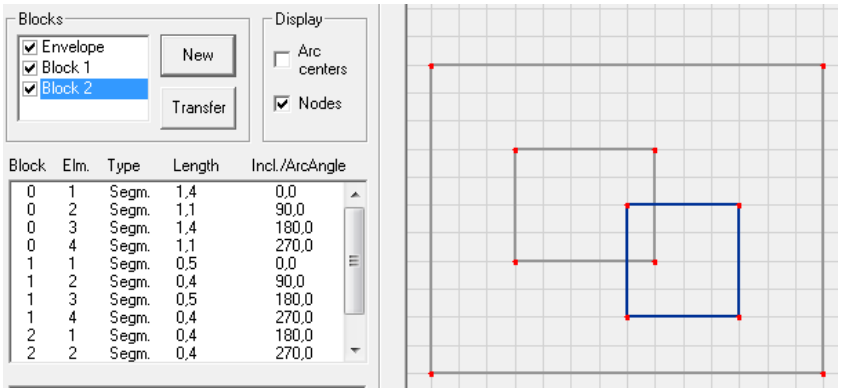


Figure B.18 – Example of situation for Error Message #5 (cf. Table 1). Blocks 1 and 2 intersect each other.

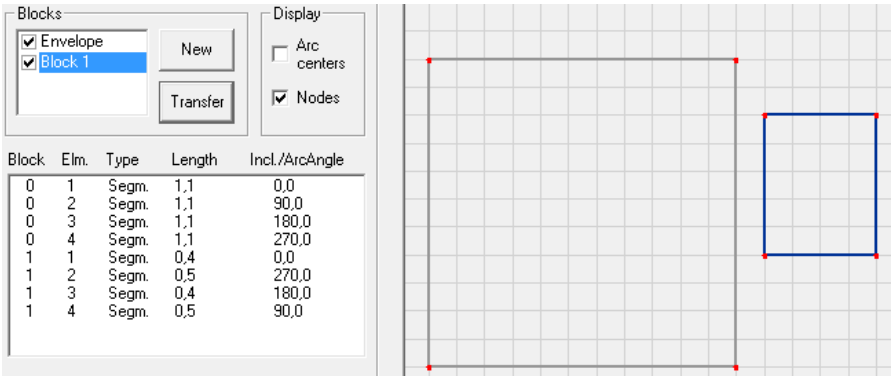
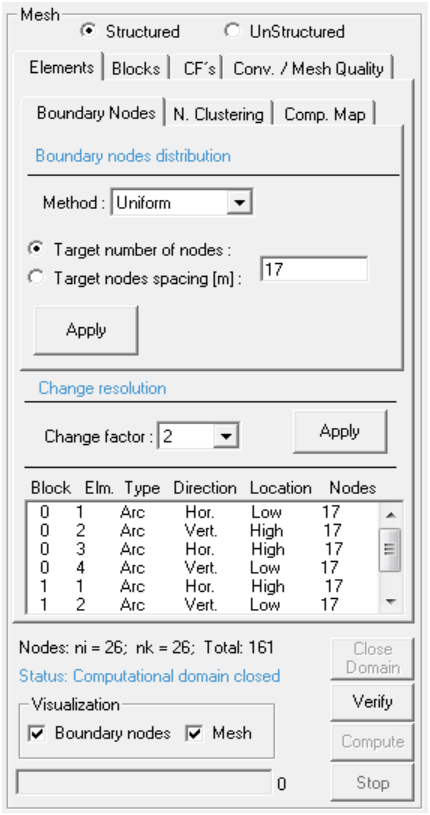


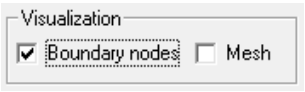
Figure B.19 – Example of situation for Error Message #6 (cf. Table 1). Block 1 is outside the envelope.

B1.3 – Mesh Generation: Structured type

By pressing the *Option-Button Mesh*, **EasyCFD** enters the auto-verification described in the precedent section. If no errors are found, an initial equally spaced mesh is assigned to the boundary elements. The mesh spacing is automatically adjusted so as to achieve approximately 160 mesh nodes over the envelope perimeter. This is a low resolution mesh with the only purpose to initialize the mesh generation process.



As an example, let´s consider the geometry displayed in Figure B.20. You may have noticed that, when selecting an element, each of its two geometric nodes (element sides or edges) are coloured differently as a means of identifying each side: orange for side 1 and light blue for side 2.



When selecting *Boundary nodes* or *Mesh*, in the *Visualisation* frame (cf. at right), the corresponding visualisations are obtained, a depicted in Figure B.21 and Figure B.22. If the mesh is not yet computed (like in the example of Figure B.22), the displayed mesh corresponds to the initial solution, where the inner mesh nodes distribution is obtained by linear interpolation from the boundaries.

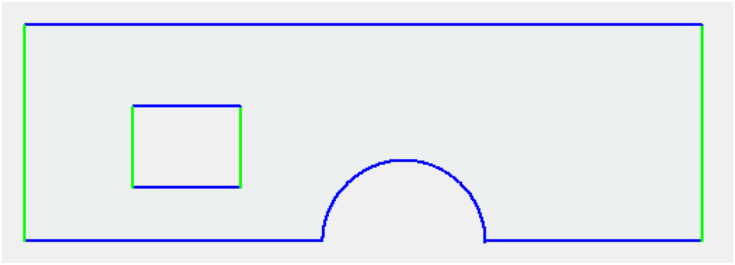


Figure B.20 – Geometry for example.

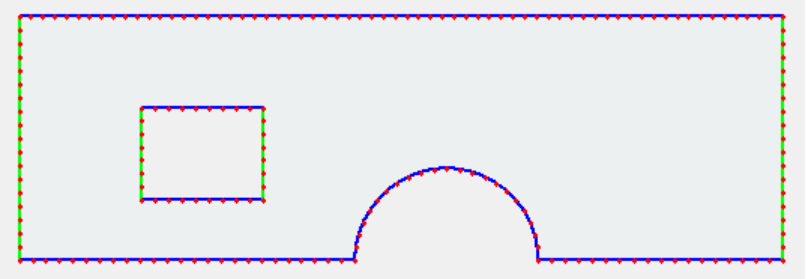


Figure B.21 – Display of boundary mesh nodes.

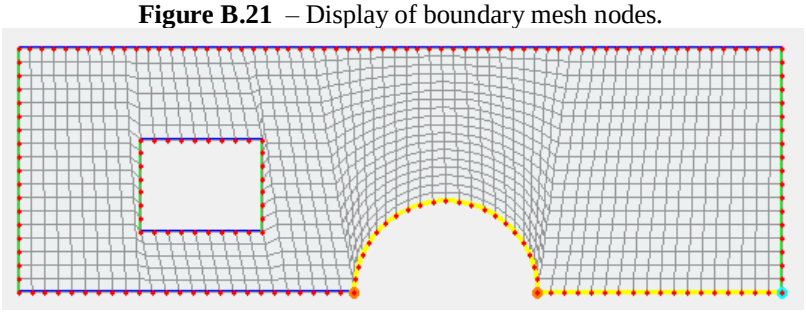
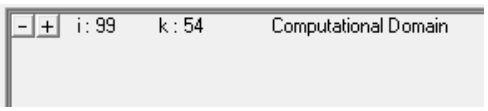


Figure B.22 – Geometry of initial mesh (linear interpolation from boundaries).

B1.3.1 – Computational Domain; Concept of Computational Direction

As described in section A2, the computational domain represents the indexation of the bidimensional arrays that store the variables. It may be regarded as a representation of mesh nodes taking a unitary spacing and perfect perpendicularity between the mesh lines. The physical domain is obtained by properly deforming the computational domain (and vice-versa). The location of each mesh node in the computational domain is given by their computational coordinates (ξ, ζ), which represent the (i, k) indexation of the storing arrays in the computer implementation. The computational domain is displayed at the top left of the physical domain representation, as shown in Figure B.23.

The size of the window representing the computational domain can be changed by pressing the buttons "+" and "-", as show in the right. The labels *i:* and *k:* display the indexation at the cursor location.



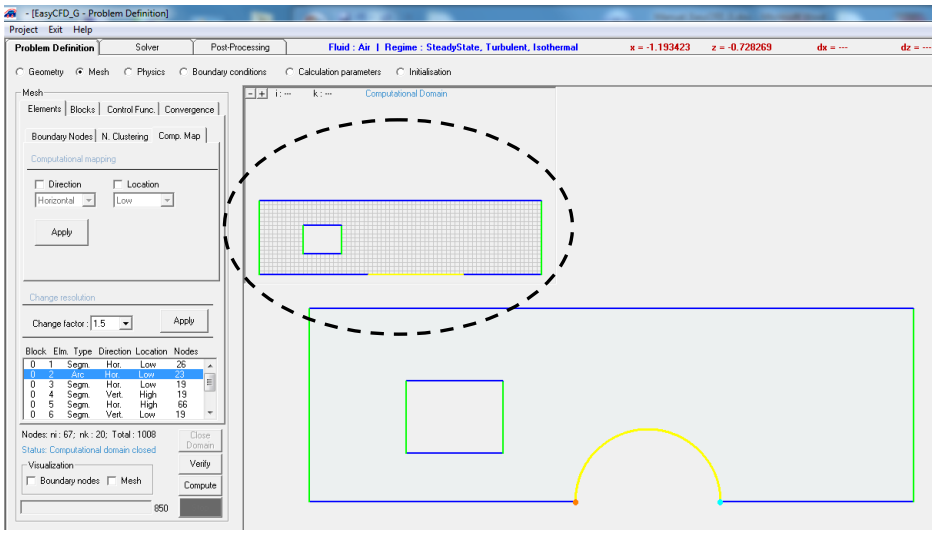


Figure B.23 – User interface, showing the computational domain representation.

In the computational domain, the boundaries may be horizontal (corresponding to ξ lines, with constant *k* index), identified through the blue colour, both in the computational and in the physical representation, or vertical (ζ lines), identified through the green colour, both in the computational and in the physical representation.

You should note that there is not any direct relation between the horizontality (or verticality) in the computational and in the physical domain. Figure B.24 highlights some elements of a C-shaped geometry, showing the relation between the computational and the physical domain (in fact, this is not the only solution for this problem; cf. Section B1.3.4).

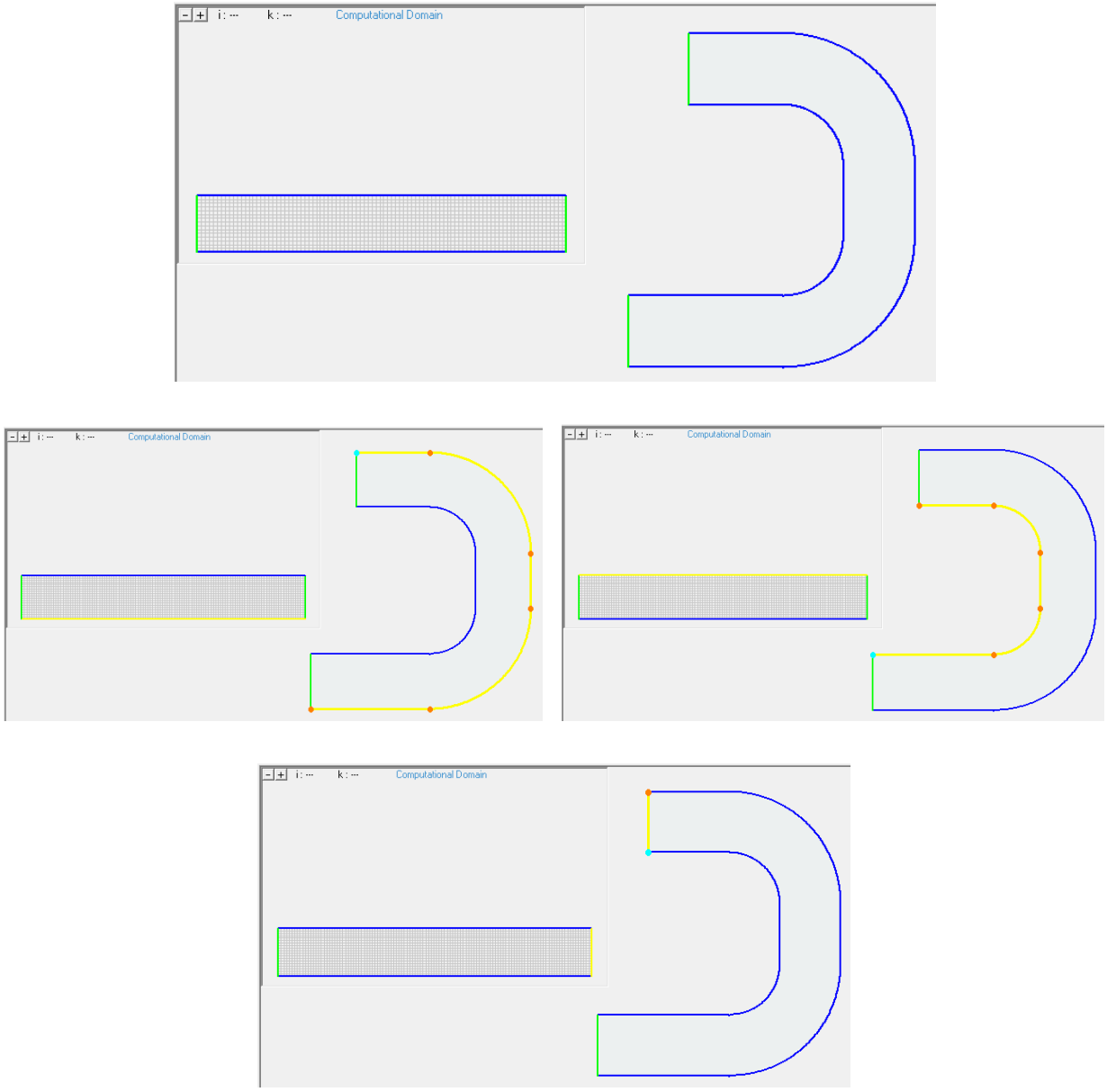
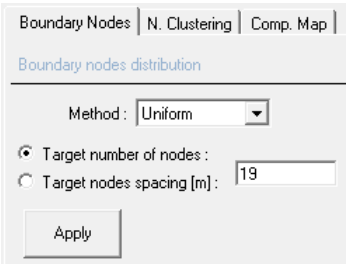


Figure B.24 – Example of correspondence between the computational and physical representation for several elements in a C-shaped geometry.

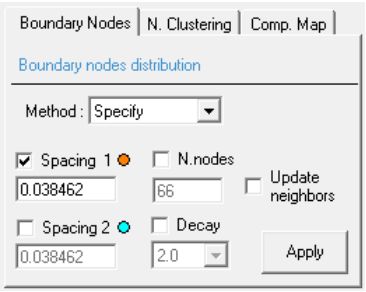
B1.3.2 – Mesh distribution on the boundaries (Elements - Boundary Nodes tab)

The 1D mesh distribution on the boundaries is computed with the algorithm described in section A5.1. Controls for 1D mesh generation are available under the *Boundary Nodes* tab, as shown at the right.



When choosing *Uniform* for the *Method*, an equally spaced mesh distribution is adopted for the selected elements. In this case, you may either impose the number of nodes or the nodes spacing.

When choosing *Specify* for the *Method*, you may impose the spacing at each of the elements ends (side 1 or side 2, which are identified with different colours). The number of nodes (*N. nodes*) and *Decay* (exponent *c*, in equation (5.2)) can also be chosen in this part of the interface (refer to section A5.1 for a description of the 1D mesh generation method). Only the values with a checked mark in the associated *CheckBox* are applied (button *Apply*) to the selected elements. By checking *Update neighbors*, the selected spacing (*Spacing1*, *Spacing2* or both) are transmitted to the neighbor elements, provided they possess the same computational direction (cf. Figure B.25). This is a useful tool to avoid sudden changes in the mesh spacing.



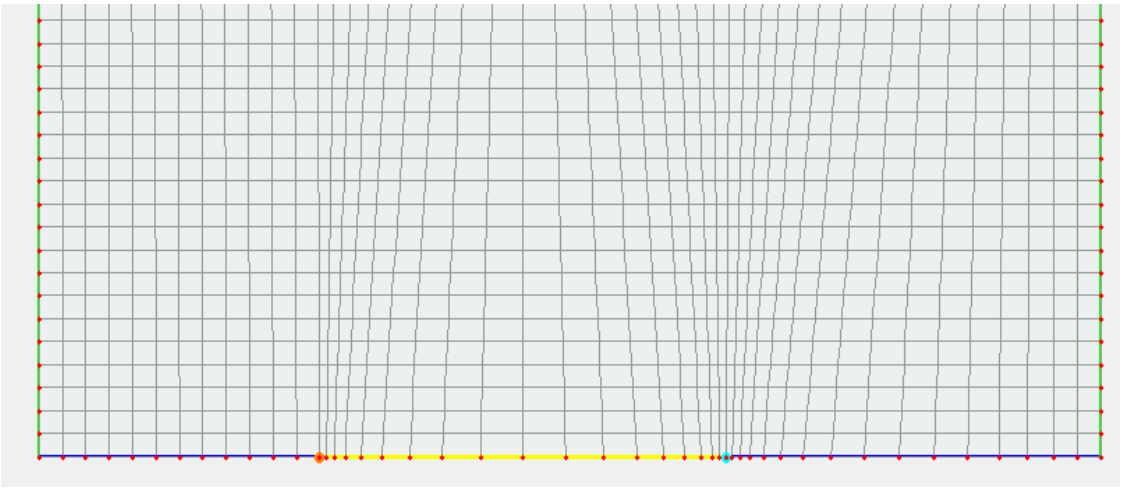
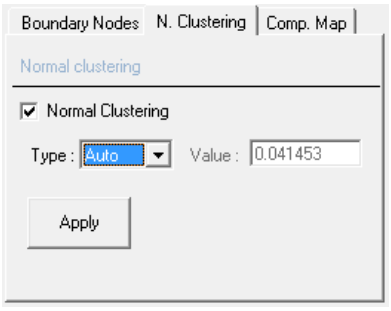


Figure B.25 – Transmission of mesh spacing to the neighbour elements (in this case, yellow element right edge spacing was transmitted to the left edge of the blue element on the right).

B1.3.3 – Nodes clustering (Elements – N. Clustering tab)

The clustering distance represents the normal distance of the mesh nodes to the adjacent boundary (Δs parameter in equation (5.10)). The *NormalClustering CheckButton* allows you to activate or deactivate the Steger and Sorenson control functions for clustering and orthogonality at the boundaries (cf. Section A5.2.1) for each of the selected geometric elements (*i.e.* for each of the selected boundaries).



When selecting *Auto* for *Type*, the clustering distance will be

considered as a weighted average between the boundary nodes spacing of the two perpendicular neighbour elements (note: this perpendicularity concept referrers to the computational domain; these elements will be referred to as **computationally perpendicular neighbour elements**), or taking the imposed mesh clustering of the closest element with the same computational direction. To clarify, let´s examine Figure B.26. The yellow element has an imposed mesh clustering. Its neighbour elements, in blue, show a clustering that is a weighted average between the yellow element clustering and each computationally perpendicular element (vertical boundaries).

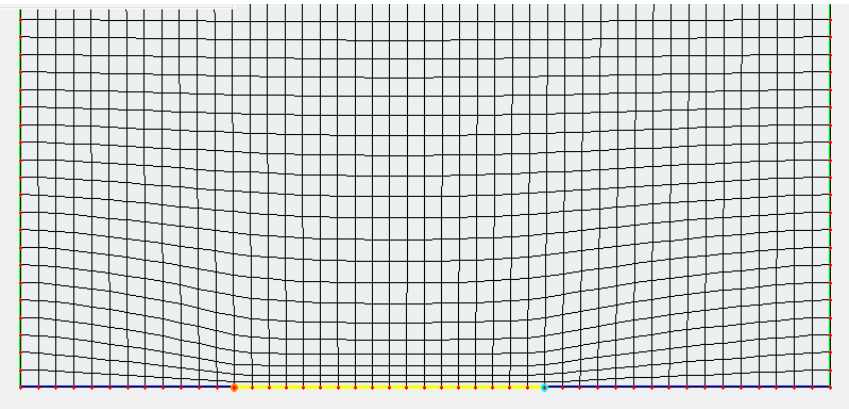


Figure B.26 – Example for the clustering computation.

As a second example, consider the case depicted in Figure B.27. The mesh clustering at the lower boundary (selected elements displayed in yellow) is computed as a weighted average between the mesh spacing at lower edge of each of the two vertical boundaries (Δs_1 and Δs_2).

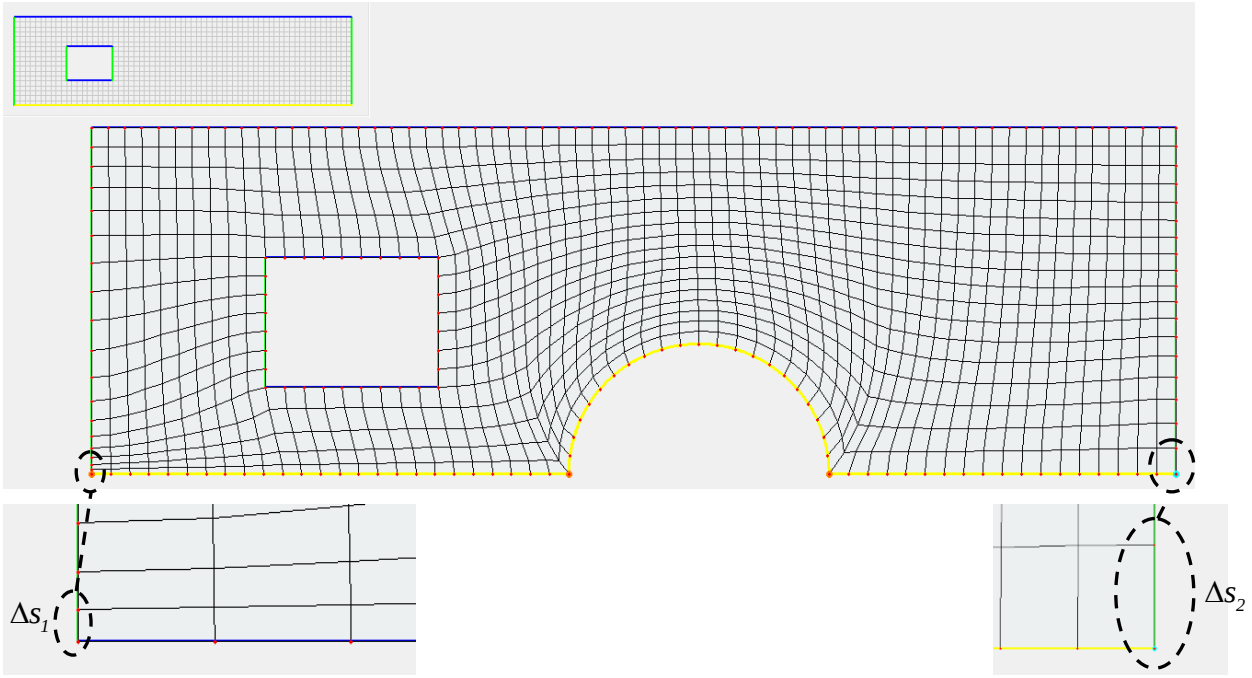


Figure B.27 – Distances Δs_1 and Δs_2 for computational of the automatic clustering distance.

Note: both Δs_1 and Δs_2 are taken as the values obtained from the projection along the normal to the boundary. Figure B.28 exemplifies this concept.

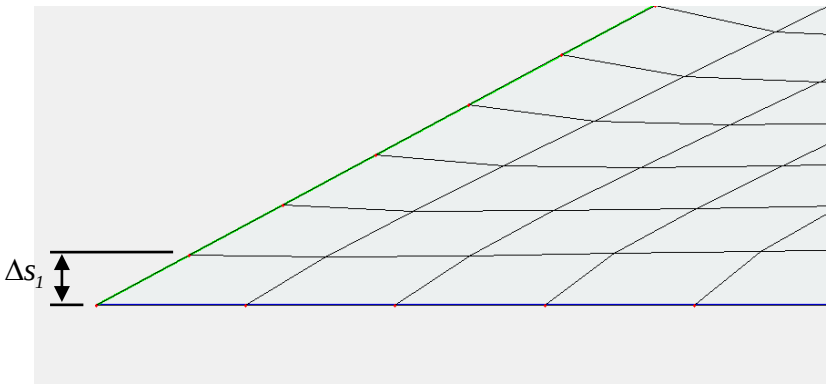


Figure B.28 – Definition of Δs_1 distance.

The *Auto Type* also works if the perpendicular edges are oriented in the opposite direction, as explained next.

Consider the case for the small rectangular "hole" in the domain shown in Figure B.27. If the clustering distance needs to be reduced using the *Auto Type*, one must decrease the mesh spacing at the edges of the computationally perpendicular line segments, as exemplified in Figure B.29.

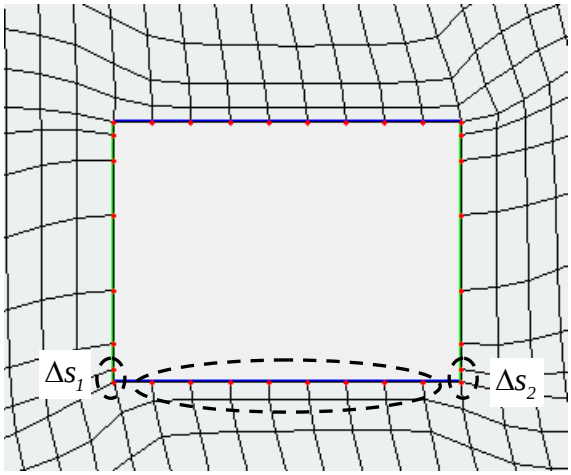
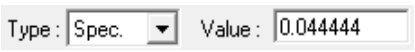


Figure B.29 – Automatic clustering. The Δs_1 and Δs_2 values will reflect on the mesh clustering of the central region of the lower edge.

When selecting *Spec.* for *Type*, the specified clustering distance *Value* is considered. In situations like the one exemplified above (Figure B.29), this allows imposing the mesh clustering value without changing the mesh spacing on the computationally perpendicular neighbour elements (cf. Figure B.30).



However, in the opposite situation. *i.e.*, when the computationally perpendicular neighbour elements are in the inner domain region (like the situation depicted in Figure B.27), mesh spacings Δs_1 and/or Δs_2 may conflict with the imposed clustering value, potentially causing divergence of the iterative calculation when generating the mesh. Figure B.31 shows an example, where the mesh clustering imposed at the lower boundary (in yellow) is 1/6 of the mesh spacing of one of the computationally perpendicular neighbour elements. In this case, a warning message is displaying, alerting for this situation (cf. Figure B.32) and the conflicting elements are highlighted in red and magenta.

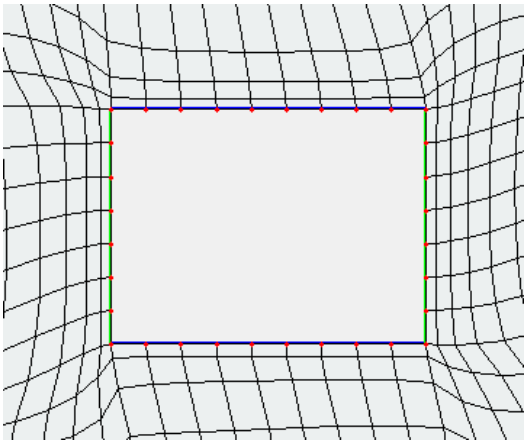


Figure B.30 – Example of application of a Specified clustering.

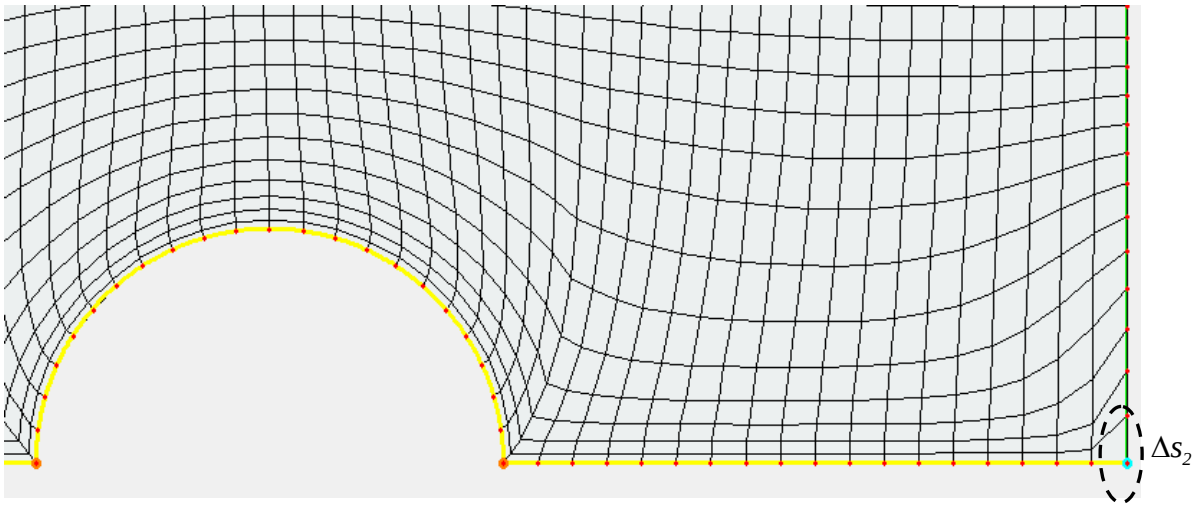


Figure B.31 – Situation where an imposed mesh clustering is conflicting with the mesh spacing of one of the computationally perpendicular neighbour elements.

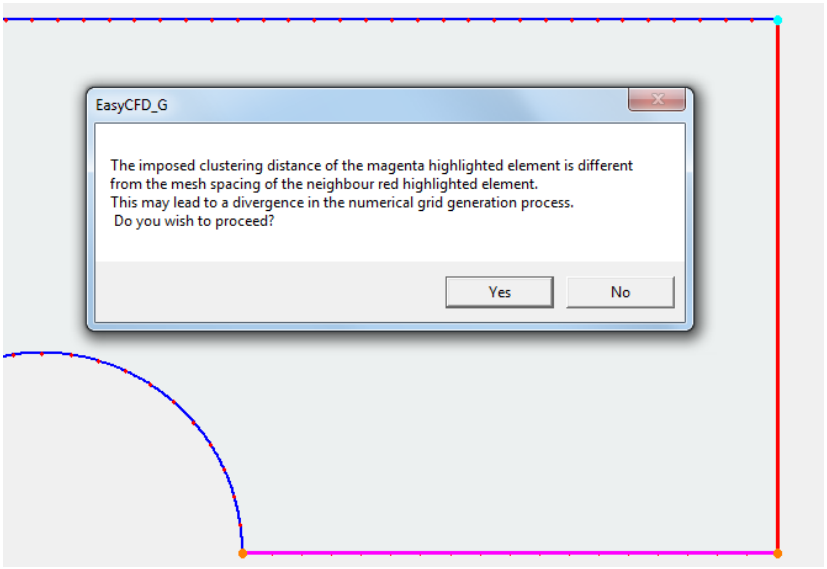
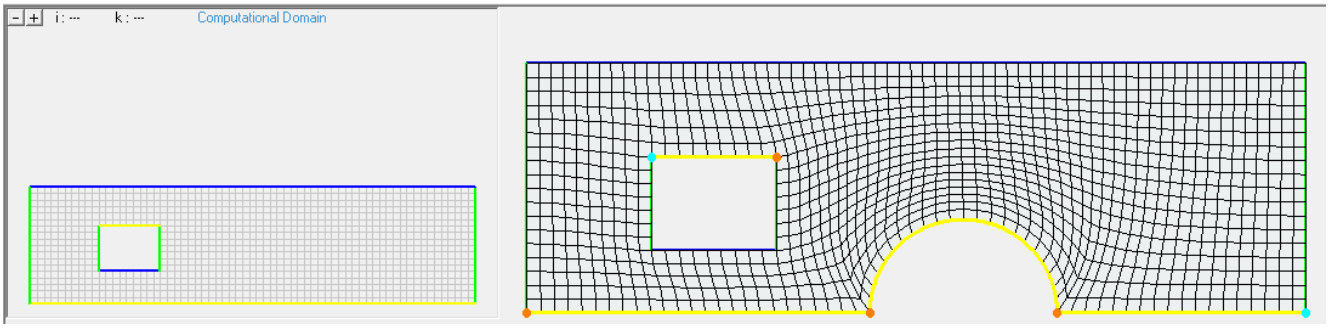


Figure B.32 – Situation where an imposed mesh clustering is conflicting with the mesh spacing of one of the computationally perpendicular neighbour elements.

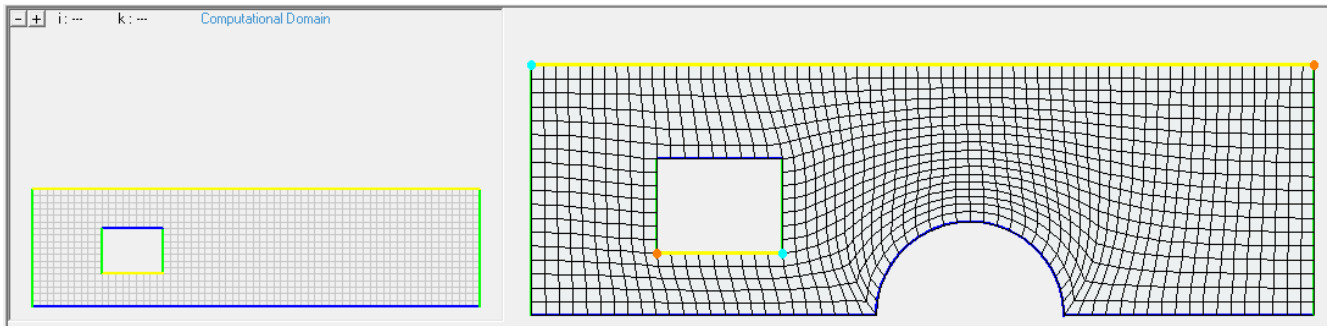
B1.3.4 – More on the Computational Domain. The Comp. Map Tab

Each geometric element has an associated **Computational Direction** (**horizontal** or **vertical**, as described in section B1.3.1) and an associated **Computational Location**, which may be **High** or **Low**. An element has a low location if, in the computational domain, the direction perpendicular to the element encounters higher values of the computational coordinate. The reverse situation occurs for elements with a high location. Figure B.33 up to Figure B.36 exemplify all combinations of computational directions and computational locations. Once again, note that these concepts are defined in the Computational Domain (not in the Physical Domain).



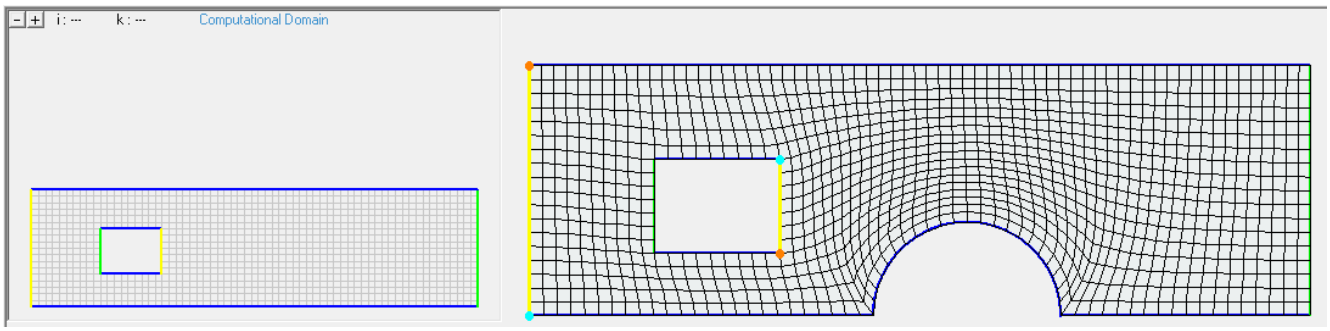
Computational Domain *Physical Domain*

Figure B.33 – Highlighted elements are: Computational Direction: **Horizontal**; Computational Location: **Low**.



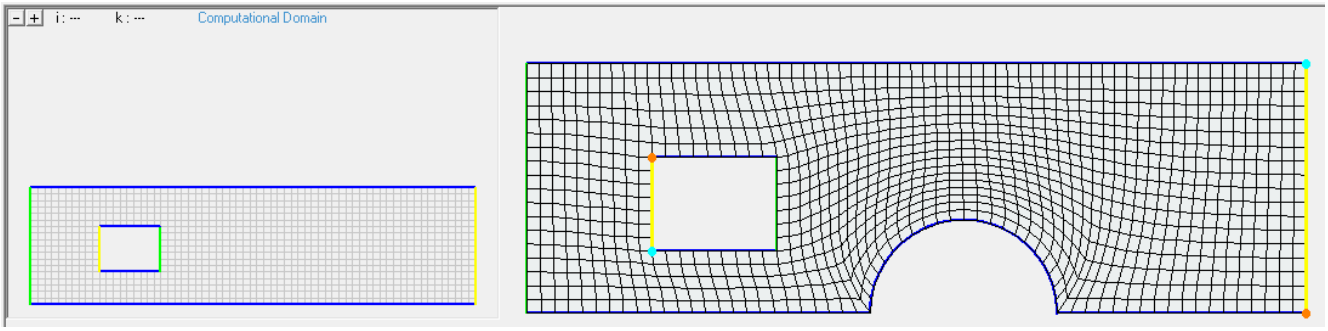
Computational Domain *Physical Domain*

Figure B.34 – Highlighted elements are: Computational Direction: **Horizontal**; Computational Location: **High**.



Computational Domain *Physical Domain*

Figure B.35 – Highlighted elements are: Computational Direction: **Vertical**; Computational Location: **Low**.

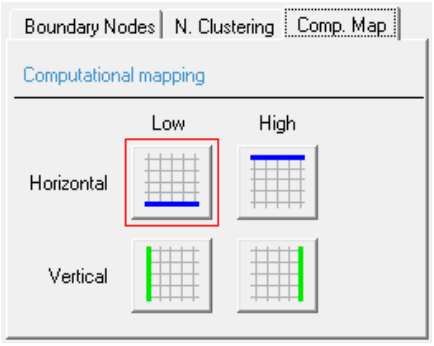


Computational Domain *Physical Domain*

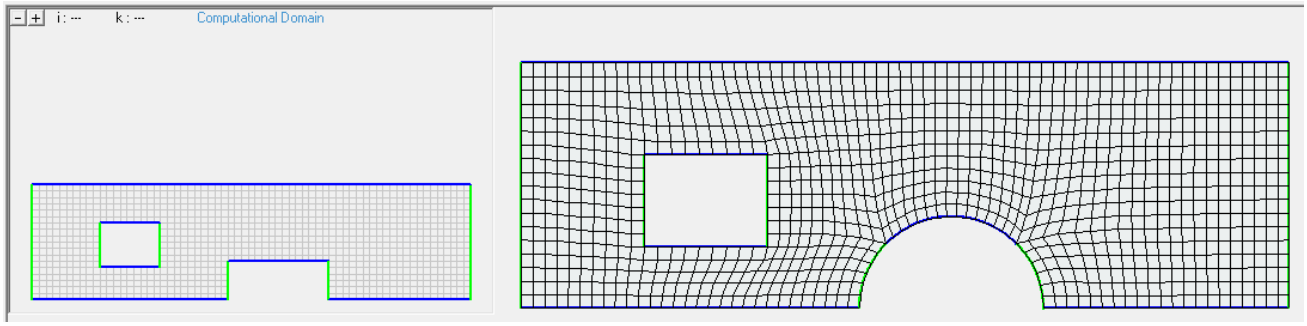
Figure B.36 – Highlighted elements are: Computational Direction: **Vertical**; Computational Location: **High**.

The elements computational characteristics (Direction and Location) are automatically assigned for new elements, when starting the Mesh module. As first approach, the computational direction is assigned based on the closest physical direction (horizontal or vertical). For arcs, the physical direction is taken as the direction of the line segment connecting the arc edges. In many cases, this initial attribution must be corrected manually using the *Comp. Math. Tab*.

The *Comp. Map Tab* allows you to change the computational characteristics (direction and location) of the selected elements.



The examples shown previously exhibit the default settings, which were adequate for this problem. Actually, the semi-circle part at the bottom boundary could have been considered differently. An alternative solution would be to split the semi-circle into three parts, considering two parts as a vertical boundary, and one part (the middle one) as horizontal (cf. Figure B.37).

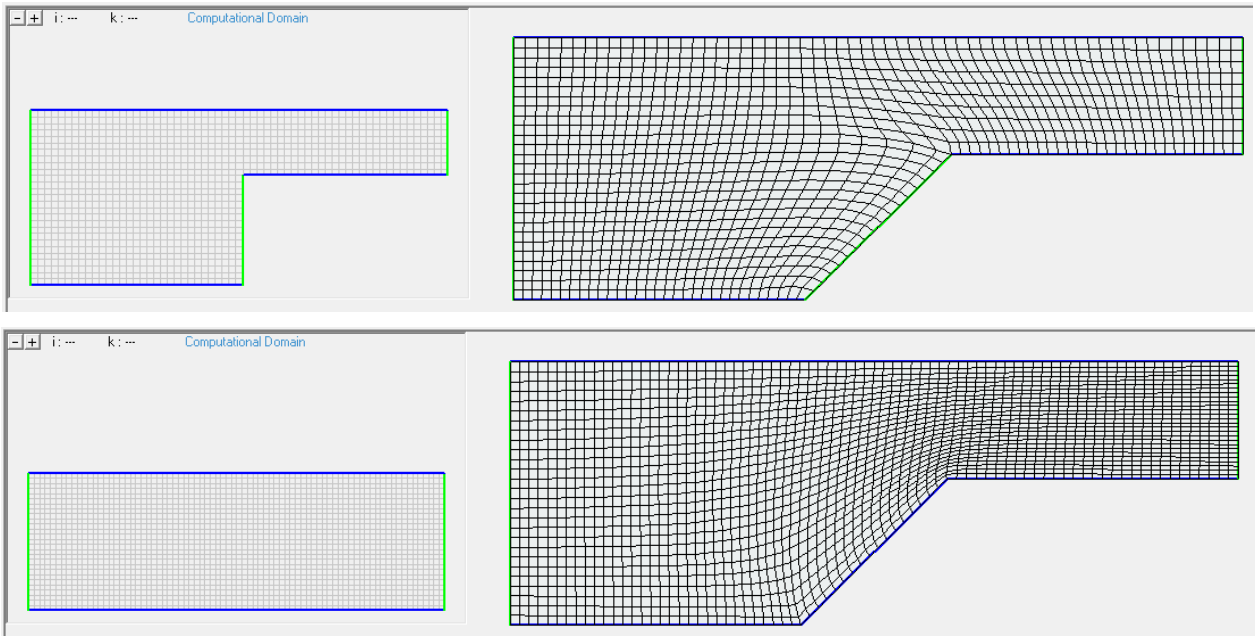


Computational Domain *Physical Domain*

Figure B.37 – Alternative solution for meshing a half-cylinder.

Figure B.38 shows another example of two possible solutions for the same problem.

As previously referred, the default attribution for computational directions and locations is not always satisfactory and, sometimes, errors may even arise. Consider Figure B.39. In this situation, all geometric elements are identified as vertical (the start and end points of the arcs are aligned vertically), so, as initial attribution, all elements will be computationally vertical. This will raise an error message, since, at least, two computationally horizontal elements must exist. In this case, the correction should be done manually in the *Comp. Map Tab*.



Computational Domain *Physical Domain*

Figure B.38 – Two solutions for the same problem.

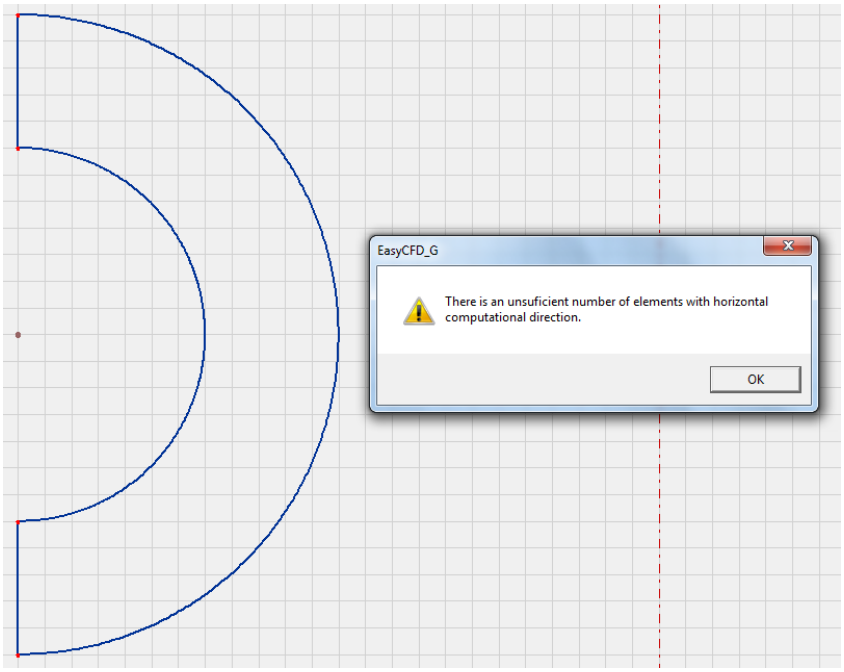


Figure B.39 – Two solutions for the same problem.

B1.3.5 – The concept of Closed Computational Domain

For simplicity, consider a rectangular computational domain, such as the one depicted in Figure B.38. If one increases the number of mesh nodes in, say, the lower boundary, the upper boundary cannot match the indexation of the lower boundary. In its graphical representation, the computational domain doesn´t close (cf. Figure B.40). A similar situation may occur when changing the computational direction and/or position of some geometric elements.

For the computational domain to be closed, there must be a perfect much between the number of mesh nodes in the horizontal low and horizontal high boundaries and similarly for the vertical boundaries.

Closing an open computational domain is achieved by properly adjusting the number of mesh nodes in some boundaries. This process can be done either manually or automatically, by pressing the *Close Domain* button. You can prevent the number of mesh nodes of some elements to be changed during this process, by selecting them before pressing *Close Domain*.

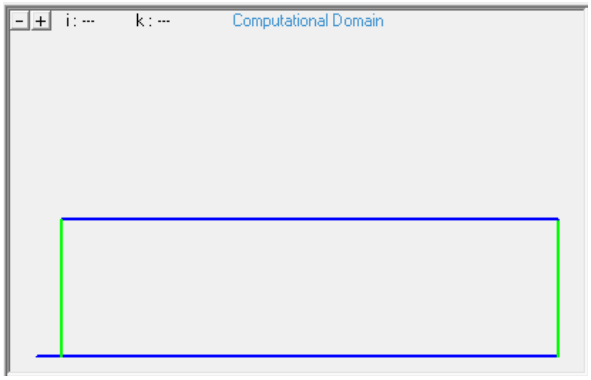
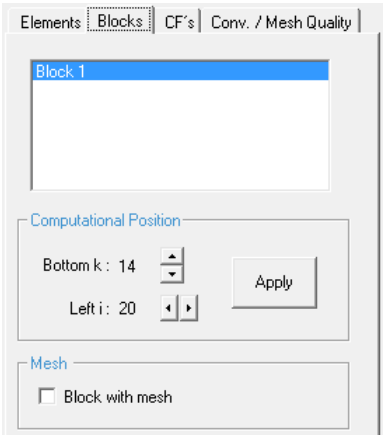


Figure B.40 – Open computational domain. The number of mesh nodes in the low horizontal boundary is higher than in the high horizontal boundary.

B1.3.6 – Blocks Computational Position and Blocks Meshing: The Blocks Tab

Adjusting the blocks computational position



The computational position of existing blocks can greatly affect the mesh skewness. The blocks computational position relative to the envelope and relative to other blocks should reflect the corresponding relative position in the physical domain.

Let´s consider the example given in Figure B.41(a). In this case, the computational position of block 1 is deviated in the up-right direction when compared with its relative position in the physical domain. As a result, the mesh will appear with a higher skewness degree. Changing the block computational position is done using the up-down and left-right arrows (cf. at right). **Note**: several blocks may be selected simultaneously in the *ListBox* and moved together.

Figure B.41 (b) shows the corrected layout.

Meshing a block

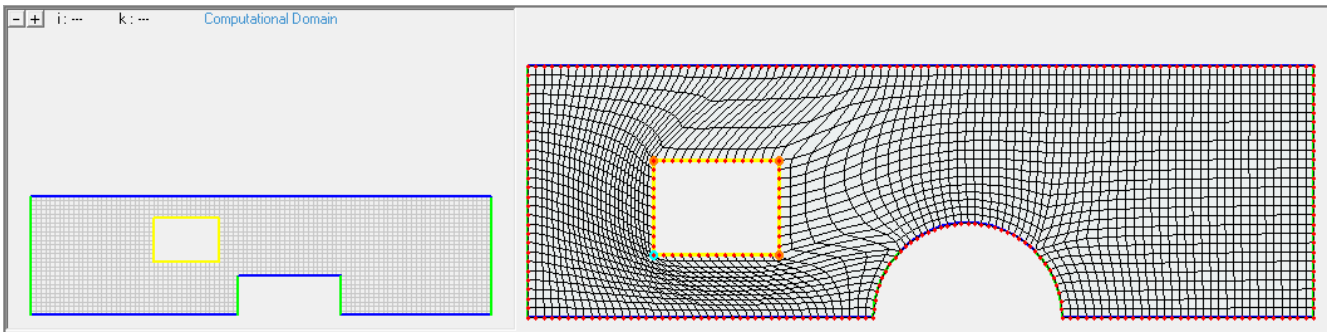
A block may be meshed or unmeshed.

As we´ll see in sections B1.6.7, B1.6.8 and B1.6.9, a block may represent:

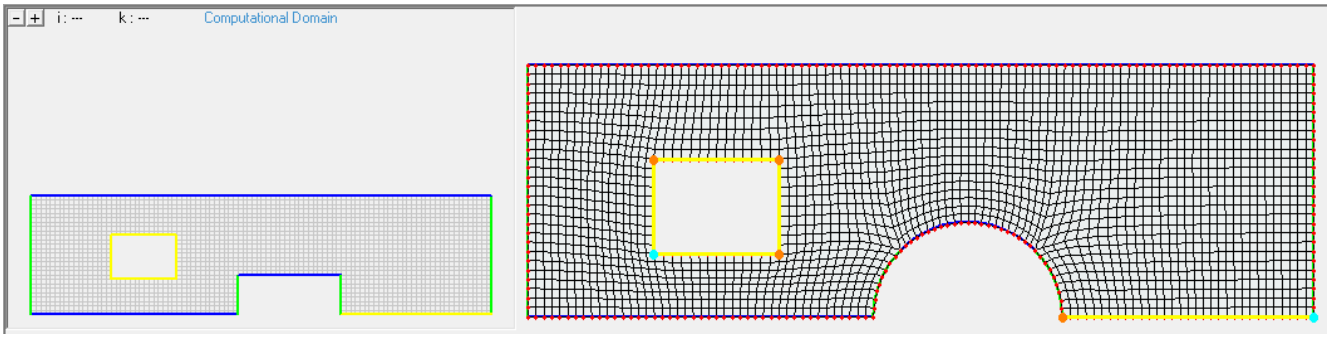
- **A void region**. In this case, the block must not be meshed.

- **A solid region** for conjugate heat transfer problems. In this case, the block must be meshed in order to solve the heat conduction equation.
- **A fluid region**. This may be a good option to achieve a better control of the mesh characteristics within a fluid region or simply to define a fluid region to compute statistical quantities in the post-processing. A meshed block representing a fluid region may also be used to simulate thin plate (cf. section B1.6.8).

Only meshed blocks may contain other blocks (meshed or unmeshed) within it. Figure B.42 shows an example of a meshed block containing a void block.



(a)



(b)

Figure B.41 – Influence of the blocks computational position on mesh quality.

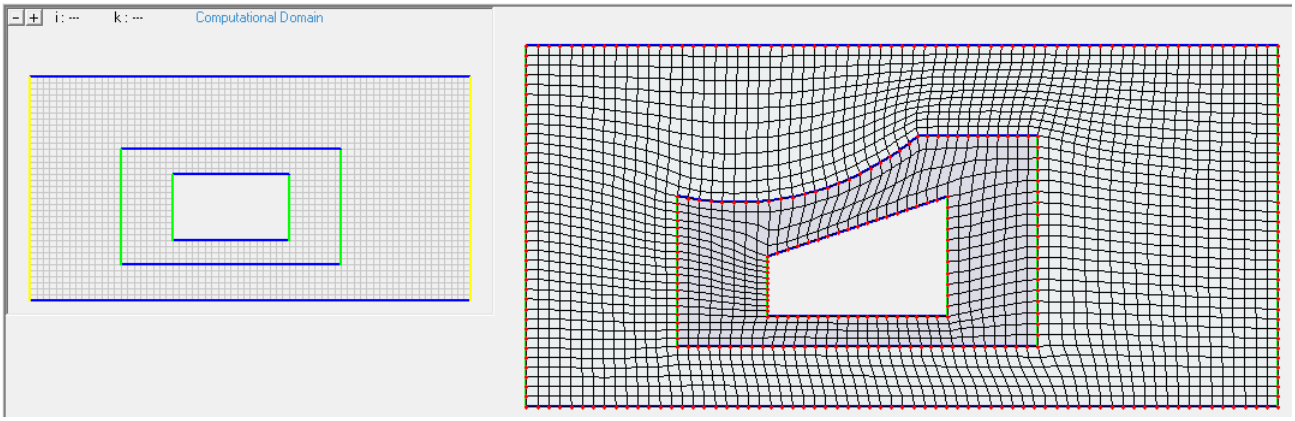
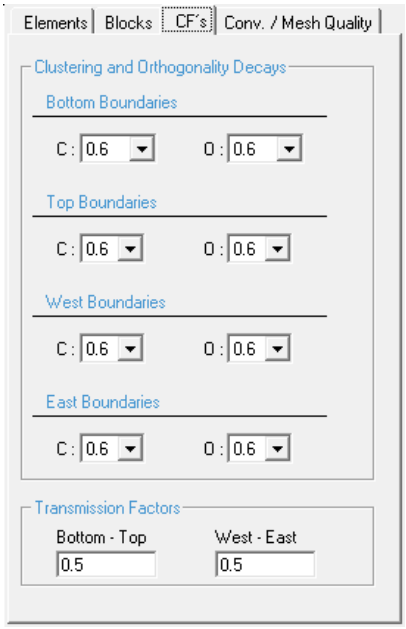


Figure B.42 – Example of meshed block containing a void block..

B1.3.7 – Control functions settings: The CF´s Tab



This menu contains the settings for the *Clustering and Orthogonality Decays* associated with the control functions by Steger and Sorenson (cf. Section A5.2.2). The decay factors *b* (*O*rthogonality decay) and *a* (*C*lustering decay), as defined in equations (5.6) and (5.7), may be chosen for each boundary location type (Bottom, Top, West and East).

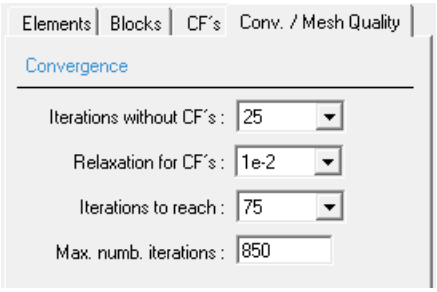
Control functions by Thomas and Middlecoff for transmission of the grid spacing into the interior of the domain (cf. section A5.2.2) are multiplied by a factor in the range 0 to 1, chosen independently for *Bottom-Top* transmission and for *West-East* transmission, as a means for adjusting their strength.

B1.3.8 – Numerical solution control and mesh quality: The Conv./Mesh Quality

Convergence

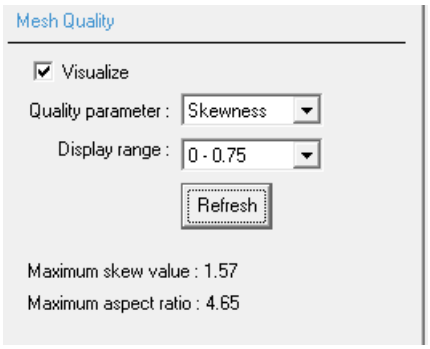
The process of numerically solving the mesh generation differential equations presented in section A5 may be prone to convergence difficulties unless the terms associated with the clustering and orthogonality control functions are under-relaxed. As referred in section A5.2.3, the under-relaxation factor for the control functions starts at zero (total under-relaxation) and is increased gradually (lesser under-relaxation) during the course of the iterations.

This menu allows the user to choose the settings to some parameters that help controlling the convergence process.



- *Iterations without CF´s*: number of iterations to be performed prior to introduce the control functions. Increasing this number may stabilize the calculation;
- *Relaxation for CF´s*: maximum value for the under-relaxation factor for the control functions. This is the most important parameter to control convergence. Its value should be decreased if converge difficulties as encountered;
- *Iterations to reach:* number of iterations during which the under-relaxation factor for the control functions is increasing. In the example shown above, the maximum value for the under-relaxation factor is reached at iteration *100* (= *25* + *75*). This value should be increased if converge difficulties as encountered;
- *Max. Numb. Iterations*: maximum number of iterations allowed to be performed during the mesh generation process. The process can, nevertheless, be stopped manually by pressing the button *Stop* and restarted by pressing again *Compute*.

Mesh Quality



Mesh quality is directly related to the mesh skewness and to the aspect ratio. The menu displayed at right allows you to visualize both these mesh parameters (by choosing *Quality parameter*), with a color code in the selected *Display range*.

Skewness: is a measure of the average angle between the mesh lines of each mesh element. Skweness 0 indicates perfect orthogonality , while value 1 is obtained for a 45º angle.

Aspect ratio: represents the ratio between the maximum and the minimum edge length of a mesh element.

Figure B.43(a) and (b) show visualizations for both mesh quality parameters.

(*a*)

(*b*)

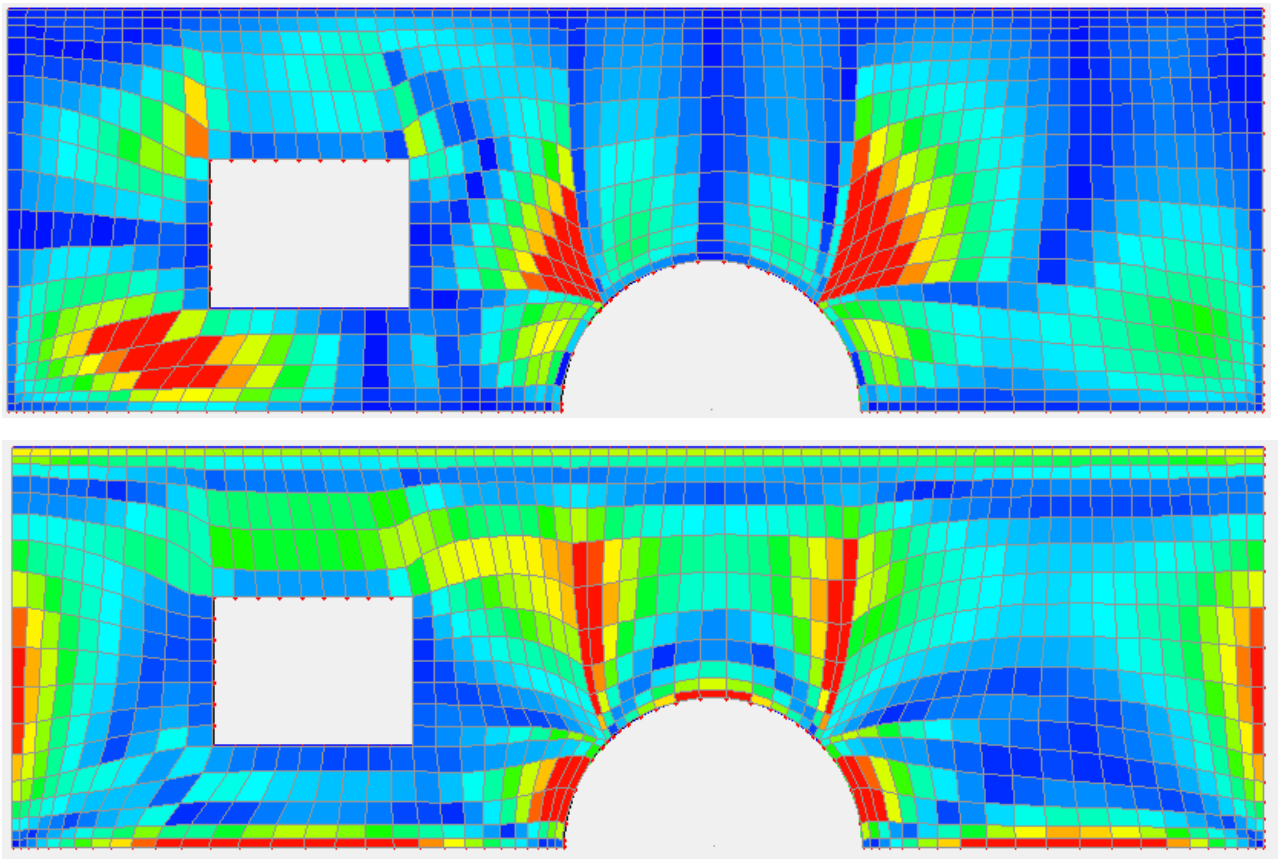
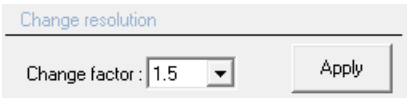


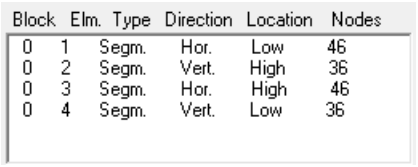
Figure B.43 – Example of visualization of mesh quality. a) Skewness in the range 0-0.5; b) Aspect ratio in the range 1-3.

B1.3.9 – Other menus

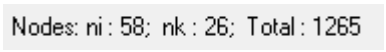


- *Change resolution*: this feature allows changing the number of mesh nodes of the selected elements by the factor specified in *Change factor*. Mesh spacings are also affected by this factor. This is a very convenient way to change the mesh resolution for the whole geometry.

- Elements listing: a list of all the geometric elements is provided, along with some information regarding its type, computational domain direction and location and number of mesh nodes. A double-click on an element, using this list or in the visualization display, opens a window with detailed information, as shown in Figure B.44.



- *Nodes*: presents the maximum indexation of mesh nodes in each computational direction, along with the net number of mesh nodes in the calculation domain.



- *Verify*: verifies all mesh settings to check for errors. This verification is also automatically performed before computing the mesh.

- *Compute* and *Stop*: these buttons (re)start and stop, respectively, the mesh computation.

- Horizontal bar: the horizontal bar provides visualisation for the percentage of iterations accomplished as relative to the maximum allowed.

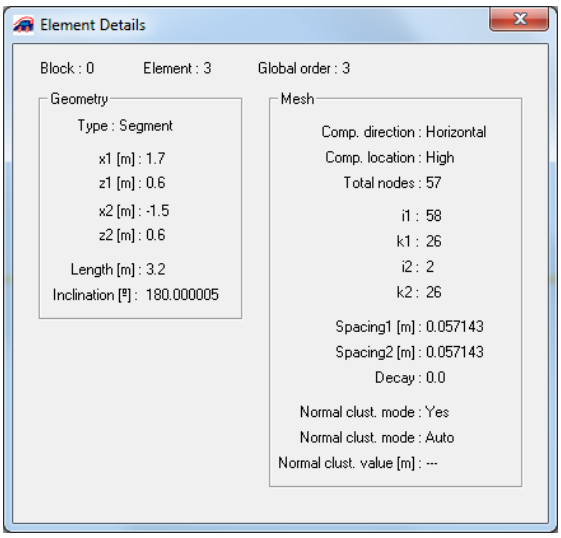


Figure B.44 – Window showing element details.

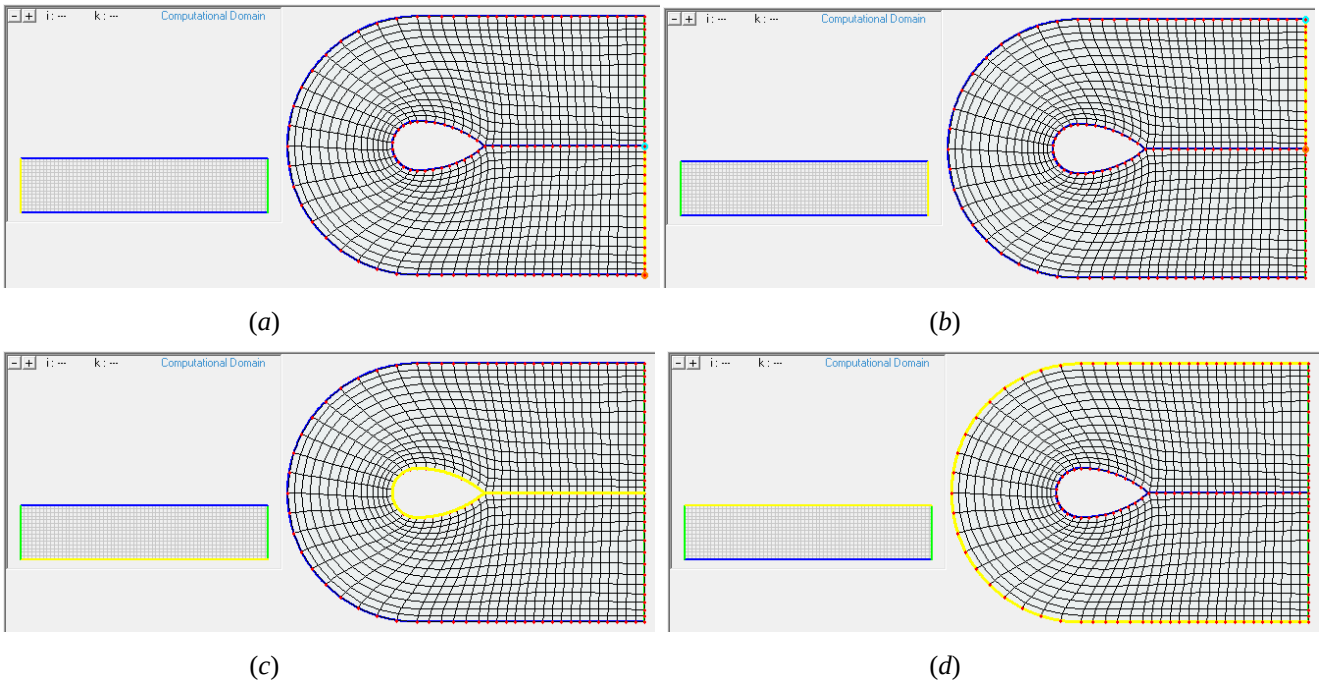
B1.3.10 – Using C-Type and U-Type meshes

Cf. example files *NACA2412_UMesh.def* and *C-TypeMesh.def*

Sometimes, the utilization of a C-type or even a U-Type mesh is the best solution for getting a good mesh quality, when computing the flow around a closed body. C-Type and U-Type meshes both take a folded domain that touches himself along a line segment, like the one presented in section B1.2.5.

U-type meshes are well suited for bodies that possess a round edge and a sharper edge, like an airfoil. Figure B.45 shows the computational arrangement for a U-type mesh generated around such a body. Note, for the yellow line, the correspondence between its location in the physical domain and in

the computational domain. By default, the line where the mesh touches itself (horizontal straight yellow segment in Figure B.45(c)) is considered as pervious to the flow.



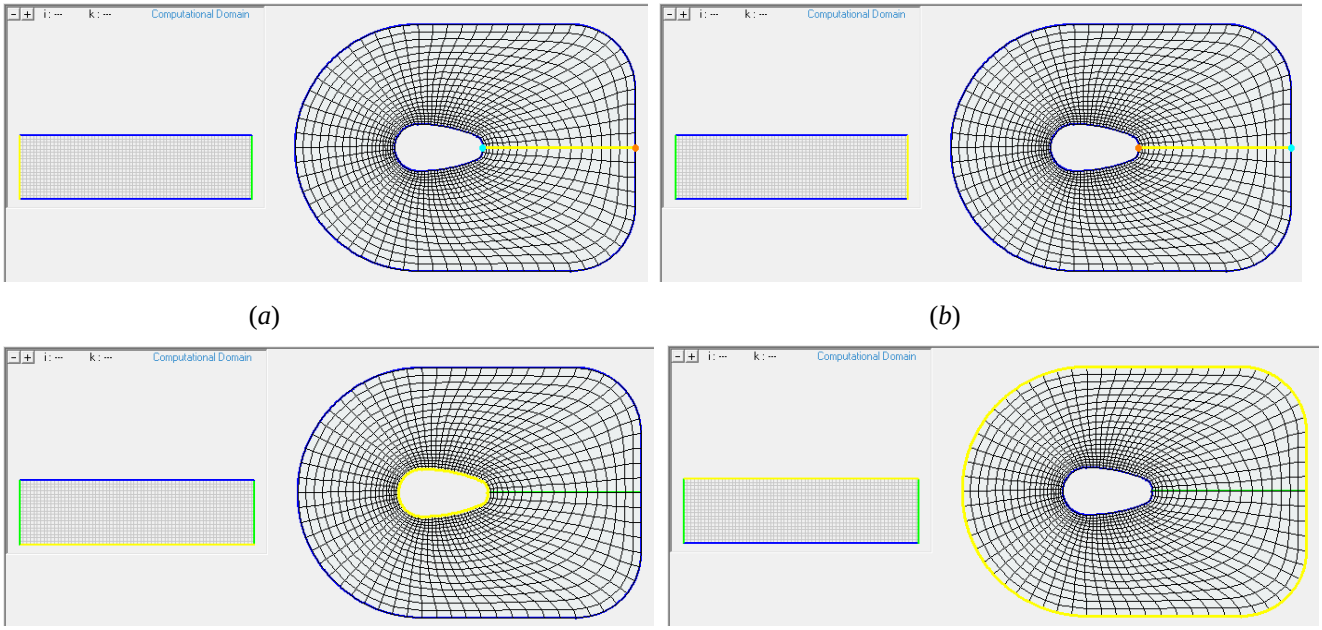
(*a*) (*b*)

(*c*) (*d*)

Figure B.45 – Computational arrangement for a U-Type mesh.

Although C-type meshes can be used for bodies with a sharp edge, like the one presented above, they are more suited for rounded bodies. Figure B.46 shows the computational arrangement for a C-type mesh around such a body. Note that the coincident horizontal lines are, in this case, in opposite sides of the computational domain.

For both U-type and C-Type meshes, the central body is defined using the envelope itself.



(*a*) (*b*)

(*c*) (*d*)

Figure B.46 – Computational arrangement for a C-Type mesh.

B1.3.11 – Using split meshes

Cf. example file *SpliMesh.def*

When the leading and the trailing edge of a body are both sharp, it may be more convenient to employ a split mesh. In this case, the body is defined as block with zero thickness in the computational domain. **EsyCFD** allows only for split meshes in the horizontal computational direction. Figure B.47 shows an example.

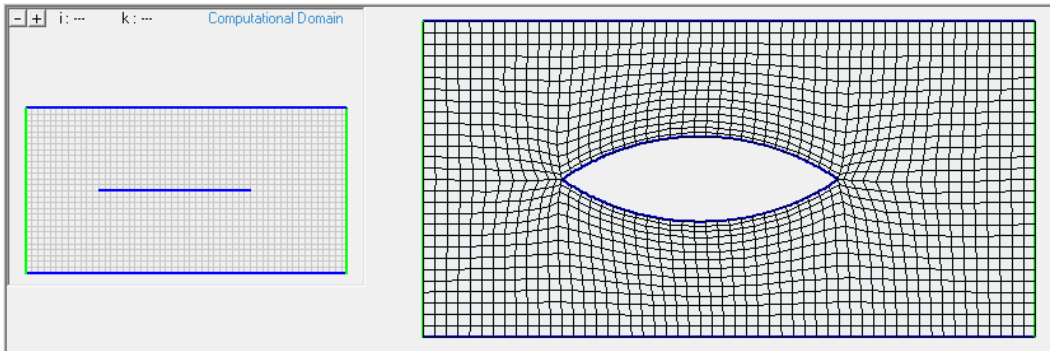
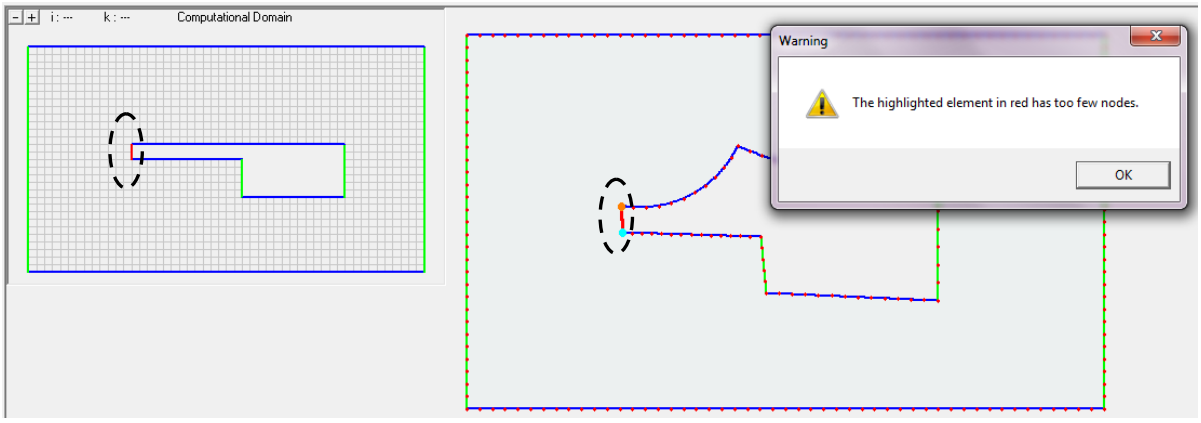


Figure B.47 – Computational arrangement for a split mesh.

B1.3.12 – Some advices on mesh generation

Every feature in the geometry must be meshed with, at least, four nodes (in which the edge nodes are included). This comprises geometric elements (such as depicted in Figure B.48(a)) and also gaps between geometric elements (such as depicted in Figure B.48(b)).

(a)



(b)

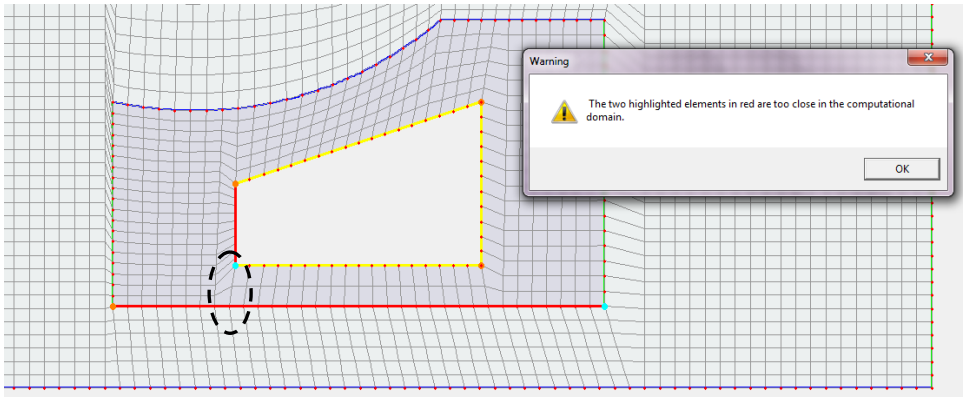
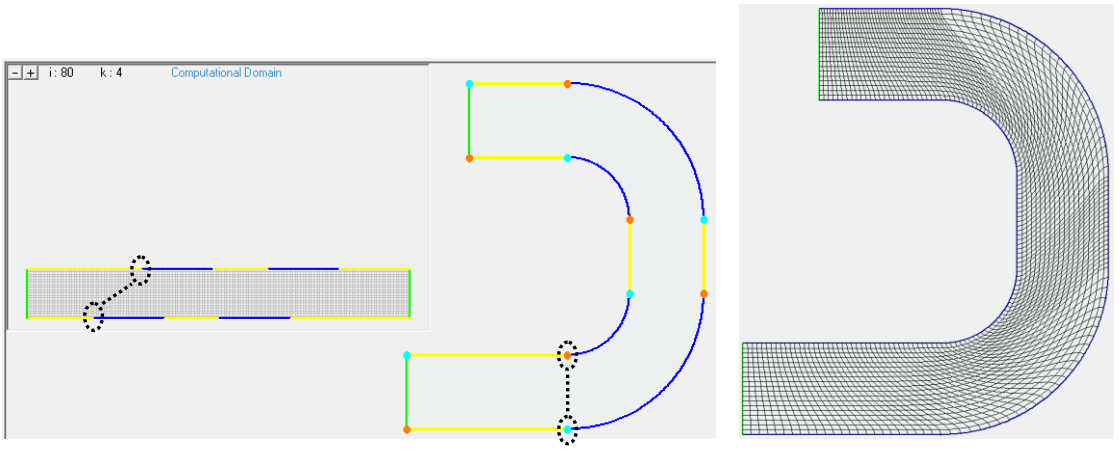


Figure B.48 – Examples of insufficient mesh nodes to resolve a feature.

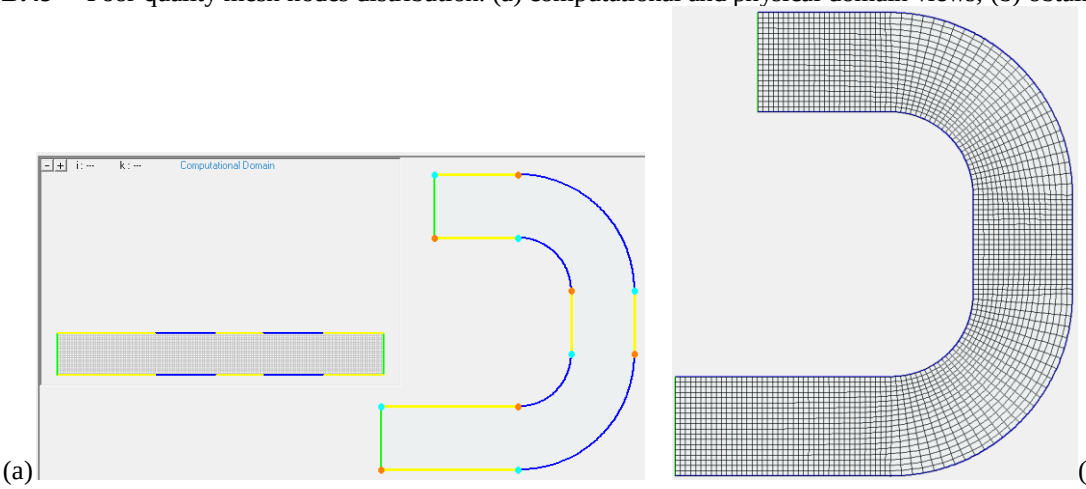
In order to reduce skewness, the relative distances and comparative lengths in the physical domain should be preserved as much as possible in the computational domain. This can be achieved by:

- properly positioning the inner blocks using the tools available in the *Comp. Map* tab, as described in section B1.3.6.
- adjusting the distribution of mesh nodes between the geometric elements. In the example shown in Figure B.49, you may notice that elements of equal length in the physical domain have different lengths in the computational domain (i.e., these elements have different number of mesh nodes). This may lead to a skewed mesh, as shown in Figure B.49(b). The corrected mesh is presented in Figure B.50.

(a) 

(b)

Figure B.49 – Poor quality mesh nodes distribution. (a) computational and physical domain views; (b) obtained mesh.

(a) 

(b)

Figure B.50 – Correct mesh nodes distribution. (a) computational and physical domain views; (b) obtained mesh.

Often, breaking some geometric elements at selected locations may constitute a great help for the meshing stage. Figure B.51 shows the mesh around a car shape. In this example, the low boundary of the envelope (soil) was broken at two location vertically aligned with the vehicle front and rear ends, in order to help finding the correct computational position and size for block 1 (representing the vehicle).

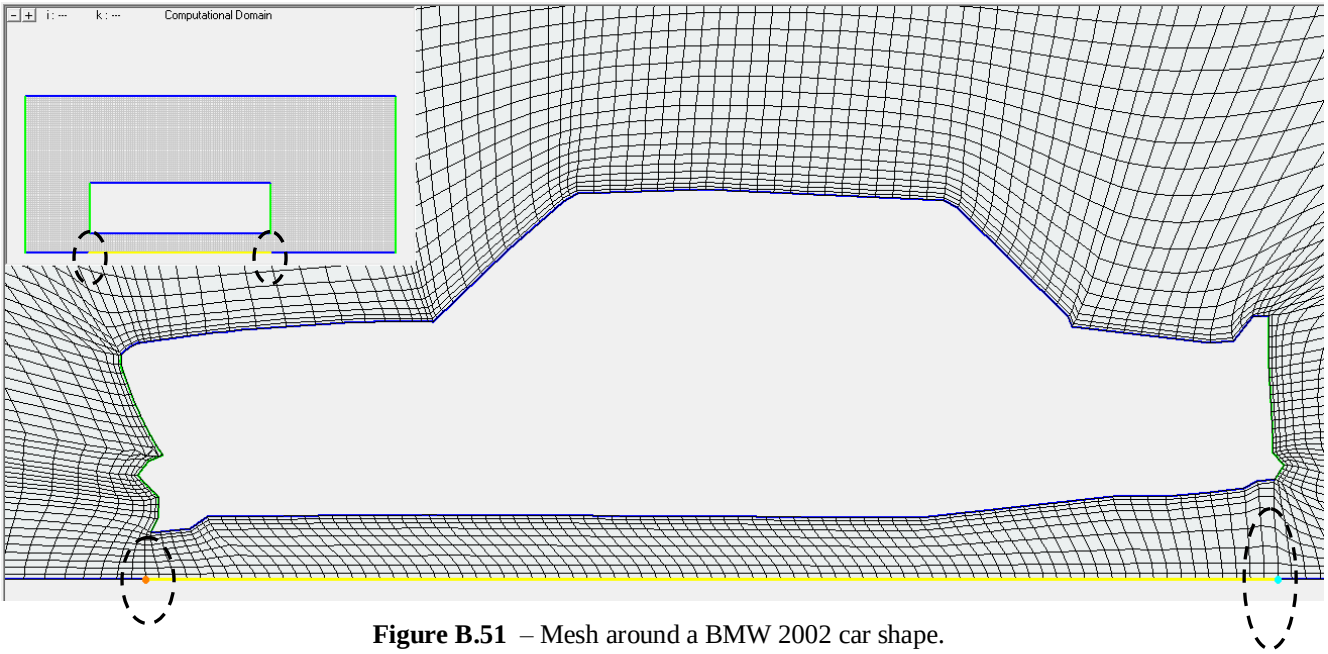
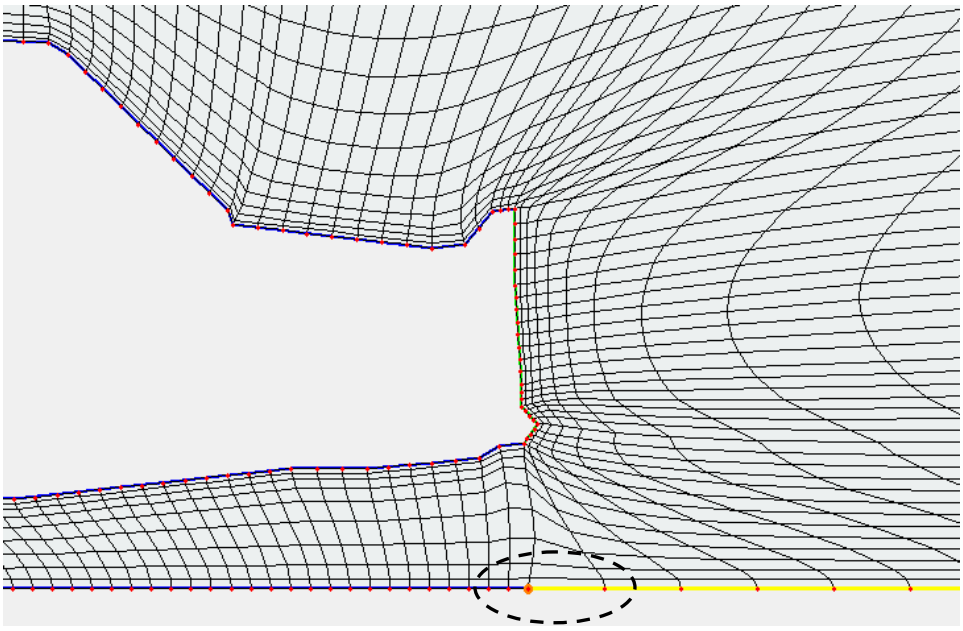
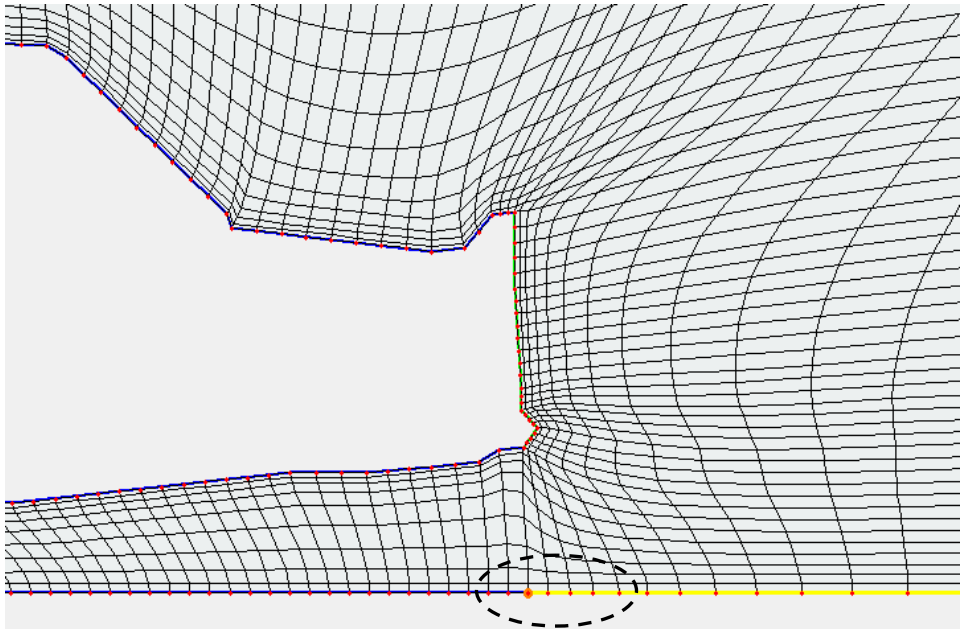


Figure B.51 – Mesh around a BMW 2002 car shape.

Abrupt changes in mesh size should also be avoided. This can occur when two contiguous geometrics elements have imposed number of nodes with uniform spacing, such as depicted in Figure B.52(a). This is easily corrected without changing the number of nodes, simply by specifying, in the *Boundary Nodes Tab*, a *Specified Type* (cf. section B1.3.2) and ensuring that the contiguous elements share a similar mesh size, by checking *Update neighbours*. Figure B.52(b) presents the correct solution.

(a)





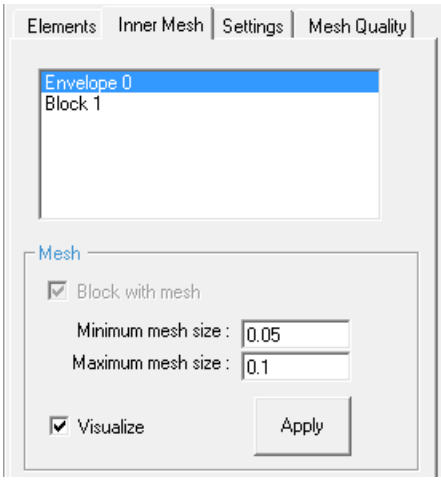
(b)

Figure B.52 – Meshing of contiguous elements. (a) abrupt change in mesh size; (b) correct solution, ensuring a smooth transition.

B1.4 – Mesh Generation: Unstructured type

From the user point of view, mesh generation for the unstructured type is much simpler than for structured type. In this case, the concept of vertical and horizontal computational directions does not exist, since there are no computational alignments. Generation for the 1D meshes on the domain boundaries is identical to the structured type.

B1.4.1 – Inner Mesh Size



For each closed block, provided a mesh is to be generated in that block, minimum and maximum values for mesh size must be specified. The mesh generation process tries to ensure that mesh size is within these limits through the insertion of tucks and/or wedges (cf. section A5.3.5).

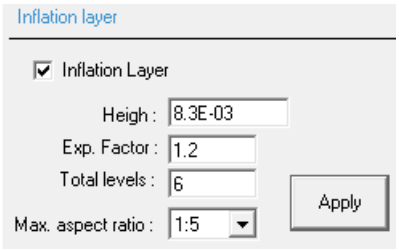
By checking *Visualize*, the two chosen sizes are visualized by two squares displayed at the top right side of the visualization area.

If these values are not changed by the user, default values are assigned when mesh generation is started.

B1.4.2 – Inflation layers

By default, rows are built with the goal of obtaining nearly squared elements. This may not be desirable near boundaries. In these regions, inflations layers may be used to control the projection distance. The distance of the first node to the boundary (inflation layer first *Height*), the ratio between the height of two consecutive layers (*Exp. Factor*) and *Total levels* must be specified.

The maximum aspect ratio controls the aspect ratio of the first layer volumes, by adjusting the nodes distribution along the boundary geometric element.



When using inflation layers, the nodes distribution along the boundary geometric element is adjusted so as to satisfy the specified maximum aspect ratio. Near corners and reversals, the nodes spacing is automatically adjusted so as to achieve nearly squared elements. This is exemplified in the next figure.

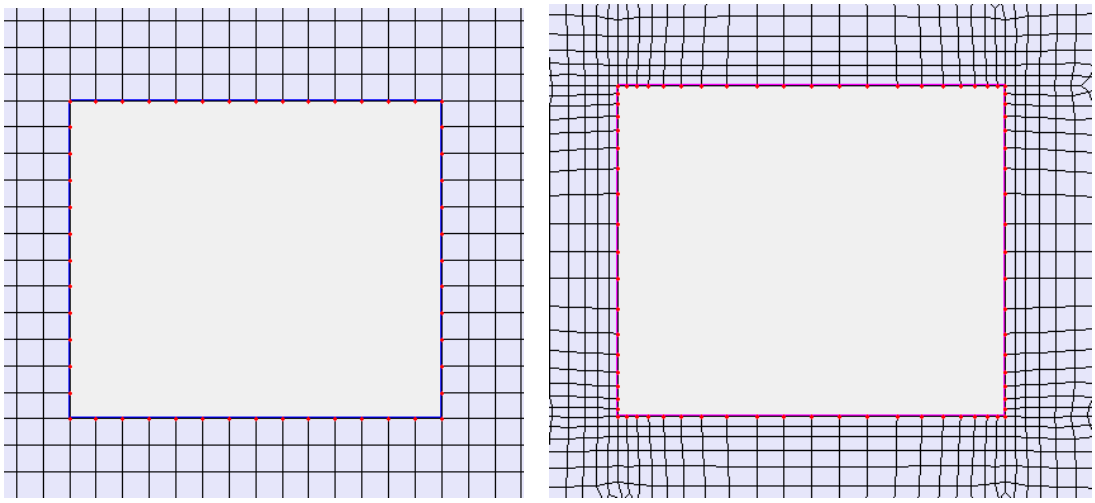
(a)  (b)

Figure B.53 – Examples of automatic adjustment of nodes along the boundary elements, when using inflation layers. (a) – without inflation layer; (b) – with inflation layer.

Near ends, the neighbour element nodes spacing is automatically adjusted to allow the correct growth of the inflation layer. This is exemplified in the next figure.

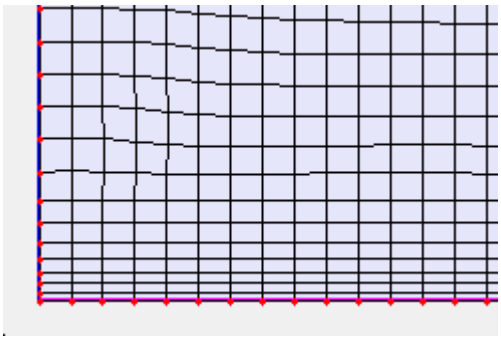
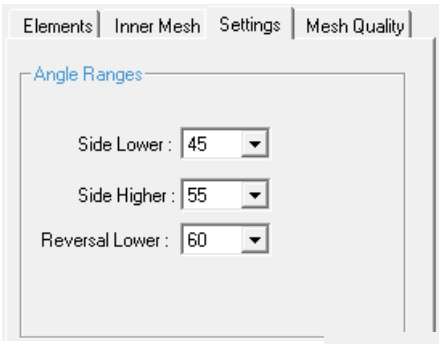


Figure B.54 – When a end exists near an inflation layer, the nodes distribution of the neighbour element is automatically adjusted.

An inflation layer with a maximum aspect ratio of 1:3 is by default assigned to all geometric elements of inner blocks. Elements in the envelope are not assigned, by default, any inflation layer. This can, of course, be override with the appropriate controls.

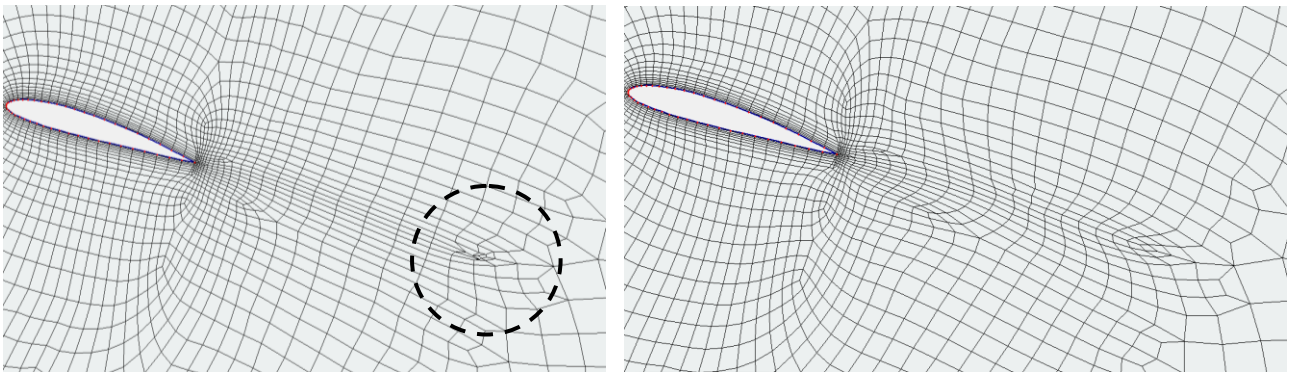
B1.4.3 – Settings



Settings for unstructured meshes consist on the specification of values for the sigma values defining the type of row projection, as described in section A5.3.1 (*Side Lower*: σ_1; . *Side Higher*: σ_2; . *Reversal Lower*: σ_3).

B1.4.4 – Smoothing the mesh

Although a mesh smoothing algorithm (cf. Blacker and Stephenson, 1991 for details) is applied after mesh generation, in order to ensure smooth transitions, further smoothing may be necessary to further enhance the mesh. This is accomplished every time the *Smooth Mesh* button is pressed. This is exemplified in Figure B.55 where further smoothing was applied in order to smooth some irregularities that arose from the intersection of two paving rows.



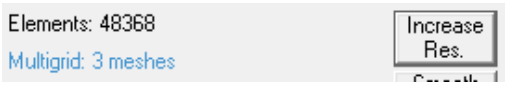
(a) (b)

Figure B.55 – Mesh further smoothing after mesh generation. (a) before further smoothing; (b) after further smoothing.

B1.4.5 – Increasing mesh resolution through interpolation; multigrid.

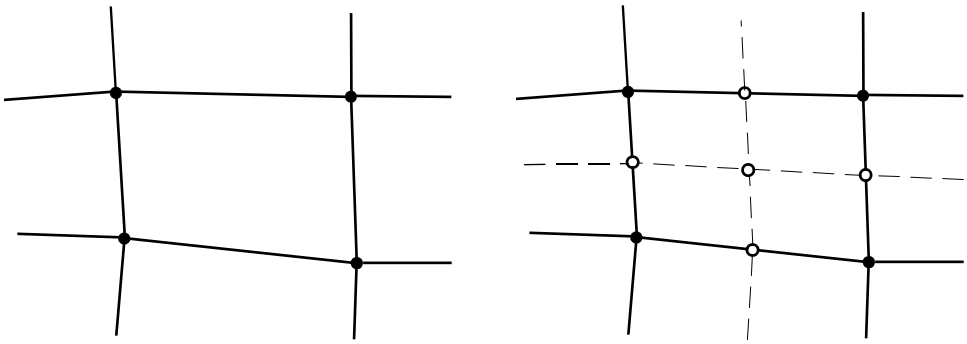
Large non-structured meshes can be very time consuming to generate and also lead to lengthy computations when solving the fluid flow. In the interpolation method, each mesh element (control volume) is split into four elements, a process that is carried out quite fast.

When increasing mesh resolution, the new mesh is automatically saved as (taking *ExampleCase* as the project name):



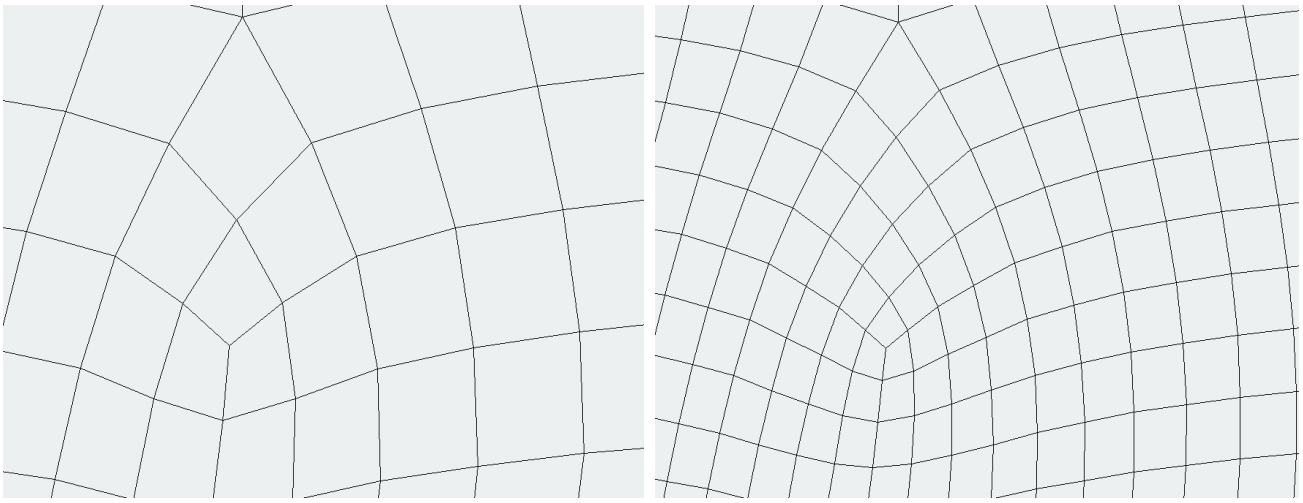
ExampleCase_n.msh

where *n* is the mesh level (*0* for the base mesh). This will allow multigrid calculations, as described in B2.1.3).



(a) (b)

Figure B.56 – Interpolation process for splitting the mesh. (a) before split; (b) after split.



(a) (b)

Figure B.57 – Mesh (a) before split; (b) after split.

B1.4.6 – Some advices on unstructured mesh generation

Intersection of rows close to the boundaries

The intersection of paving rows is the most critical process when building the mesh. This can lead to some irregularities that, most often, are removed by the mesh smoothing algorithm. Nevertheless, this algorithm is not applied for nodes less than 5 rows from the boundary. If two rows intersect less than 5 rows from the boundary, the irregularities cannot be removed, as exemplified in Figure B.58. In this case, the solution is to decrease the mesh clustering value at the boundaries.

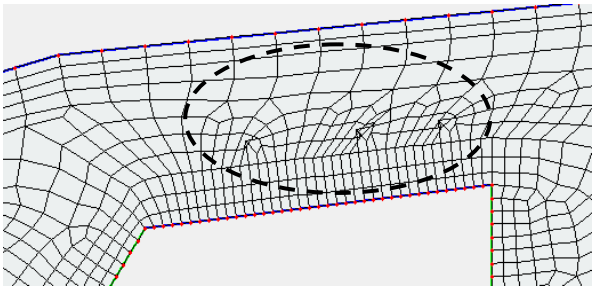


Figure B.58 – Irregularities created by the intersection of two rows.

Using closed inner blocks to help controlling mesh size

Closed inner blocks may be useful to control mesh size near the region of interest. This exemplified in Figure B.59 for the mesh around an airfoil.

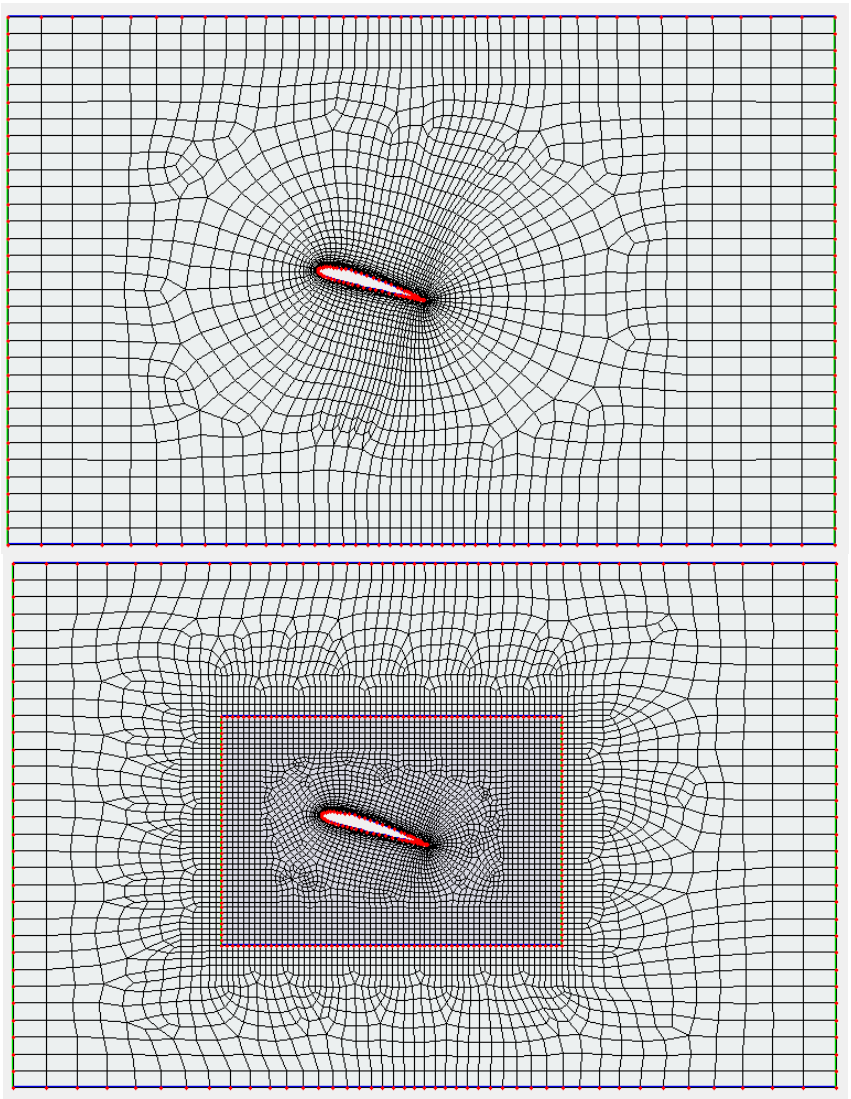
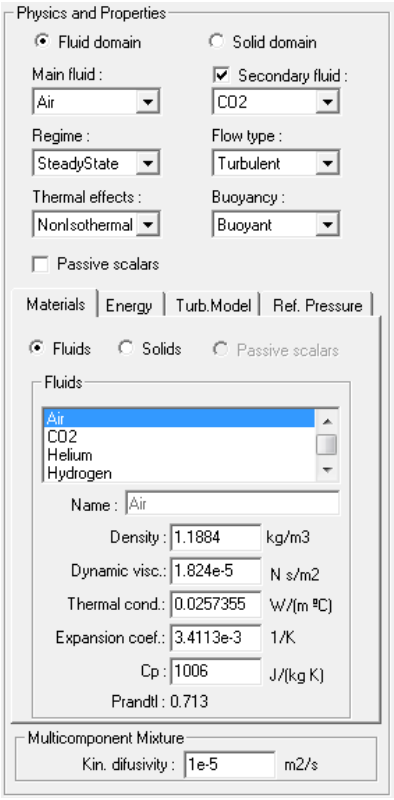


Figure B.59 – Using a meshed inner block to help controlling mesh size; (a) without inner block; (b) with inner block.

B1.5 – Physics and Properties



The physics of the problem are defined in this section.

B1.5.1 – Main settings

The first two *OptionButtons* 

allow the choice between:

- *Fluid domain*, which may also include solid blocks;
- *Solid domain*, where no fluid is present. This case corresponds to a pure conduction problem.

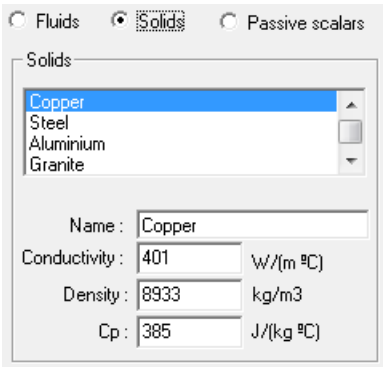
The next two *Combo-Boxes* allow choosing the *Main fluid* and the *Secondary fluid*, if multi-component flow is considered (by checking the *Secondary fluid* button).

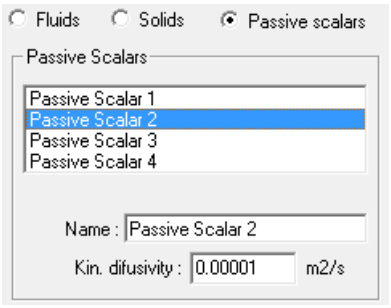
Solution for *Passive scalars* is activated in the corresponding *Check-Box*. 

The next four *ComboBoxes* allow definition of the flow *Regime* (*SteadyState* or *Transient*), the *Flow type* (*Laminar* or *Turbulent*), the *Thermal effects* (*Isothermal* or *NonIsothermal*) and the *Buoyancy* inclusion (*Buoyant* or *non-Buoyant*).

B1.5.2 – Materials

Several materials are predefined, for *Fluids*, *Solids* and for *Passive scalars*. Their properties can be changed in the corresponding *TextBoxes*, as shown on the right. The fluids *Air* and *Water* are special in the sense that only they allow non-Boussinesq calculations (cf. section A1.8). The names of these two fluids cannot be changed.

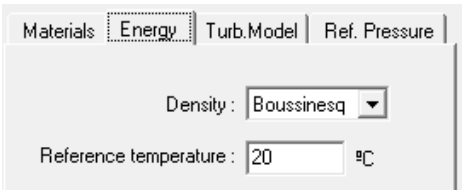




B1.5.3 – Energy

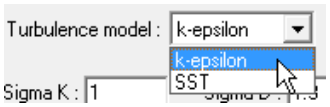
The user can choose to adopt, or not, the *Boussinesq* approximation to deal with *Density* variations (cf. section A1.8).

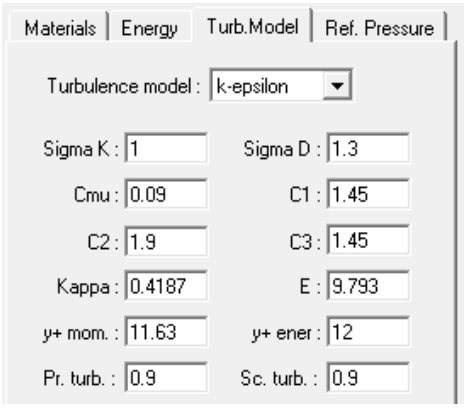
The *Reference temperature*, necessary to compute the buoyancy term, should be specified on the *TextBox* shown on the right.



B1.5.4 – Turbulence model

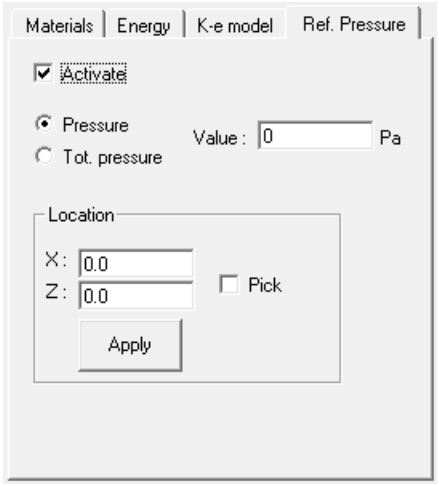
If *k-ε* is the chosen model, the values for several variables of this model can be changed in the *TextBoxes* shown on the right.





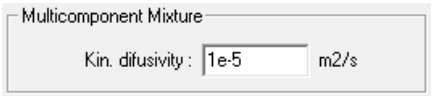
B1.5.5 – Reference pressure

As described in A3.2, pressure levels are established by pressure corrections that take place during the iterative process. Final pressure values are, thus, arbitrary (though pressure gradients are not). The user can, nevertheless, impose a desired pressure (or total pressure) at a chosen location in the fluid domain (the remaining pressure field will, thus, be shifted according to this imposition). The location may be specified through the *X* and *Z* coordinates, or picked with the mouse.



B1.5.6 – Multi-component mixture properties

The kinematic diffusivity characterizing the fluids pair may be changed in the *TextBox* shown at right.

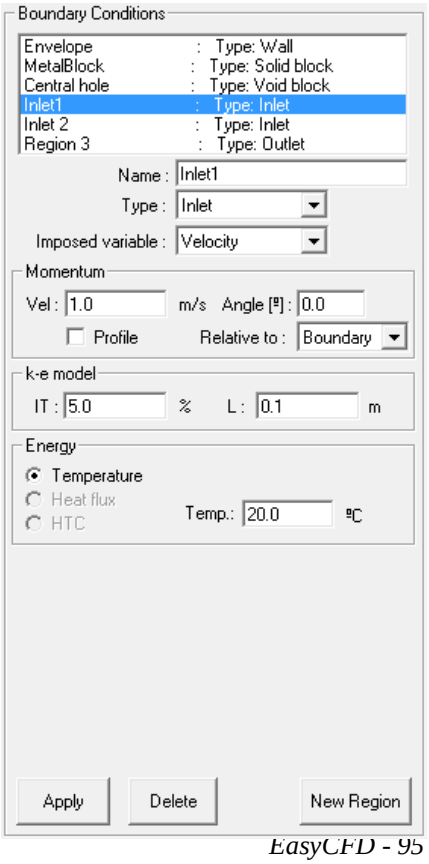


B1.6 – Regions and Boundary Conditions

Regions are zones, in the envelope or in blocks, which are created with the purpose of imposing particular boundary conditions, or simply for analysis in the post-processing.

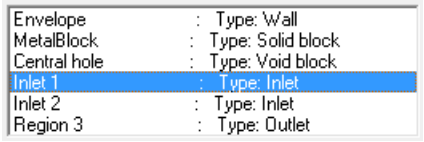
The envelope and blocks constitute, themselves, regions that are automatically created. These regions (envelope and blocks) are unique, in the sense that they cannot be deleted and their boundary condition for *Momentum* is *Wall* with zero velocity (for blocks, this applies to those which are impervious to the flow: void blocks and solid blocks).

You can create, in the envelope, in void blocks and in solid blocks, other regions with specific characteristics. The characteristics of these new regions replace the default attributes for the envelope or the block (including creating wall boundaries with non-null velocity). To create a new region proceed as follows:



1. Select the boundary elements to assign to the new region (note: the selected elements cannot be already assigned to other region);
2. Press the button *New Region*.

With this procedure, the newly created region will have the defaults attributes (wall type, zero velocity and zero heat flux).

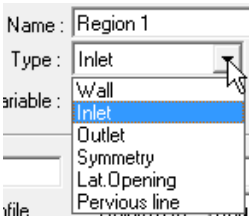


To change the settings for a region, make the corresponding selection in the Regions List (cf. at right), choose the desired settings and press *Apply*. If one or more boundary elements are also selected, this process will also change the boundary elements associated to the region (the selected region will be composed exclusively by the selected elements).

Please note that the new settings become effective only after pressing the button *Apply*. Values and attributes not yet effective are easily identifiable by their red colour.

Each region is identified by its *Name*, as shown on the right.

The *Type* of boundary condition is selected in the corresponding *ComboBox*. For *Inlets* and *Outlets*, different *Imposed variables* are available.

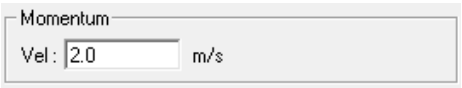


Boundary values assigned to regions can be either constant or given as a *TableFunction* (cf. section B1.6.12).

B1.6.1 – *Wall* boundary condition

Momentum

The wall velocity (*Vel*) is defined as positive when it corresponds to a counter-clockwise rotation of the corresponding block. The velocity direction is displayed with an arrow, as exemplified **in Figure B.60.**



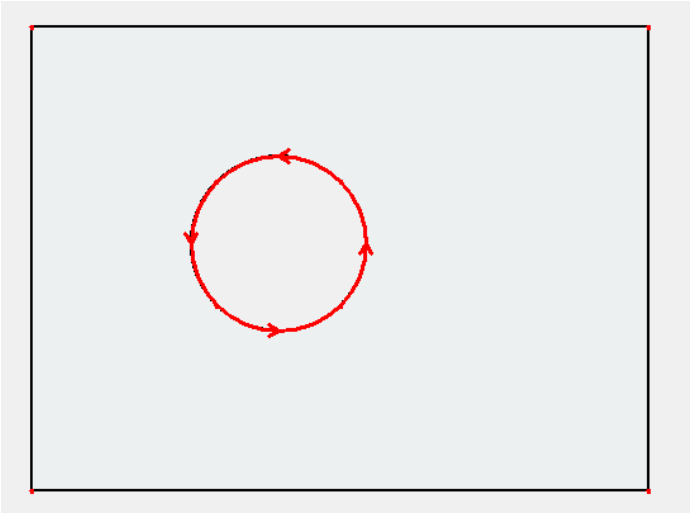
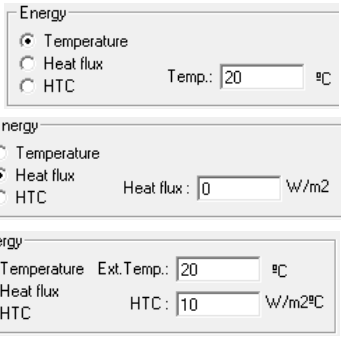


Figure B.60 – Simulation of a rotating cylinder. A new region was created with the four elements composing the cylinder and a positive velocity was assigned to that region.

Energy

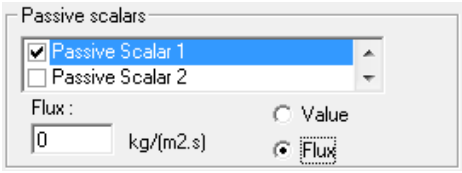
The following options are available:



- Imposition of wall *Temperature*;
- Imposition of wall *Heat flux* (the heat flux is defined as positive when entering the domain);
- Imposition of a Heat Transfer Coefficient (*HTC*) and external temperature (*Ext. Temp*).

Passive scalars

The wall condition may be set independently for each of the four scalars available simultaneously. Options are *Value* and *Flux*. As for energy, the flux is positive if entering the domain.

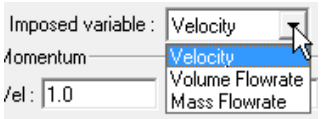


Secondary fluid concentration

For multi-component fluid flow situations, walls are treated as impervious for secondary fluid concentration. A zero flux is, thus, assigned.

B1.6.2 – *Inlet* boundary condition

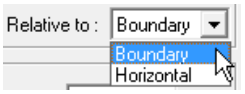
Three different choices are available for the imposed variable (flow condition) in inlets:



Velocity, *Volume Flowrate*, *Mass Flowrate:* Values must be positive if a constant value is specified (corresponding to a uniform velocity profile). If a *TableFunction* is used (cf. section B1.6.12), negative values are allowed for time-dependent conditions. This allows the simulation of pulsatile flows (cf. example in section C5).

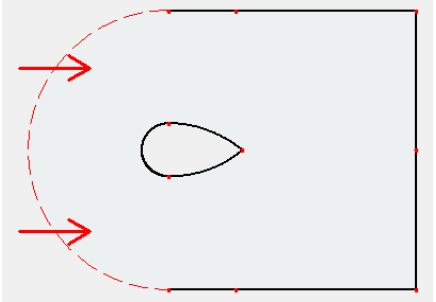
The angle θ is the angle, in degrees, between the velocity vector and:

- the vector perpendicular to the inlet, pointing into the domain, if *Relative to: Boundary* is selected;



- the x-axis (horizontal direction) if *Relative to: Horizontal* is selected.

The *Relative to: Horizontal* is the proper choice when using C-Type or U-Type meshes, as in these cases, most likely, the inlet boundary is a curved line (cf. example at right).



When the option *Profile* is checked, the inlet profile is made similar to the profile solution at certain computational distance downstream of the inlet. This transposition process takes place at every

iteration, ensuring that the inlet mass flow remains unchanged. An inlet developed velocity profile is, thus, easily obtained, without the need for a large entrance length. The computational distance downstream and relaxation factor to apply in this process are defined in *Calculation parameters* (cf. section B1.8.1). The downstream location from which the profile is transposed is represented as shown in Figure B.61.

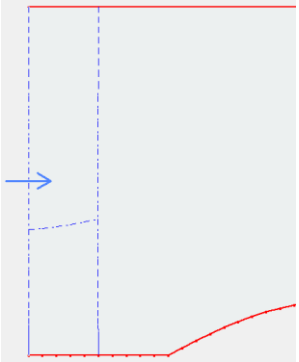


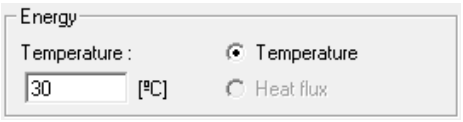
Figure B.61 – Representation of an inlet with profile transposition from downstream.

k-ε model

The input data are:

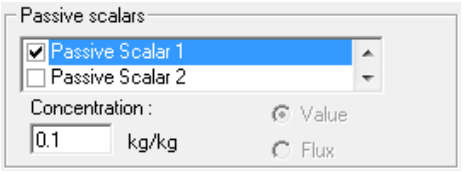
- *IT*: Turbulence intensity $IT = 100 \times \sqrt{\frac{2}{3}k} / V_{in}$, where V_{in} is the inlet velocity; [%];
- *L*: Turbulence length scale $L = \frac{k^{3/2}}{\varepsilon}$; [m]. *L* may be taken from a fraction of the inlet dimension (typically*10%*).

Energy



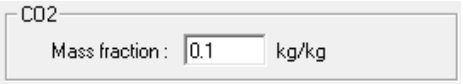
The *Temperature* value must be specified for all the inlet boundaries.

Passive scalars



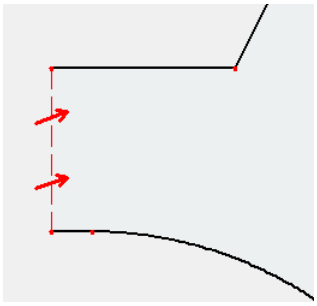
The value for the concentration of passive scalars must be specified for all inlet boundaries.

Secondary fluid concentration

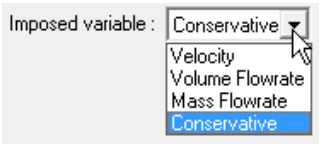


For multi-component fluid flow situations, the mass fraction of the secondary fluid must be specified for all inlet boundaries

After validating the entered values with the button *Apply*, the inlet region is displayed with an arrow that indicates the flow direction.



B1.6.3 – *Outlet* boundary condition



The choice of the flow condition at the outlets includes the available options for inlets plus the additional option *Conservative*.

Unlike inlets, a fundamental difference exists between the *Velocity* and the remaining options. If *Velocity* is chosen, a uniform velocity profile is imposed. For the remaining options, the velocity profile shape computed at the control volumes upstream of the outlet is assigned to this boundary.

The outlet mass flow rate is automatically computed for all the *Conservative* outlets so as to satisfy global mass conservation.

For all the remaining scalars (temperature, turbulence quantities, passive scalars and secondary fluid concentration), a parabolic condition is assigned; thus, no other specification is needed.

B1.6.4 – Symmetry boundary condition

No values are necessary to be defined for this type of boundary.

B1.6.5 – *Lateral opening* boundary condition

Lateral openings are boundaries that may accept inflow and outflow, or a mix of both. Inlet velocities are not specified at these boundaries, as they come as a result of the computation. Values for scalar quantities (such as temperature, turbulent kinetic energy, etc,) must be specified, to be used if inflow is verified. This boundary type is to be used for situations where the flow is predominantly parallel to the boundary. This boundary is considered as a symmetry boundary during the first 10 iterations, thus providing better numerical stability than if an outlet type boundary was used. Figure B.62 shows a typical application case, where a boundary layer is developing from the bottom boundary, creating a small vertical velocity component that leads to a very small (yet not negligible) outflow at the top boundary.

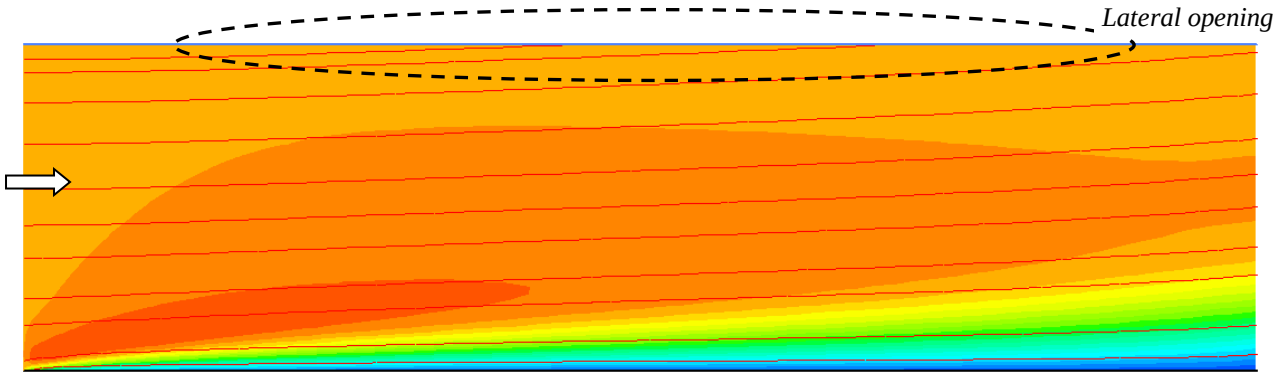


Figure B.62 – Utilisation of a lateral opening for a boundary-layer development.

As referred above, lateral openings admit inflow. Nevertheless, this may lead to numerical instabilities and erroneous results if the inflow region is dominant in the corresponding boundary. This is exemplified in Figure B.63, where the top boundary presents a large portion of inflow. In this case, a much better solution would be to increase the calculation domain, by displacing the top and bottom boundaries farther from the airfoil. This would even allow you to use a symmetry boundary instead (which is equivalent to a slip wall), which is more robust numerically. In this case, you should ensure that the lateral walls are far enough so they do not influence significantly the flow around the airfoil.

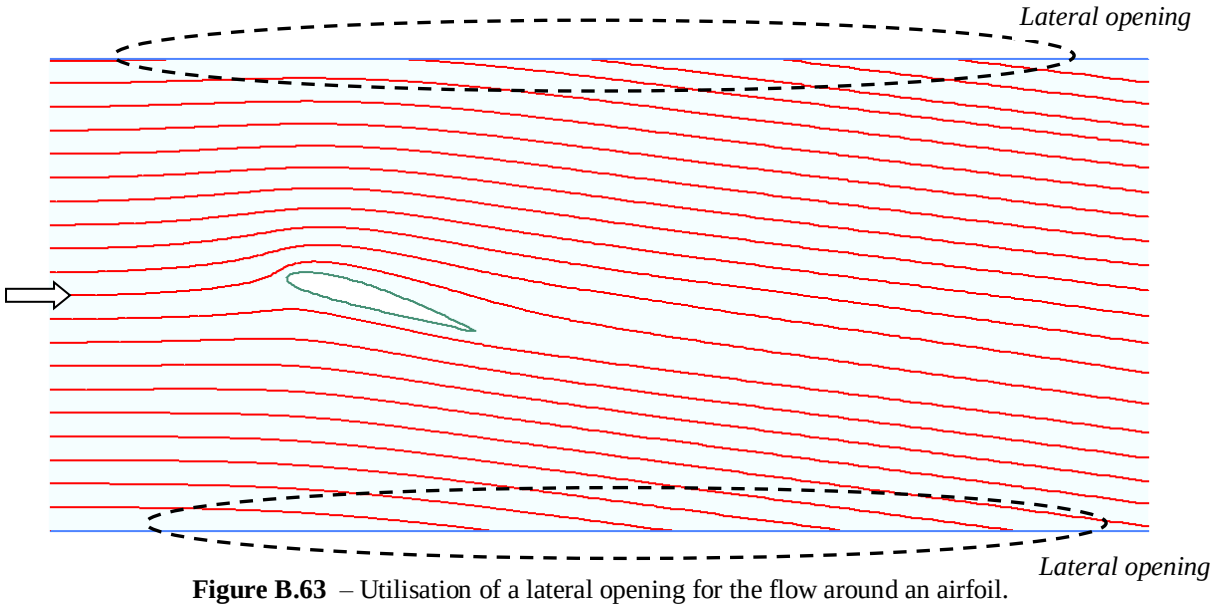


Figure B.63 – Utilisation of a lateral opening for the flow around an airfoil.

In summary, lateral openings should be used when the flow is expected to be predominantly parallel to the boundary, mostly with outflow component (as compared to inflow). Inflow as admitted as long as it is restricted to a small portion of the boundary.

B1.6.6 – *Pervious line* boundary condition

This boundary condition is used to impose specific values for fluid velocity inside the calculation domain (to simulate a fan for instance). It can only be assigned to delimiting boundaries of pervious blocks. You may impose velocity values and direction, in a similar way as for inlets. For pervious lines, negative velocity values are allowed. Imposition of velocity values can also be done directly to open blocks. In this case, there is no need to create a new region.

Figure B.64 demonstrates the utilization of a pervious line region to simulate a fan inside a cylindrical shell.

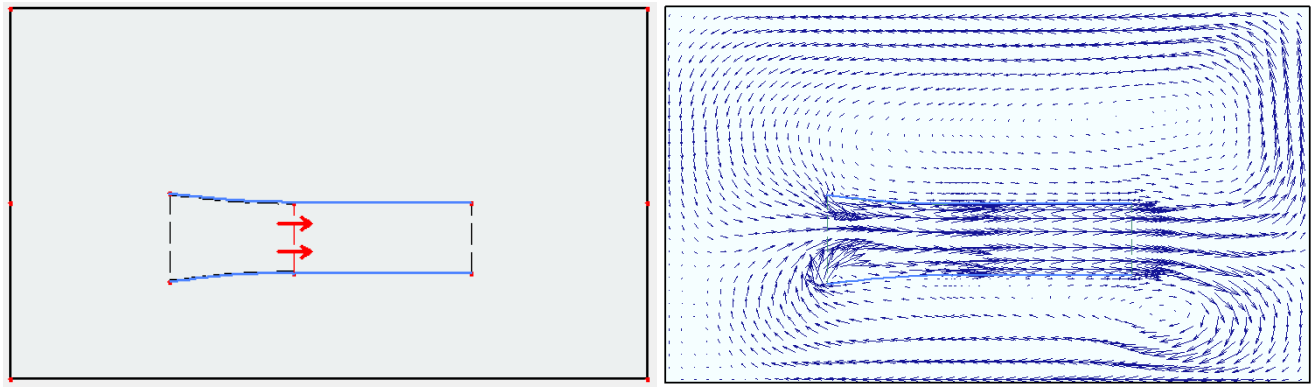
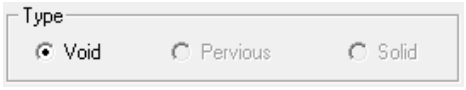


Figure B.64 – Utilisation of a pervious line region to impose fluid velocity inside the domain.

B1.6.7 – Void blocks



A void condition can only be assigned (and is the only choice) to unmeshed blocks. Void blocks (cf. Figure B.65) are impervious to the flow and may receive regions with the available boundary conditions: inlet, outlet, wall or symmetry.

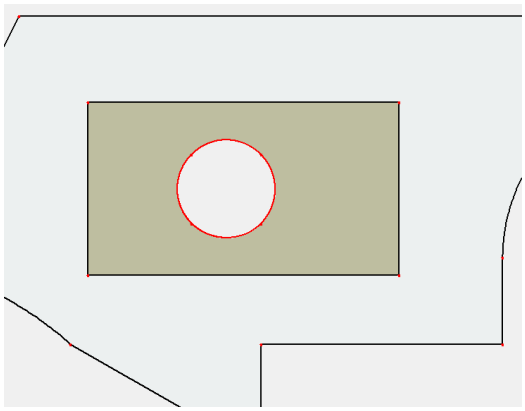


Figure B.65 – Example of a void block (delimited by the red line), inside a solid block (filled in gray).

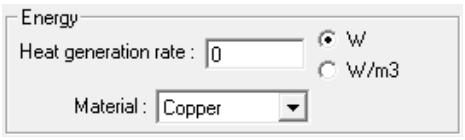
B1.6.8 – Pervious blocks

Pervious blocks are regions that are pervious or partially pervious to the flow (pervious to momentum). They can be used with the following purposes:

- For including porous regions in the simulation (cf. Section B1.6.13);
- for imposing a volumetric *Heat generation rate*, by specifying the total heat generation rate (*W*) or, alternatively, the heat generation rate per unit volume (*W/m3*) (the third dimension is considered as unit);
- for imposing a volumetric *Generation rate* of passive scalars, by specifying the total generation rate (*kg/s*) or, alternatively, the generation rate per unit volume (kg/m^3s);
- to create a space region for defining particular initial conditions, as for instance, a certain temperature (cf. section 0);

- to create a space region for the quantitative analysis in the post-processing phase (cf. section B3.7).

B1.6.9 – Solid blocks



This type of condition is only available for meshed blocks in non-isothermal problems, to compute heat transfer by conduction. The solid *Material* for the block must be specified as well as the internal *Heat generation rate*. Additional regions can be defined at the block boundaries.

B1.6.10 – Pervious line blocks and thin plates

Cf. example file *SpliMesh.def*

Open blocks are, by default, pervious to the flow (they are called *pervious line blocks*). Nevertheless, these blocks can be used to simulate thin plates, by creating, in some of their geometric elements, regions of wall type, such as exemplified in Figure B.66.

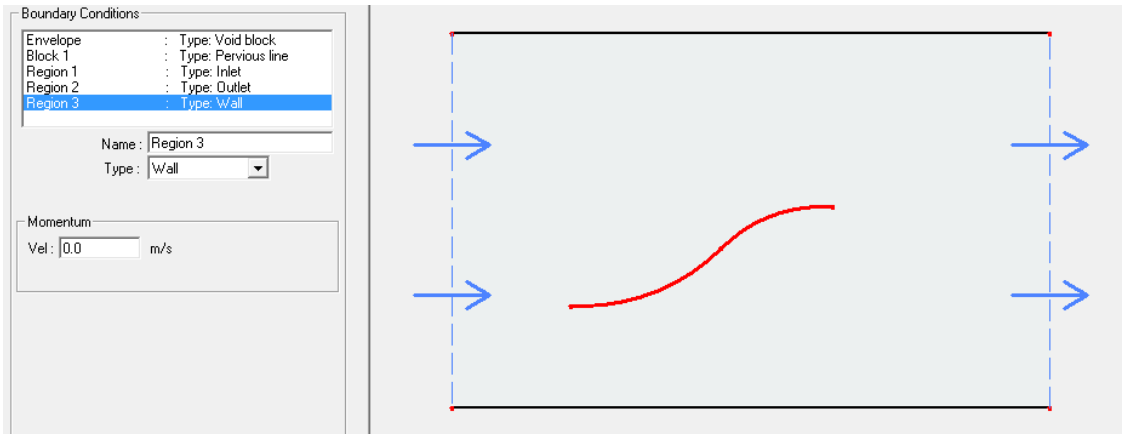
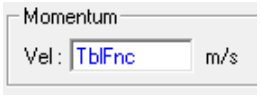


Figure B.66 – Utilization of a pervious line block to create a thin plate.

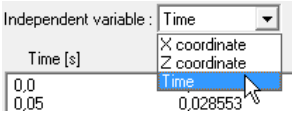
B1.6.11 – Dummy boundary condition

A dummy boundary condition has no special effect on the problem physics and is used only to create a region identifiable for the post processing purposes, such as, for instance, to compute a heat flux or any statistical quantity. It can be applied to any element of any block, provided it has no other boundary conditions already assigned. Due to its character, no values are necessary to be defined for this type of boundary.

B1.6.12 – Space/Time dependent boundary values: *TableFunctions*

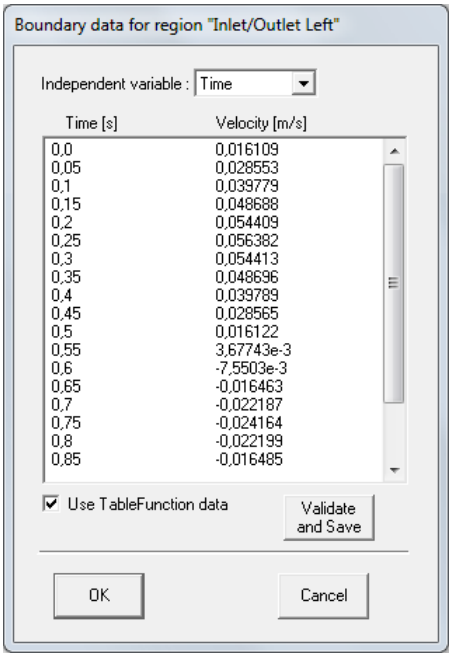


Boundary values can be either constant (when a numerical value is specified) or space/time dependent. Space or time dependency is specified through a table, listing the *Independent variable* (*X*, *Z* or *Time*) and the dependent variable values, as show on the right.



To open the *TableFunction DialogBox*, press the **F5** key when the corresponding input *TextBox* has the focus. The values should be listed in two

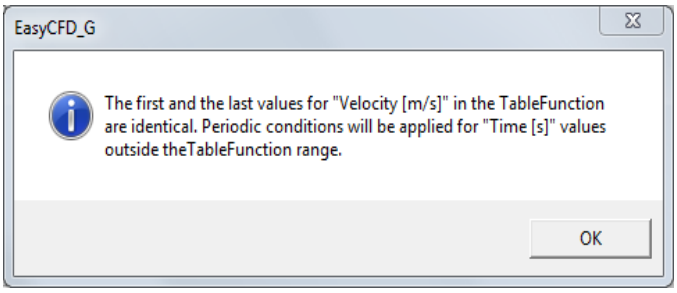
columns separated by, at east, a blank space or a tab (note: you may also directly paste values from a text file or from other software such as *Microsoft Excel*).



For a space-dependent profile, you may choose either *X* or *Z* as the independent variable. The proper choice should be made based on the boundary inclination: of course, for example, the *X* dependency should be the choice for an horizontal or nearly-horizontal boundary. *TableFunction* data may be validated to check for any errors by pressing *Validate and Save*. By pressing *OK*, the *TableFunction* data will be saved associated with the boundary; nevertheless, this data will be effectively used only if the *Use function table data* is checked. If a boundary value is given by a table function, its input *TextBox* will show *TblFnc*, as depicted on the right. To override this and use a single value, you may either open the *TableFunction DialogBox* (by pressing **F5**) and unchecking the *Use TableFunction* data button or, simply by entering a numeric value in the input *TextBox*.

TableFunction data is limited to *50* values. Values are computed from *TableFunction* data using linear interpolation.

If the boundary independent variable value (*X*, *Y* or *Time*) falls outside the *TableFunction* range, two possible situations may be considered:



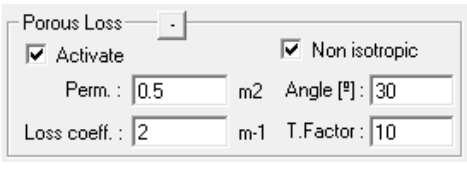
a) If the first and the last dependent variable values in the *TableFunction* are similar, periodic conditions are assumed (cf. example in section C5).

b) If no periodic conditions are assumed, the boundary value will be similar to the closest value in the *TableFunction* range.

Integral quantities such as mass flowrate, volume flowrate and volumetric heat generation rate cannot be space-dependent (thus, only *Time* can be selected and the independent variable).

TableFunctions are not available for the following boundary variables: External temperature and heat transfer coefficient (cf. section B1.6.1); values for passive scalars (either concentration or flux).

B1.6.13 – Porous losses

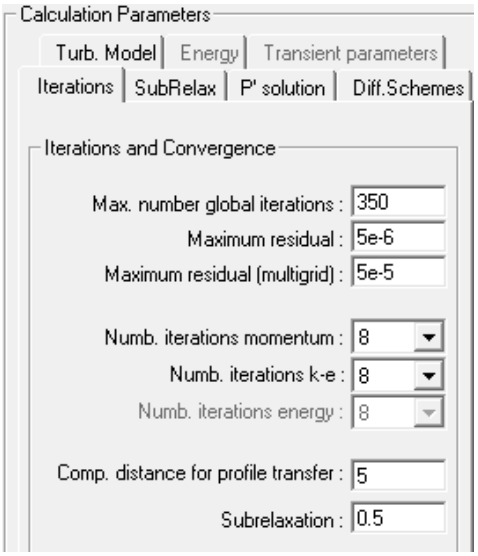


Momentum losses through porous regions may be assigned to pervious blocks. Input data are the viscous permeability coefficient (***Perm***; K_p in section A1.5), the quadratic loss coefficient (***Loss coeff***; *C*) and, for ***non isotropic*** porous media, the streamwise direction ***Angle*** (φ) and transverse factor (***T. Factor***; F_t).

B1.8 – Calculation Parameters

This section allows the specification of all parameters concerning the solution of the equations and convergence control.

B1.8.1 – Iterations and Convergence



Max. number global iterations is the maximum number of outer iterations allowed in the *SIMPLEC* algorithm. When this number of iterations is reached, the calculation is stopped even if the convergence criteria were not attained.

Maximum residual is the parameter R_{conv} (cf. equation (4.4)) that establishes the maximum normalized residual as convergence criterion.

Maximum residual (multigrid) establishes the criterion to change meshes during a multigrid calculation (cf. B2.1.3).

Numb. Iterations momentum is the fixed number of cycles used to solve the momentum equations within each *SIMPLEC* iteration. Allowed values are *4*, *8* or *16*.

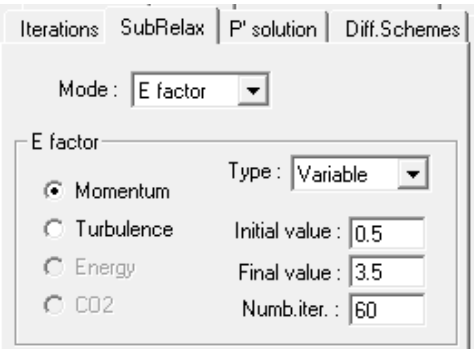
Numb. iterations k-e is the fixed number of cycles used to solve the *k* and the ε equations within each *SIMPLEC* iteration. Allowed values are *4*, *8* or *16*.

Numb. Iterations energy is the fixed number of cycles used to solve the energy equations within each *SIMPLEC* iteration. Allowed values are *4*, *8* or *16*. The same values are used for the concentration of the secondary fluid, in multi-component fluid flow problems

Comp. distance profile transfer is the computational distance, downstream of the inlet(s), where the velocity profile(s) is taken for transposition (cf. section B1.6.2). A *Subrelaxation* factor is applied, to prevent eventual instabilities during the calculation.

B1.8.2 – Sub-relaxation

There are two options for subrelaxation: using a local subrelaxation through the *E* factor, as described in section A2.3, or adopting a transient-like approach, where the solution advances in time, starting from the initial variables field.

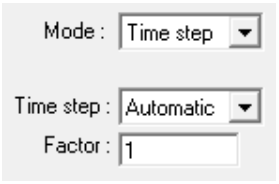


The E-factor option generally leads to a faster converges rate, but is some cases may lead to higher numerical instabilities. The transient-like approach (Time step option), provides a smoother convergence, but may require more iterations.

The *E* factor may be specified, independently, for the *Momentum*, *k-e Energy* and secondary fluid concentration (*CO2*, in the case depicted) equations. The *Type* of variation controls how *E* values depend on the global iteration number:

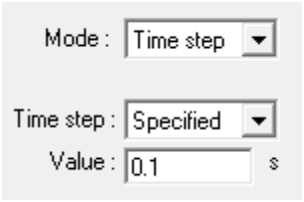
Fixed: the *E* factor is constant throughout the iteration process.

Variable: the *E* factor starts with an *Initial value*, increases linearly during the number of iterations specified in *Numb. iter.* up to the maximum specified in *Final value* and remains constant during the remaining iterations. With this method, the subrelaxation is higher at the beginning of the iterations (lower *E* values), which is an advantage to stabilize the calculation. As the calculation approaches convergence, less subrelaxation is needed and the *E* factor can be higher. Typical values are 0.5 to 3.5 during 60 iterations.



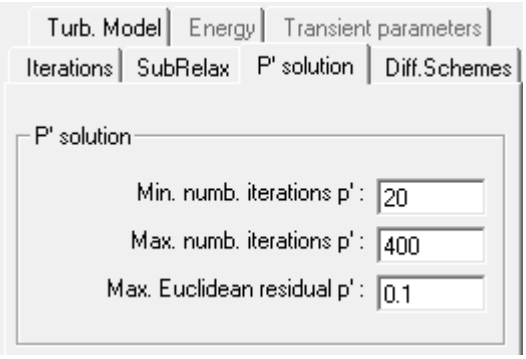
For the *Time step* option, the time step may be computed automatically based on the characteristics problem length and velocity scales. This value may also be affected by a specified factor.

Alternatively the user may specify the time step. In both cases, during the calculation, the automatic time step value is displayed.



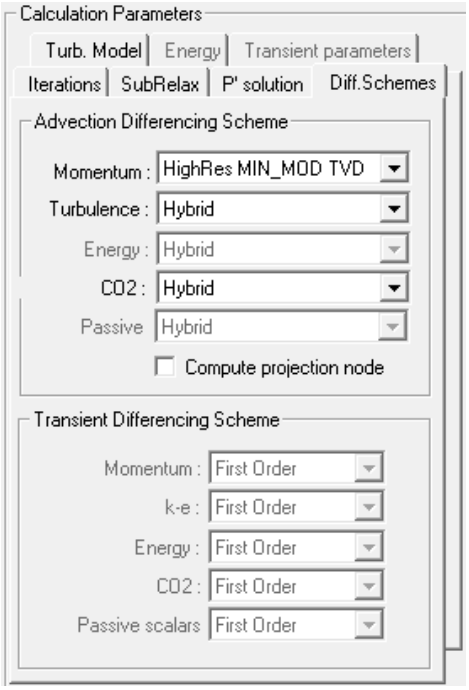
B1.8.3 – The *p'* solution

As already stated, the *p'* solution needs special care. The user may choose the minimum (*Min. numb. Iterations p'*) and the maximum (*Max. numb. Iterations p'*) number of iterations (cycles) for its solution. The *Max. Euclidean residual p'* is described by equations (4.2) and (4.3).



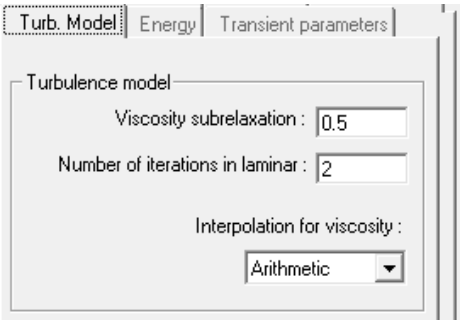
B1.8.4 – Differencing schemes

The differencing scheme for advection (cf. section A2.2.3) and for time integration (cf. A2.2.2) may be chosen independently for each type of equation, as shown on the right. Computation of the projection location of the upstream node for higher order schemes is optional. This is computationally quite expensive and is undertaken only when residuals are near convergence criterion. If not selected, the upstream control volume center is considered.



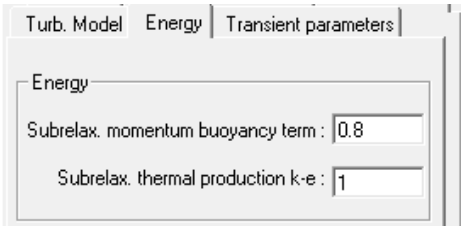
B1.8.5 – Parameters for the turbulence model

A subrelaxation is applied for the calculation of the turbulent viscosity, using an equation like (4.1), as a means of stabilizing convergence (*Viscosity subrelaxation*). Furthermore, it may be convenient to introduce the turbulence model only after performing some iterations in laminar regime (*Number of iterations in laminar)*. The *Interpolation for viscosity* may be set to *Arithmetic* or to *Harmonic* (cf. equations (2.39) and (2.41)).



B1.8.6 – Parameters for the energy equation

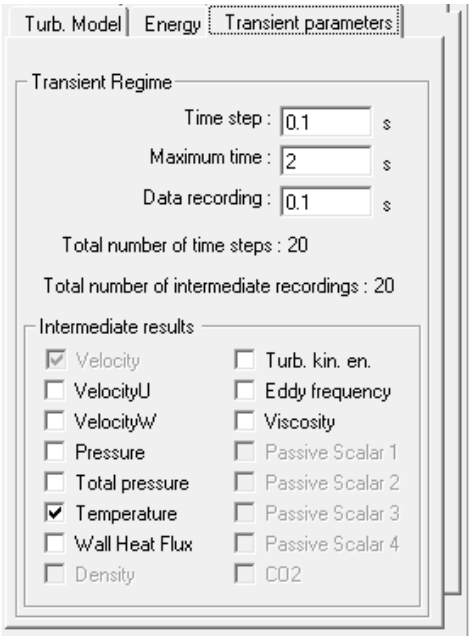
A subrelaxation factor is provided for the buoyancy term in the momentum equations (*I* in equations (1.3) and (1.3b)), and for the thermal production terms of the *k-ε* model equations (G_T in equations (1.23) and (1.24)). This may be useful in situations of high Rayleigh number (large temperature amplitude).



B1.8.7 – Parameters for transient problems

For transient problems, the time increment (*Time step*) and the *Maximum time* for the simulation must be specified. The temporal discretisation errors and numerical stability are directly related with the time step and, consequently, this parameter should be carefully chosen.

For transient problems, the *Max. number global iterations* defined in section B1.8.1 refers to the number of iterations within each time step. In this case, *5* to *10* iterations should be enough to meet the convergence criteria, otherwise, most likely the chosen time step is too large.

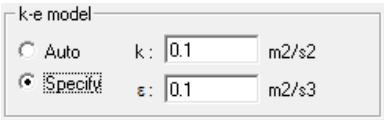


The parameter *Data recording* establishes the time interval between two successive data recordings (intermediate results). Since the total number of files may easily become be very large, the user can select which variables should be stored, so as to minimize storage requirements. The available variables depend on the problem physics; for example: if the Boussinesq approximation is adopted, density is constant within the domain and, thus, this variable won't be available for the intermediate results. As mentioned in section *B1*, intermediate results are stored in the sub-folder *Trans* of the project folder.

B1.9 – Initialisation of Variables

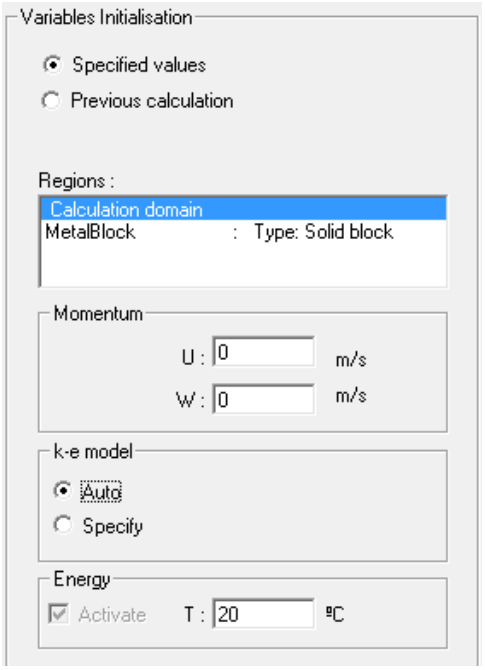
A problem may be initialized in two different ways:

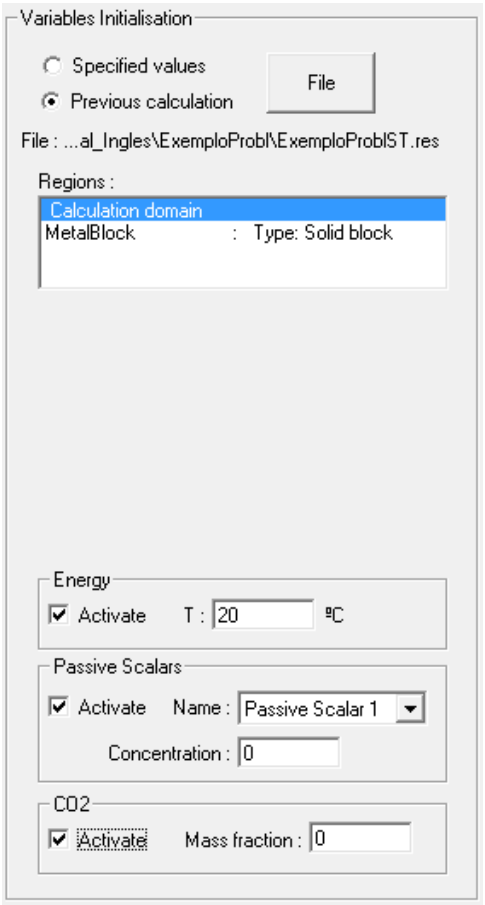
1. *Specified values:* the values entered in the *TextBoxes* shown at right are assigned to the selected region (*Regions*). Selectable regions are the *Calculation domain* (default), pervious blocks and solid blocks. For the turbulence model, you may choose an *Auto*matic initialisation, in which case an average of the inlet values are considered, or, alternatively, *Specify* the initialisation values for k and ε.



2. *Previous calculation*: if you have a previous calculation, you can use the stored values to initialize the problem (this is equivalent to select *Project – Load Results*, as described in section B1.1). This is an useful option for:

   - Post-process the results from a calculation already made. In this case, after specifying the file name, you can go directly to the *Post-processing* phase.
   - To solve a transient problem, starting from a previous steady-state calculation.

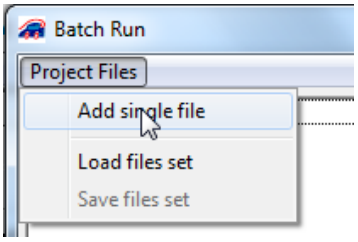


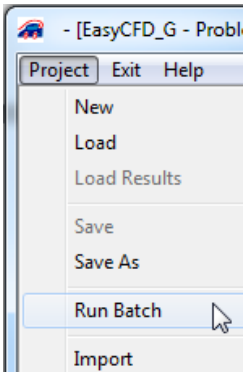


The three *ToggleButtons Activate* allow the corresponding variables to be imposed in the selected regions, overriding the values stored in the initialisation file.

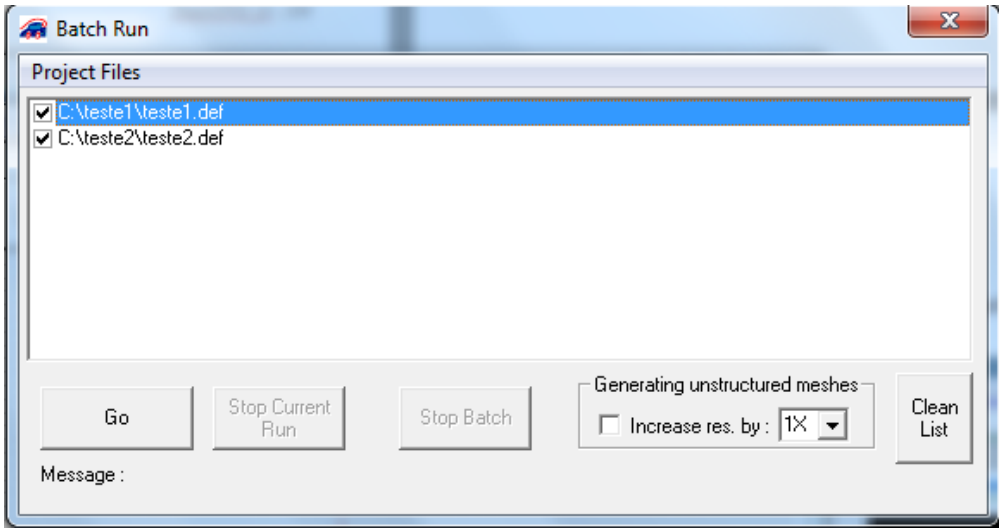
B1.10 – Running in batch

Several pre-prepared *.def* files can be run using the *Run Batch* feature. In the corresponding Dialog Box, one may add a single *.def* at a time, or load a set of files specified through their path in a text file. Only the files listed with a check mark will be run.





For each file, the mesh is generated only if the *.def* file does not contain mesh information. For unstructured meshes, by activating the *Increase res. by*: check button, the mesh size is increased the specified number of times, using the feature described in section B1.4.5.



B2 – Solver

Figure B2.1 displays the graphical interface for the *Solver* phase. Several spaces are identified for easier reference.

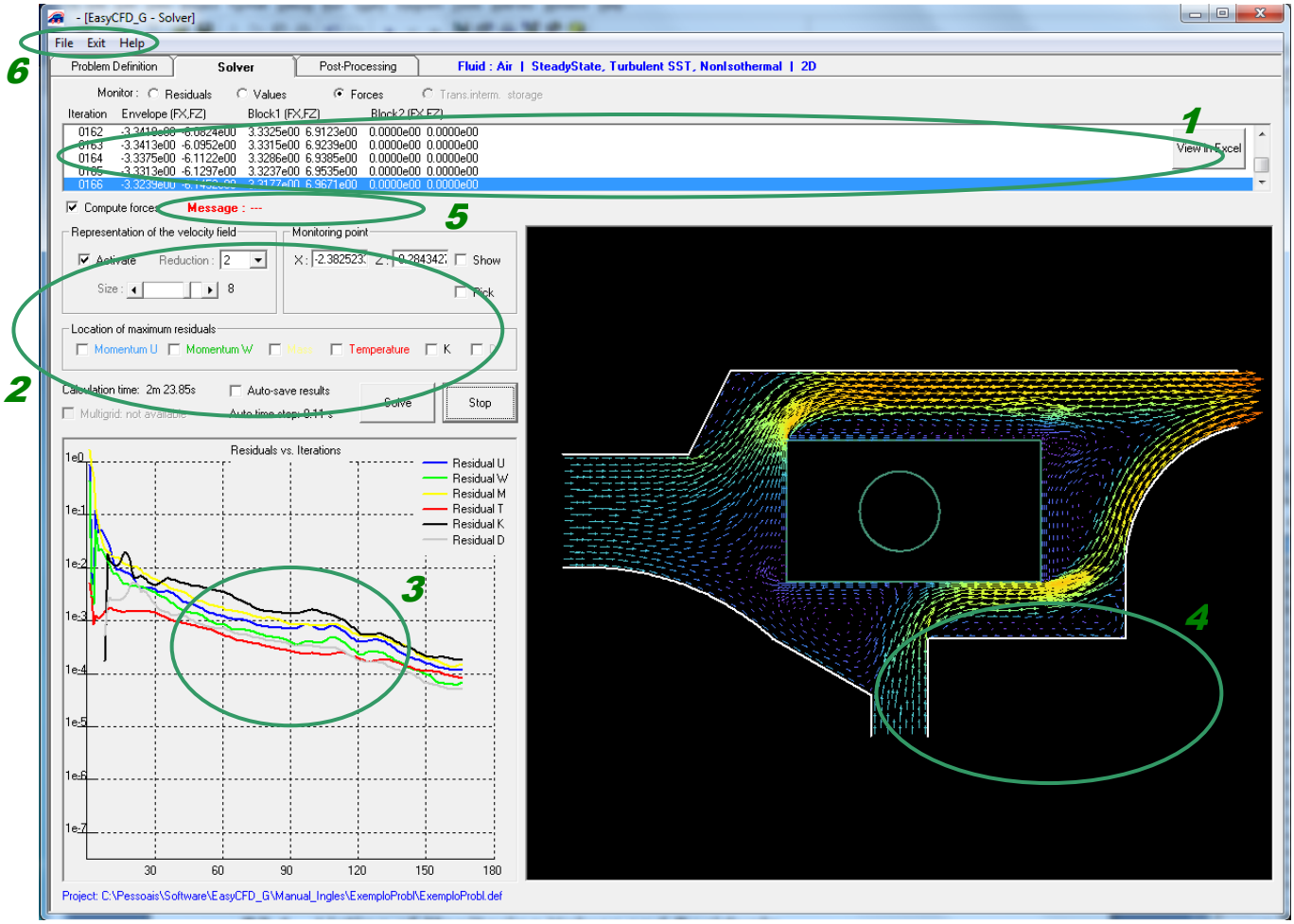
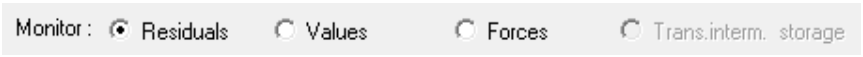


Figure B.67 – Graphical interface for the solver phase.

B2.1 – Listing of Monitoring Values and Residuals

Space 1 presents diverse information concerning the iterative process, depending of the selection made in *Monitor*:



B2.1.1 – Steady-state problems

Residuals: The first column displays the iteration number. The next seven columns display the normalized residuals for each equation and the number of cycles employed for the solution of the *p'* equation as well as its final normalized Euclidean residual. (γ_r, in equation (4.3)).

Iteration	Residual U	Residual W	Residual M	Residual K	Residual D	Residual T	Sec. Fluid	Nswp p'	Euclidean
0001	0.0000e00	0.0000e00	1.5729e00	0.0000e00	0.0000e00	1.9499e-04	########	404	2.2958e-01
0002	7.2641e-04	3.5224e-04	1.0265e00	0.0000e00	0.0000e00	2.1174e-04	########	404	2.1133e-01
0003	1.3693e-03	6.0952e-04	5.2652e-01	0.0000e00	0.0000e00	2.3910e-04	#########	404	1.9529e-01
0004	3.9878e-03	1.5641e-03	2.5331e-01	0.0000e00	0.0000e00	2.7211e-04	#########	404	2.0427e-01
0005	9.1770e-03	3.3084e-03	1.3531e-01	0.0000e00	0.0000e00	3.1110e-04	#########	404	2.2945e-01

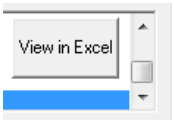
Values: The first column displays the iteration number. The next six columns display the values of the variables at a spatial location (monitoring location) selected by the user.

Iteration	Monitor U	Monitor W	Monitor K	Monitor D	Monitor T	Sec. Fluid
0001	2.6487e-02	2.9658e-03	3.7500e-03	2.0668e-04	2.0000e01	0.0000e00
0002	1.8024e-01	8.7585e-04	3.7500e-03	2.0668e-04	2.0000e01	0.0000e00
0003	3.7683e-01	-2.4762e-03	3.7500e-03	2.0668e-04	2.0000e01	0.0000e00
0004	5.9748e-01	-6.9304e-03	3.7500e-03	2.0668e-04	2.0000e01	0.0000e00
0005	8.0684e-01	-1.2009e-02	3.7500e-03	2.0668e-04	2.0000e01	0.0000e00

Forces: Forces on walls are computed at every iteration if the button *Compute forces* is checked.

Iteration	Envelope (FX,FZ)		Block1 (FX,FZ)	
0009	0.0000e00	0.0000e00	2.6382e01	-3.6618e-01
0010	0.0000e00	0.0000e00	2.9118e01	8.7687e-02
0011	0.0000e00	0.0000e00	2.7466e01	6.1028e-01
0012	0.0000e00	0.0000e00	1.4822e01	7.3382e-01
0013	0.0000e00	0.0000e00	2.7898e00	3.4237e-01

Forces values listed in the monitor can be sent to Excel, by pressing the corresponding button. If the selection at the listbox is not the latest value, the Excel will contain only the values starting at the selection; otherwise, all values are sent to Excel.



B2.1.2 – Transient problems

For transient problems, the displayed residuals correspond to the iteration prior to the time instant transition. The iteration (first column) represents the total number of iterations performed within the time instant. The last column (for both *Residuals* and *Values*) displays the current time instant.

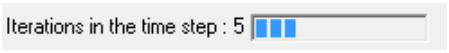
Iteration	Residual U	Residual W	Residual M	Residual K	Residual D	Residual T	Sec. Fluid	Nswp p'	Euclidean	Time instant
0005	1.6815e-04	2.4451e-03	1.5311e-01	########	########	9.6504e-06	########	036	8.6844e-02	0.1
0005	3.8060e-05	2.0186e-04	2.0480e-02	########	########	2.3627e-06	########	020	9.9706e-02	0.15
0005	1.1233e-05	5.8416e-05	6.0951e-03	########	########	1.3533e-06	########	036	9.6749e-02	0.2
0005	7.7573e-06	3.2407e-05	2.0918e-03	########	########	1.0338e-06	########	056	9.1458e-02	0.25
0005	4.5927e-06	1.8184e-05	8.6274e-04	########	########	8.1313e-07	########	056	8.9516e-02	0.3

Intermediate storage for transient problems:

Selecting *Trans.interm.storage*, the time instant and the corresponding file name for intermediate results storage are displayed.

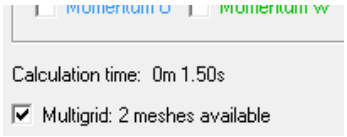
Time instant	Intermediate results stored in:
0.0	C:\Software\EasyCFD_G\Exemplos\CHT\Trans\Tr0.res
0.1	C:\Software\EasyCFD_G\Exemplos\CHT\Trans\Tr1.res
0.2	C:\Software\EasyCFD_G\Exemplos\CHT\Trans\Tr2.res
0.3	C:\Software\EasyCFD_G\Exemplos\CHT\Trans\Tr3.res
0.4	C:\Software\EasyCFD_G\Exemplos\CHT\Trans\Tr4.res

For a better control, a horizontal indication bar in *Space 2* shows the progress within each time step, relative to the maximum number of iterations specified.

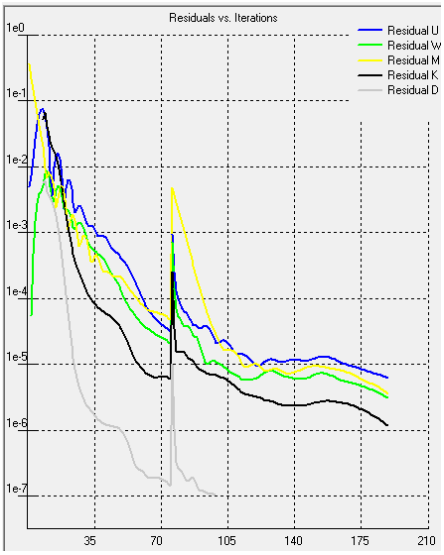


The indexation of the monitoring point is chosen in the interface block shown at right (*Space 2*). When *Show* is checked, the location of the monitoring point is represented in *Space 4* by a small circle. The monitoring point may be picked in the representation using the mouse, if *Pick* is checked.

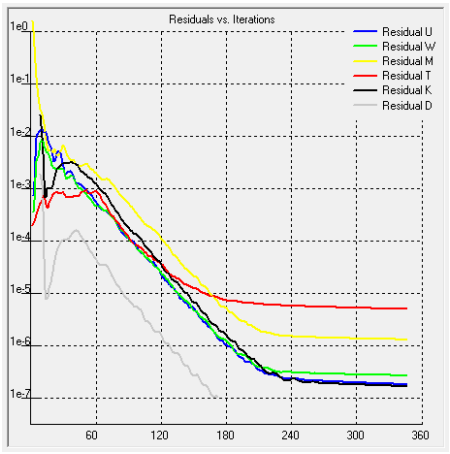
B2.1.3 – Multigrid calculation



Multigrid calculation is available for unstructured meshes, in steady-state problems, provided mesh resolution was increased using the linear interpolation method (cf. B1.4.5). This allows much faster calculations for large problems. Calculation starts with the base mesh (coarser mesh). When the limiting residual for mesh transition is reached (cf. B1.8.1), results are interpolated into the next finer mesh and calculation proceeds. If other finer meshes are available, the process is repeated. It is normal that transition between meshes leads to a jump in the residuals (as shown at right).



B2.2 – Graphical Representation of the Residuals

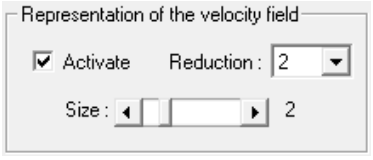


The graphics with the normalized residuals (cf. equations (4.5) and (4.6)) are presented in *Space 3*. The horizontal axis represents the iteration number, using a linear scale, while the normalized residuals are in the vertical axis, in logarithmic coordinates. A different colour is assigned to each variable, according to the displayed key.

For transient problems, the horizontal axis shows the time instant counter.

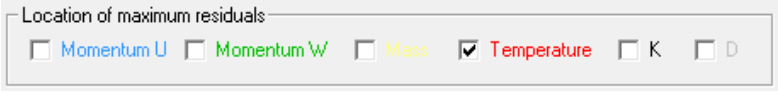
The evolution of the residuals is very much problem dependent, In the presented case, the solution presents a good convergence, though in the final stage the convergence rate is slow.

B2.3 – Vectorial Representation of the Velocity Field



The evolution of the flow field during the solution of the problem may be visualised in *Space 4*, with a vectorial representation of the velocity field (this may slow down the calculation, specially for large problems) . The controls shown on the right allow the user to adjust display settings, such as the vectors *Size* and its quantity (*Reduction*).

B2.4 – Location of Maximum Residuals



The location where the maximum residuals are verified may provide valuable information for identifying the origin of problems related to low convergence or even divergence of the iterative process. By checking

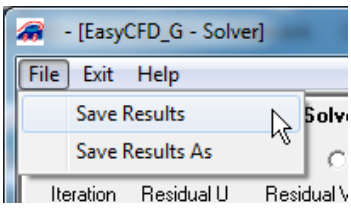
for each equation, the respective location of maximum residual is displayed with a small circle in *Space 4*.

B2.5 – Information messages

Whenever necessary, *Space 5* displays information messages, such as the current storage file for intermediate transient results, etc.

The time elapsed since the start of the computation is shown in *Space 2*

After the solver is finished, the results may be saved in a file with the default name (project name with extension .*Res*) using *Save Results*, or in a file with a chosen name (*Save Results As*). Results may, alternatively, be stored when **EasyCFD** is exited.



B3 – Post-Processing

B3.1 – Example Problem

The illustration of the post-processing capabilities will be done recurring to the example shown in Figure B.68.

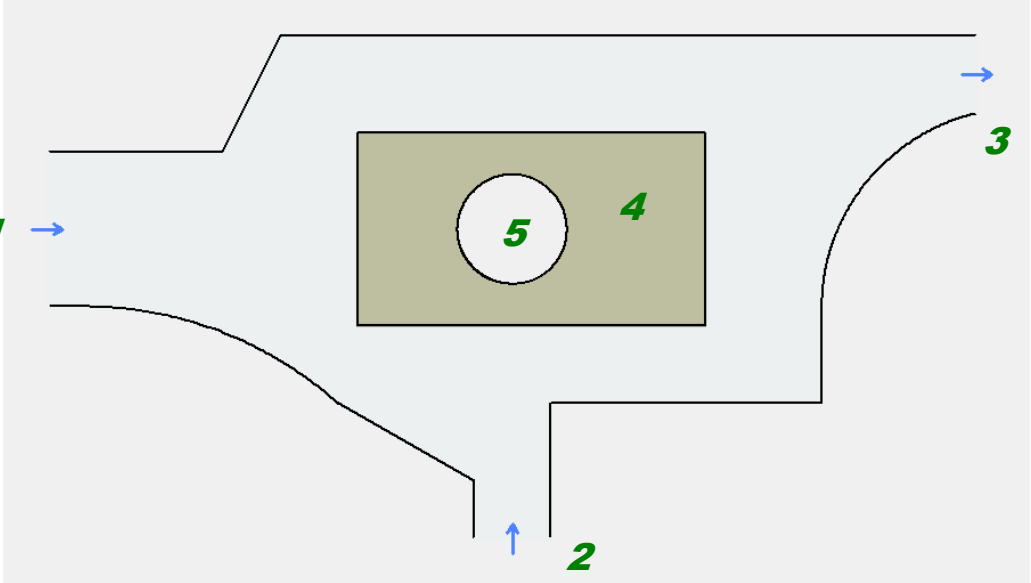


Figure B.68 – Demonstration problem.

Here is some data for this problem:

1: Region type: *Inlet*
Velocity: *1 m/s* ; *θ*: *0º*
Temperature: *20 ºC*
Turbulence intensity: *5%* ; Characteristic length: *0.01m*

2: Region type: *Inlet*
Velocity: *1 m/s* ; *θ*: *0º*
Temperature: *30 ºC*
Turbulence intensity: *5%* ; Characteristic length: *0.01m*

3: Region type: *Conservative Outlet*

4: Region type: *Solid block*
Material: *Copper*
Total heat generation rate: *0 W*

5: Region type: *Void block*

The envelope walls are adiabatic. The total horizontal dimension of the domain is *2.4 m*.

B3.2 – General Description

The post-processing phase is available only after the solver is finished or calculation data was loaded. A general view of the graphical interface is presented in Figure B.69.

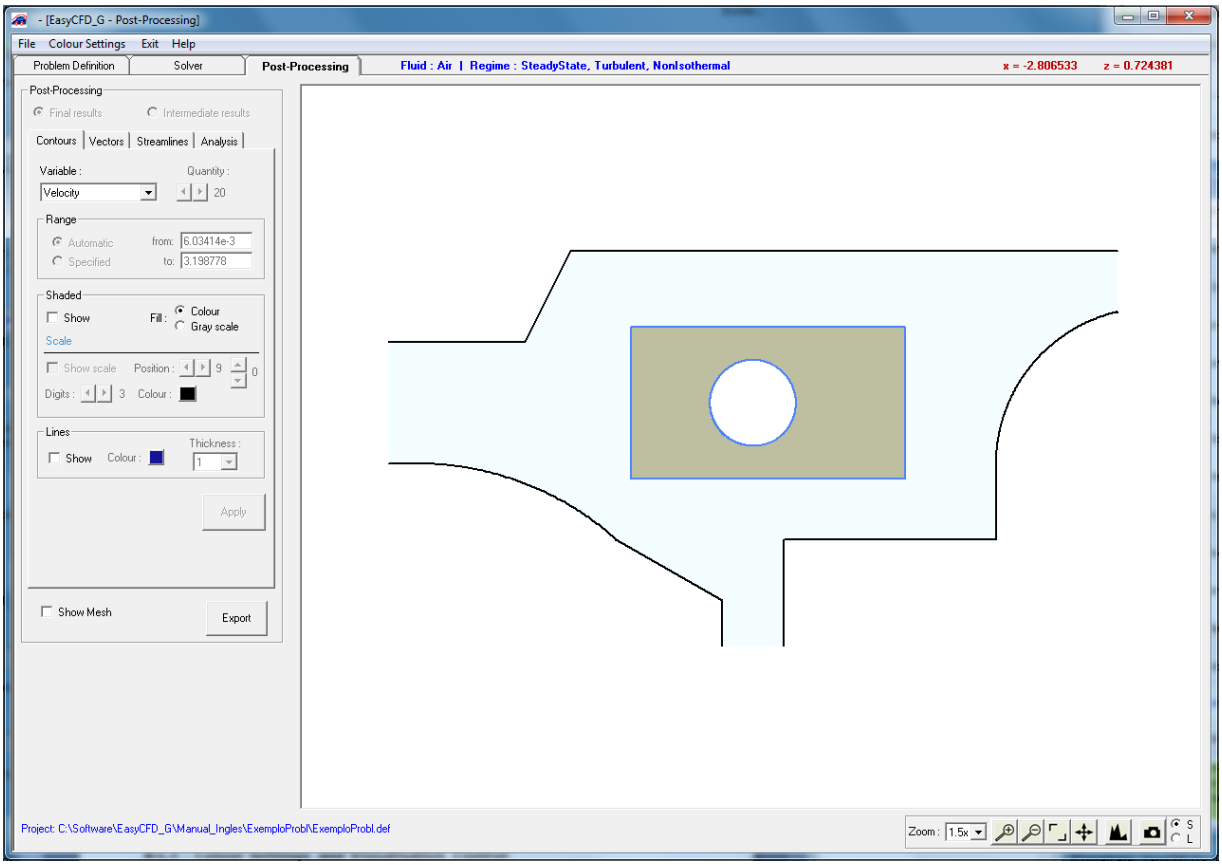
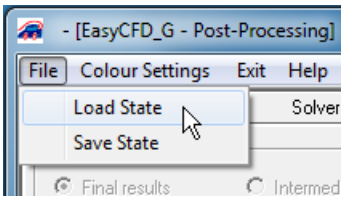
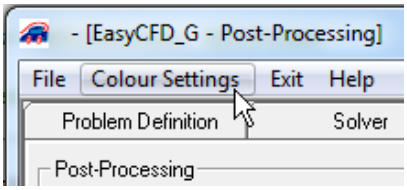


Figure B.69 – Graphical interface for the *Post-Processing* phase.

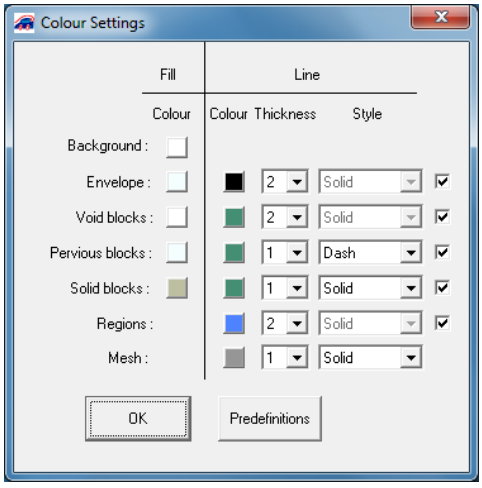
B3.3 – State File, Colour Settings and Visualisation Control



The settings for the present visualization may be saved in a file (*Save State*), so that they can be recovered later on (*Load State*). These include all the visualization definitions, such as zoom settings, type of visualization, contour values, etc.



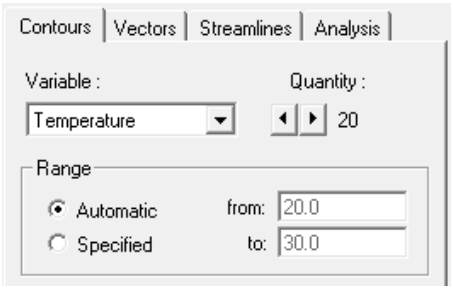
Within *Colour Settings*, you may choose the *Colour* for filling 2D regions (*Background*, *Envelope*, different types of blocks), and the *Colour*, *Thickness* and *Style* for different lines, as shown on the right. The *CheckButtons* allows you to switch on or off the display of the different entities.



The selected definitions are saved with the project. Note that styles other than *Solid* are only available for line thickness *1*.

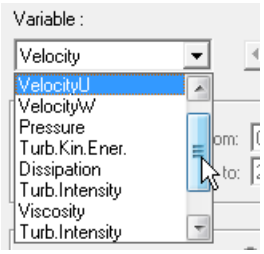
Most of visualisation control is performed in a similar manner as described in section B1.1. The post-processing, nevertheless, features the additional functionality of fitting the whole geometry, not only in the visible windows (option *S*), but also in an extended window 9 times larger (option *L*). The tool  allows you to capture the visible window (option *S*) or the extended window (option *L*) and save the corresponding image in the *.jpg* file format.

B3.4 – Contours



Contours, either shaded contours or line contours, represent the spatial variation of scalars (cf. example in Figure B.70). The *Variable* to be represented and the *Quantity* of contours are the first selection to be made.

Available variables depend, naturally, on the physics involved. Fluid flow problems (*i.e.*, non-pure conduction problems) include *Velocity*, *VelocityU* (*x* component of velocity), *VelocityW* (*z* component), *Pressure* and the *Stream Function* ψ defined as:



$$u = \frac{\partial \psi}{\partial z} \quad ; \quad w = -\frac{\partial \psi}{\partial x}$$

Non-isothermal problems include *Temperature*. If buoyancy effects are present, *Total pressure* (p_{tot}, which takes into account the hydrostatic contribution), is available:

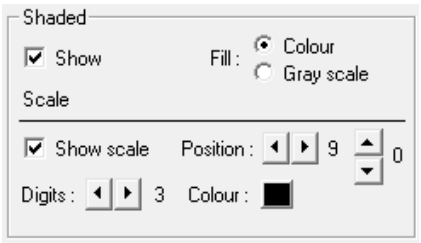
$$p_{tot} = p - \rho g z$$

For turbulent problems, turbulent kinetic energy (*Turb.Kin.Ener*), its rate of *Dissipation*, *Viscosity*, turbulence intensity (*Tub. Intensity*) and y^+ (cf. equation (1.28)) are available.

Choices for *Range* are:

- *Automatic*: the contours are displayed using all available range for the chosen variable.
- *Specified*: the minimum (*from*) and the maximum (*to*) contour values are chosen by the user. This option is particularly useful for analysing a particular region of the domain, for example.

B3.4.1 – Shaded contours



The representation of shaded contours is activated in the corresponding *CheckButton Show*. Shaded contours can be displayed either in *Colour* or using a *Gray scale*. If y^+ is the chosen variable, the graphical representation is done only for control volumes adjacent to wall-type boundaries.

The *Show scale* button activates the representation of the correspondence between colours (or gray shades) and values range. The scale horizontal and vertical *Position* relative to the visualisation window can be adjusted in the arrow buttons. The number of significant *Digits* for the labels and their *Colour* can also be adjusted.

B3.4.2 – Line contours

Line contours are displayed with a constant *Colour* using the line *Thickness* chosen by the user. This type of representation is not available for y^+.

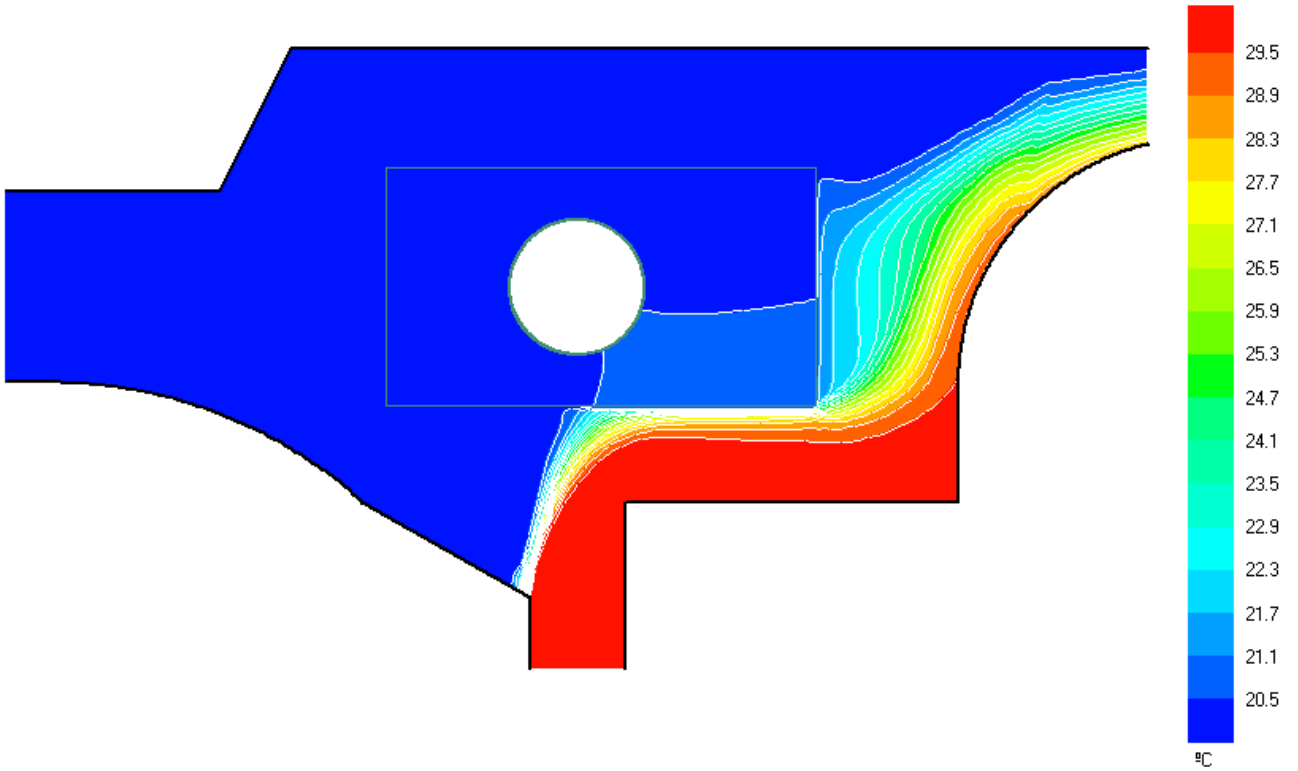
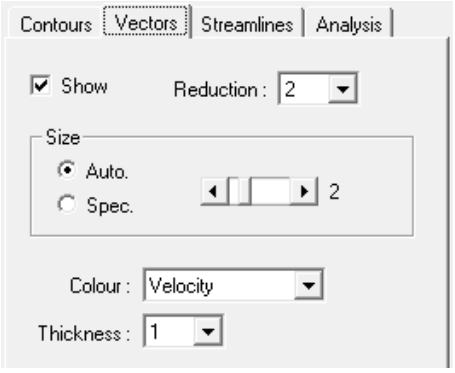


Figure B.70 – Temperature line contours (white lines), shaded contours and corresponding scale for the example problem.

B3.5 – Vectors

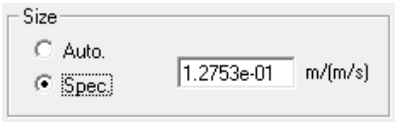
The flow velocity at the centre of each control volume is represented with a vector, the length of which is proportional to the velocity magnitude.



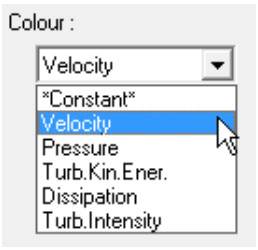
The *Size* can be:

- *Auto.*: a proportionality factor between vector length and velocity magnitude is automatically adjusted. This factor may be modified by the user, by sliding the horizontal bar in the command block.

- *Spec.*, the ratio between vector length and velocity magnitude (given in *m/(m/s)*) may be directly specified by the user. This option is quite useful if one intends to compare different velocity fields and the same scale factor is desired.



The command *Reduction* imposes a reduction for the number of vectors displayed, so as to achieve a better visualisation.



The vectors *Colour* may be:

- **Constant*: the user chooses the desired *Colour* for the vectors.
- Variable: each vector colour depends on the local value of the chosen variable.

The line *Thickness* for the vectors can also be chosen.

Figure B.71 shows a representation of the vectorial velocity field using the local velocity as criterion for the vectors colour.

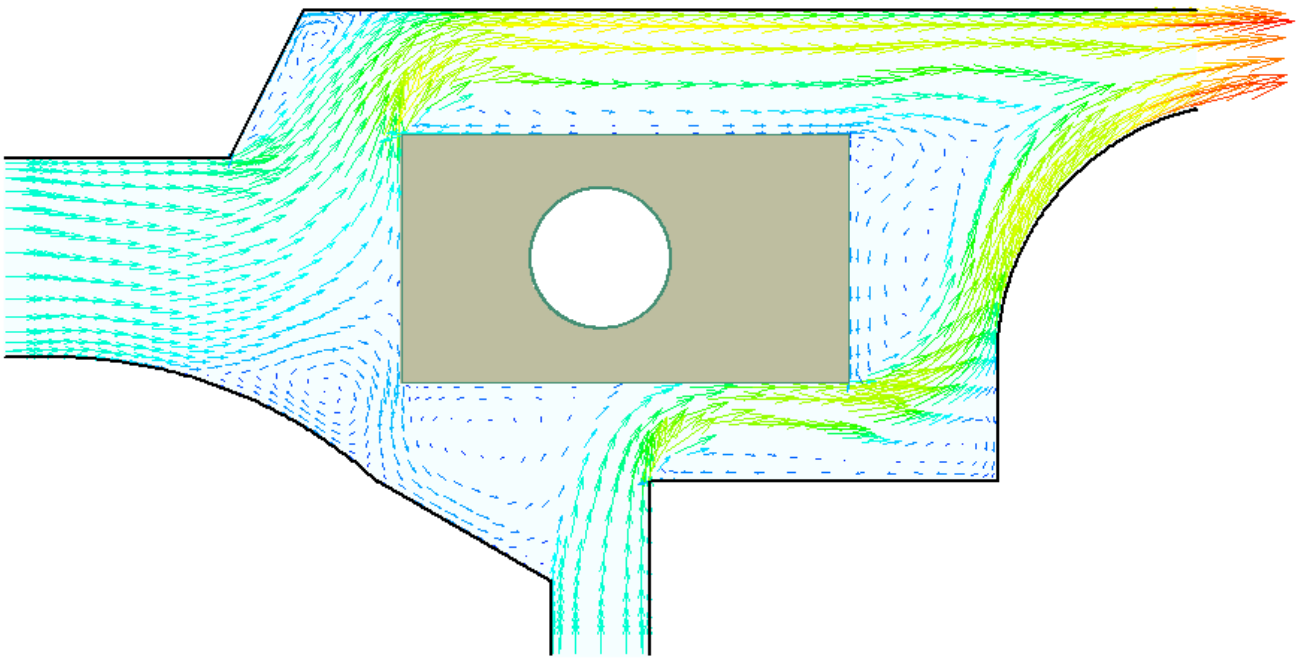
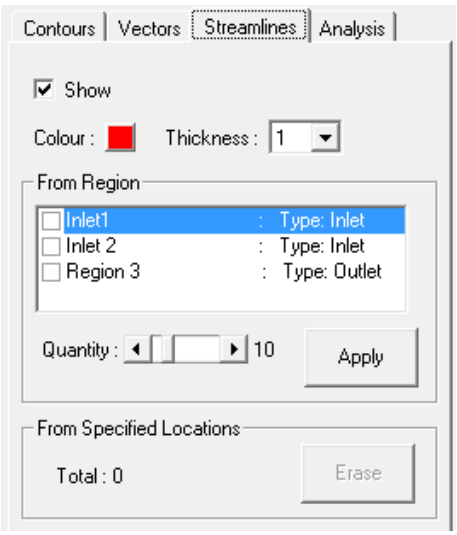


Figure B.71 – Representation of the vectorial velocity field.

B3.6 – Streamlines

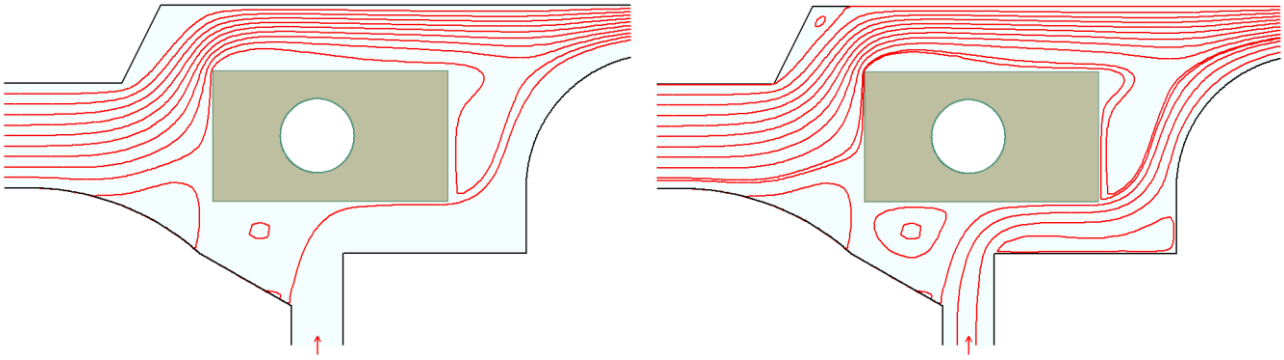
Streamlines connect locations with equal value of the stream function, representing the fluid trajectory in steady-state problems.

Streamlines can be represented using the available tools for the contours representation, as described in section B3.4.



Nevertheless, the streamlines *Tab*, described hereafter, provides additional functionality for the streamlines representation.

After checking *Show* and upon pressing *Apply*, the chosen *Quantity* of streamlines is drawn from the selected inlet or outlet boundaries, with the *Colour* and *Thickness* specified. Additionally, by using the mouse right button, you may select specific locations in the domain for displaying the streamline that passes through. An example may be seen in Figure B.72. The *Button Erase* clears the *Specified Locations* streamlines

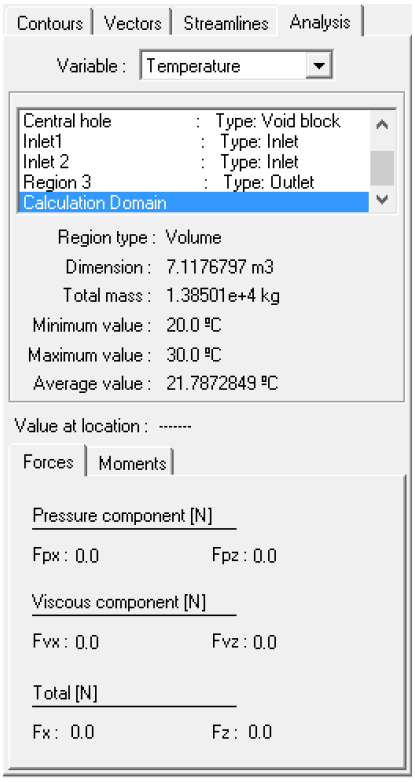


(a) (b)

Figure B.72 – Representation of streamlines.

(a) 10 streamlines starting from the left inlet. (b) addition of more streamlines with the mouse cursor.

B3.7 – Analysis

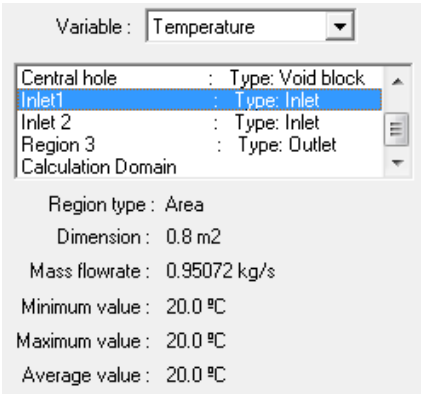


After selecting a *Variable* among the available ones, diverse information is displayed for the selected region.

Selectable regions are: the *Envelope*; a user defined region; a pervious or impervious block; or the *Calculation Domain*. After selecting the desired region, the *Maximum value*, *Minimum value* and *Average value* are displayed.

If the selected region is the *Calculation Domain*, a pervious block or a solid block, the *Dimension* has the units m^3 (the *Region Type* is *Volume*, as the third dimension is considered with length *1 m*) and the *Total mass* for this region is displayed. The maximum, minimum and average values are computed in that volume space, for all control volume centres. It should be noted that wall values do not enter in this calculation; this fact should be taken into account when interpreting the results when, for instance, the maximum value of the selected variable is located on a wall (this may happen for temperature, when the heat flux is imposed on the wall).

If other region type is selected (the *Envelope* or a boundary region, like inlets, outlets or walls), the corresponding dimension is given in m^2 (once again, the third dimension is considered with length *1 m*; the *Region Type* is *Area*) and the *Mass flow* through that region (zero, if a wall) is presented.



You may notice, when examining Figure A.8, that since variables are computed at the control volume centres, there are no available values exactly at the boundaries location. The extrapolation of variables values for the domain boundaries is performed considering the following criteria:

Outlet boundaries

The second derivative is zero along the direction perpendicular to the boundary, for all variables.

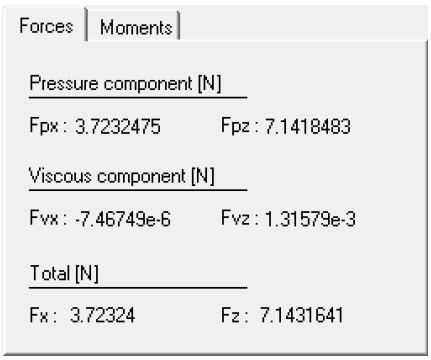
Symmetry boundaries

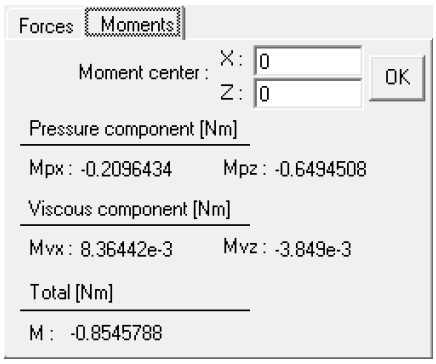
The first derivative is zero along the direction perpendicular to the boundary, for all variables.

Walls

- Velocity: the velocity component perpendicular to the wall is zero and the velocity component parallel to the wall has the same velocity as the wall.
- Pressure: the first derivative is zero.
- Turbulence kinetic energy: its value is zero.
- Dissipation rate of turbulence kinetic energy: the first derivative is zero.
- Temperature: if the heat flux is imposed, the wall temperature is computed based on the heat flux value, using equations (1.35) and (1.36).
- Other scalars: similar to temperature.

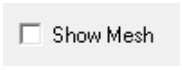
For boundary regions (*Area Type*), the acting forces, decomposed along the *x* and *z* direction, for both the pressure and viscous components, are presented. Moments computed relative to a selected location (moment center) are also available. Moments are defined as positive in the counterclockwise direction.





B3.8 – Representation of the Mesh

The representation of the mesh lines is activated with the *CheckButton Show Mesh*. The colour for the grid lines is defined in the *Dialog-Box Colour Settings* (cf. section B3.3).



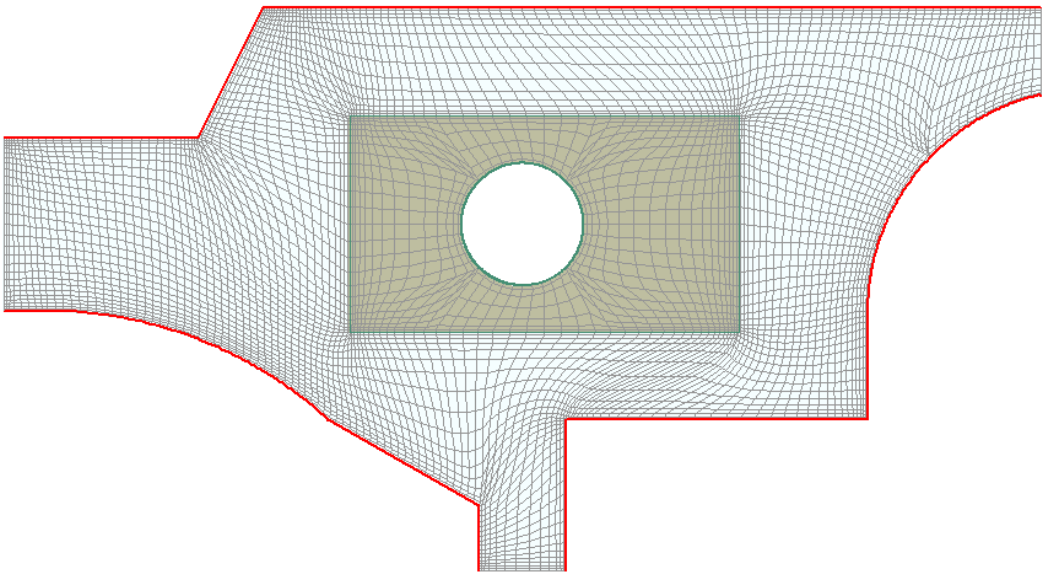
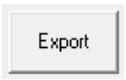


Figure B.73 – Mesh representation in the post-processing.

B3.9 – Data Export

The *Export Button* brings up the *DialogBox* for controlling the export process. Data can be exported for locations along a line, or for an entire region, as described next.



B3.9.1 – Line

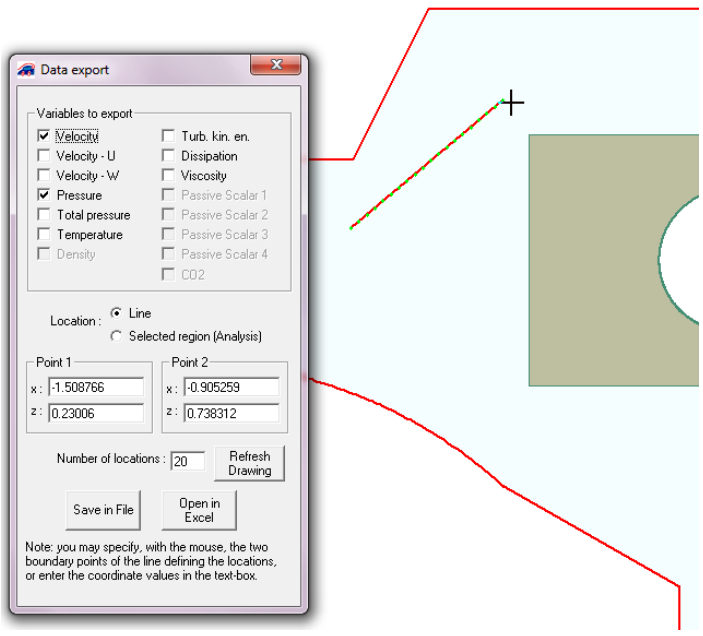


Figure B.74 – Selecting a line for data export.

For this option (cf. Figure B.74), you should specify the start and end point of a line segment in the calculation domain (*Point 1* and *Point 2*). This can be done either using the mouse (hold down the *Shift* key to easily define horizontal or vertical lines), or entering the coordinates directly in the corresponding *TextBoxes*. The line is sampled with the specified *Number of locations* (visible a green dots in the graphical display). *Refresh Drawing* updates the visualisation for the latest changes.

Two options are available:

- *Save in File*: Data can be exported into a file in TecPlot (only available for region export, unstructured mesh)) format or exported as text using the following format: the first line identifies the location (*Line*, in this case); the second line is a header, listing the name of the variables. The remaining lines store the coordinates and the values of the selected variables. (cf. Figure B.75).
- *Open in Excel*: Data is sent to a Microsoft Excel sheet, which is automatically opened (cf. Figure B.76).

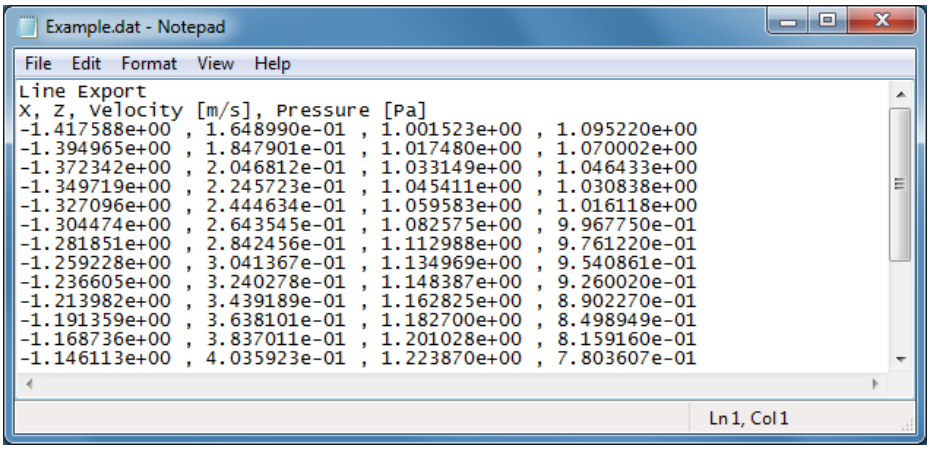


Figure B.75 – Data exported into a file from a defined line of points.

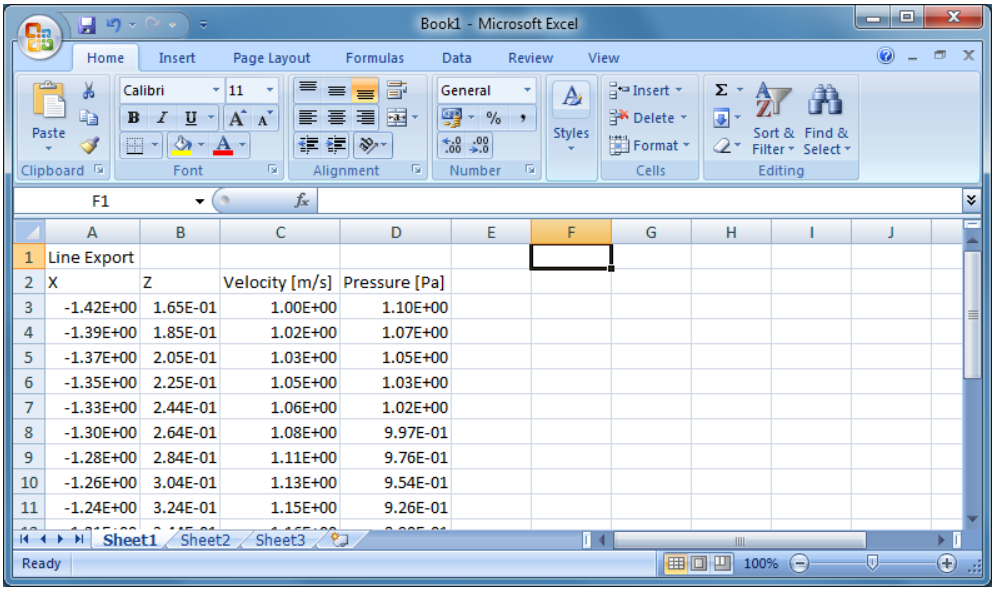
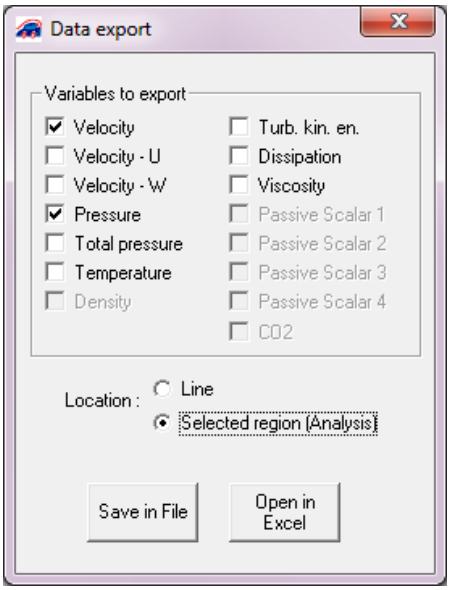


Figure B.76 – Data exported into Microsoft Excel from a defined line of points.

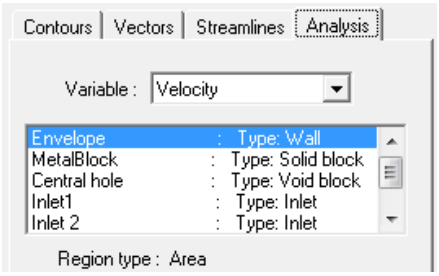
B3.9.2 – Selected region

This option exports the data for the region selected in the *Analysis Tab*.



There are two possible situations:

- If the selected region is of the *Area Type* (such as the *Envelope*, a *Void Block* or a boundary region), data is exported for locations at the middle point between mesh nodes over the region, as depicted in Figure B.77. Data can be exported into a file or into Microsoft Excel, in the same way as described in the preceding section.



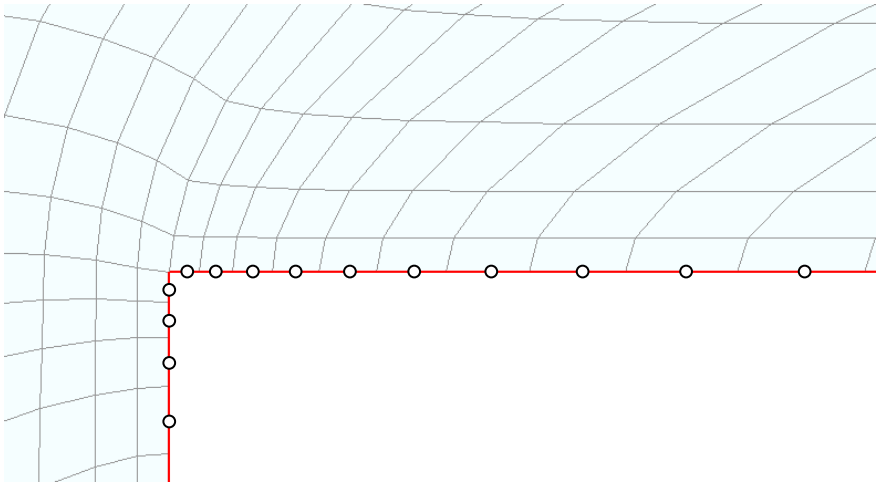


Figure B.77 – Identification of locations for export in an Area Type region.

- If the region is of the *Volume Type* (such as the *Calculation Domain*, a *Solid Block* or a *Pervious Block*), data is exported for the control volume centres in that region. Data exported into a file also includes, in this case, the indexation of each control volume, as depicted in Figure B.78. Exporting into Microsoft Excel is not available for *Volume Type* regions. If the region is the *Calculation Domain* and the mesh is unstructured, data can be exported in TecPlot format.

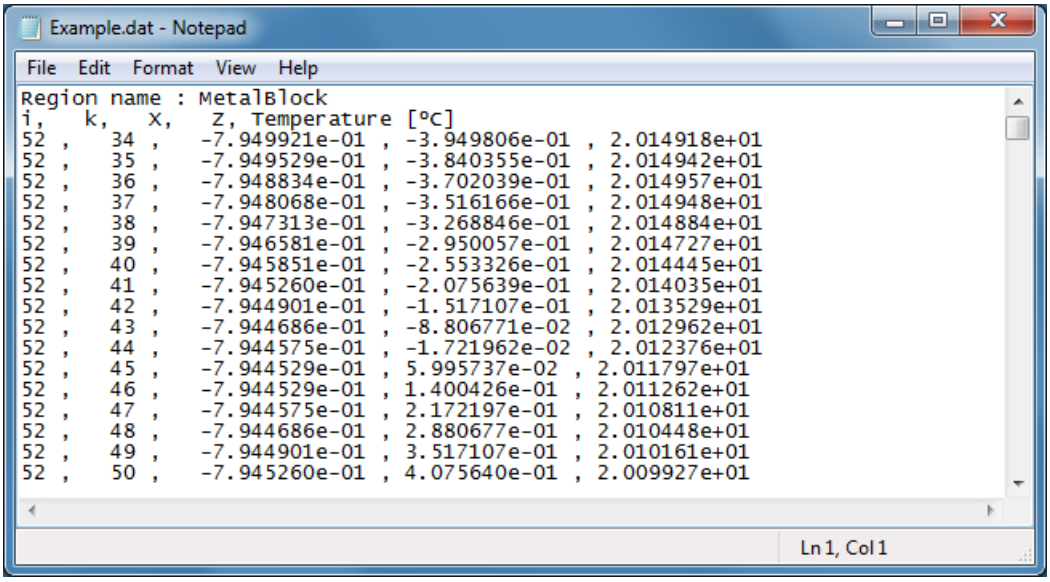


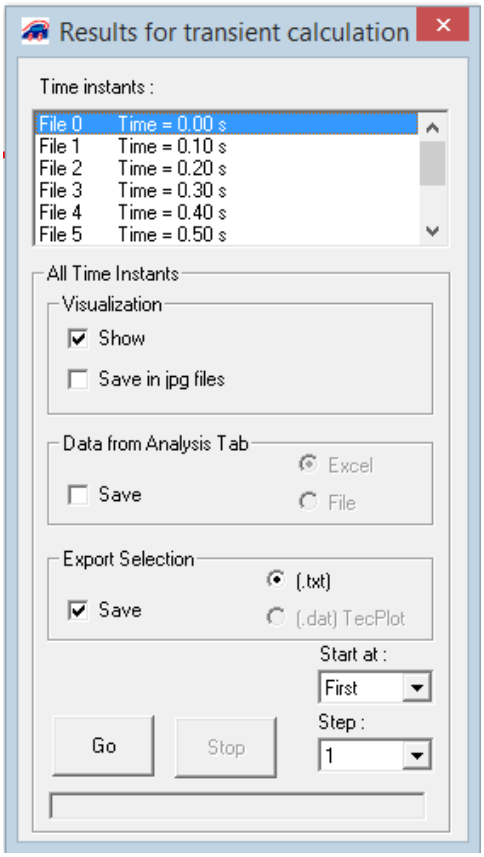
Figure B.78 – Data exported into a file (.txt format) from a Volume Type region.

B3.10 – Analysis of Transient Regime Problems

The illustration of the post-processing of transient regime problems will be done recurring to the conjugate heat transfer problem schematically depicted in Figure B.79. This corresponds to the initial state. Calculations are performed using a time step of *0.05 s* and results are stored every *0.1 s*. This problem is supplied with the EasyCFD installation, with the name *CHT*.

The saved intermediate data (cf. section B1.8.7) may be visualised upon checking *Intermediate results*. This command is unavailable for steady-state problems. The *DialogBox* shown on the right displays the stored *Time instants*. The post-processing tools are the same as previously described, but limited to the variables for which intermediate results were stored.

The available time instants may be worked in two ways, as described next.



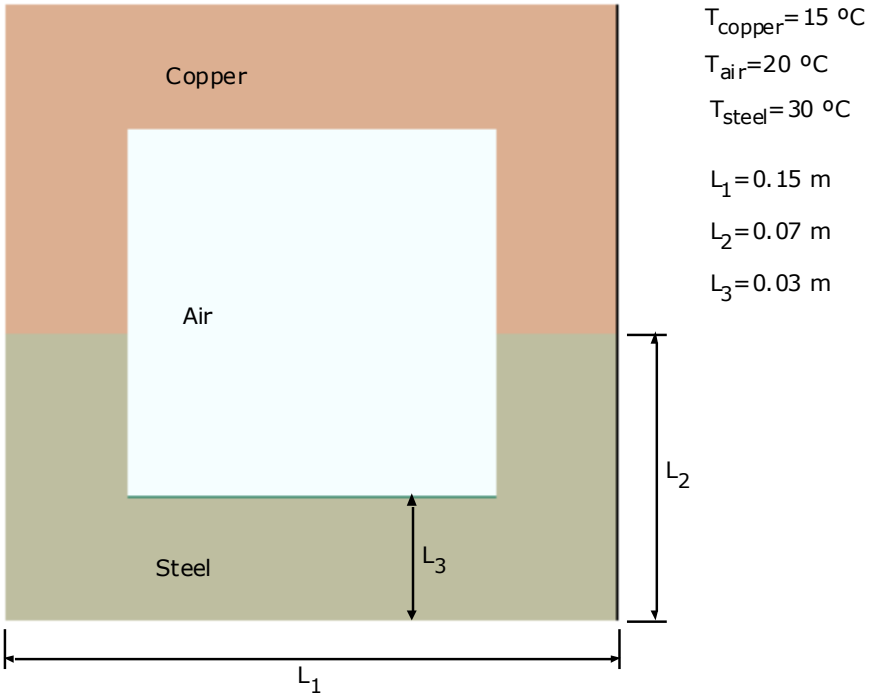


Figure B.79 – Definition of transient problem.

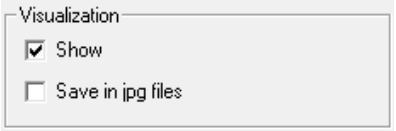
B3.10.1 – Single time instant

For the individual mode, you simply need to click in the desired time instant using the available list. The visualisation area and the data presented in the *Analysis Tab* (cf. section B3.7) are automatically updated with the solution corresponding to the selected time instant.

B3.10.2 – All time instants

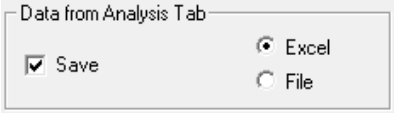
In this alternative, the *PushButton Go*, in *All time instants*, launches an automatic process where all available time instants are selected sequentially, either starting at the first available, or at the currently selected time instant (based on the selection made for *Start at:*).

Visualisation

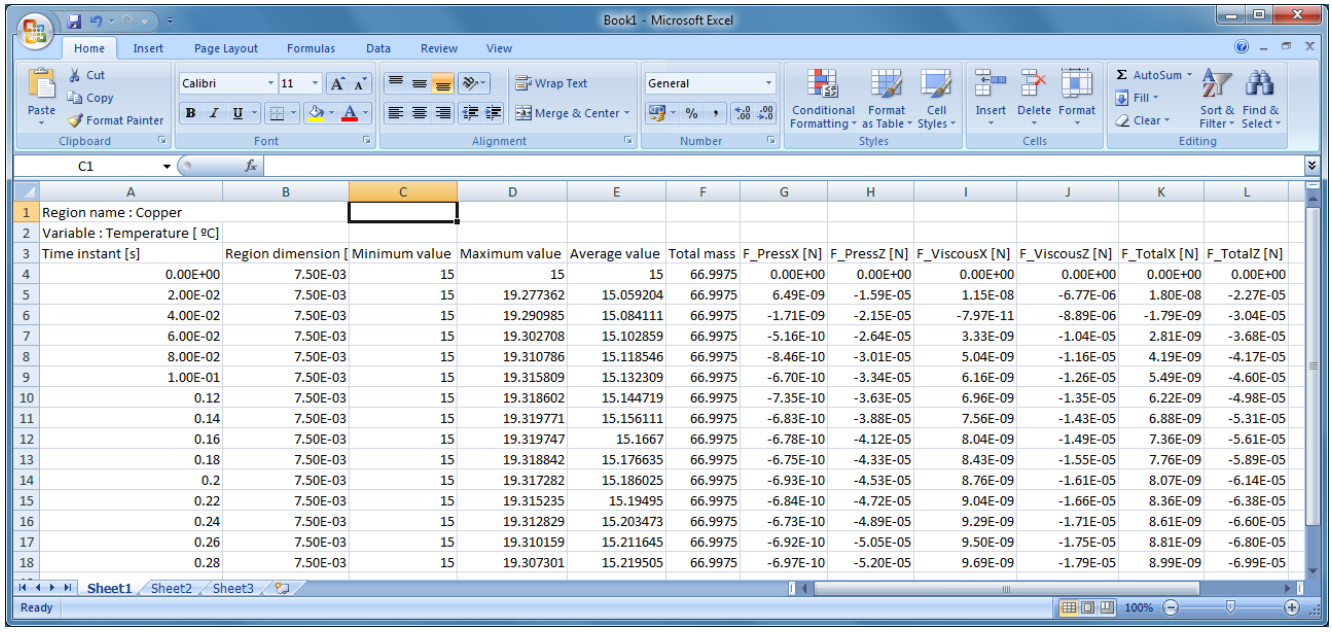


By checking *Show*, the visualisation is updated as the time instant changes sequentially. If, additionally, *Save in jpg files* is checked, the several visualisations corresponding to the time instants are saved in image files with *jpg* format, in the subfolder *\Trans\Images* of the project folder, with names *Tr0.jpg*, *Tr1.jpg*,etc. These image files can then be used to produce animations using adequate software (such as Windows Movie Maker).

Data from the Analysis Tab



This functionality takes all the data presented in *Analysis Tab* (cf. section B3.7) and either opens a new Excel sheet with that data (cf. Figure B.80) or, alternatively, saves the same data in the file *<ProjectFolder>\Trans\StatisticsRegion.dat*.



Region name : Copper											
Variable : Temperature [ºC]											
Time instant [s]	Region dimension [	Minimum value	Maximum value	Average value	Total mass	F_PressX [N]	F_PressZ [N]	F_ViscousX [N]	F_ViscousZ [N]	F_TotalX [N]	F_TotalZ [N]
0.00E+00	7.50E-03	15	15	15	66.9975	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2.00E-02	7.50E-03	15	19.277362	15.059204	66.9975	6.49E-09	-1.59E-05	1.15E-08	-6.77E-06	1.80E-08	-2.27E-05
4.00E-02	7.50E-03	15	19.290985	15.084111	66.9975	-1.71E-09	-2.15E-05	-7.97E-11	-8.89E-06	-1.79E-09	-3.04E-05
6.00E-02	7.50E-03	15	19.302708	15.102859	66.9975	-5.16E-10	-2.64E-05	3.33E-09	-1.04E-05	2.81E-09	-3.68E-05
8.00E-02	7.50E-03	15	19.310786	15.118546	66.9975	-8.46E-10	-3.01E-05	5.04E-09	-1.16E-05	4.19E-09	-4.17E-05
1.00E-01	7.50E-03	15	19.315809	15.132309	66.9975	-6.70E-10	-3.34E-05	6.16E-09	-1.26E-05	5.49E-09	-4.60E-05
0.12	7.50E-03	15	19.318602	15.144719	66.9975	-7.35E-10	-3.63E-05	6.96E-09	-1.35E-05	6.22E-09	-4.98E-05
0.14	7.50E-03	15	19.319771	15.156111	66.9975	-6.83E-10	-3.88E-05	7.56E-09	-1.43E-05	6.88E-09	-5.31E-05
0.16	7.50E-03	15	19.319747	15.1667	66.9975	-6.78E-10	-4.12E-05	8.04E-09	-1.49E-05	7.36E-09	-5.61E-05
0.18	7.50E-03	15	19.318842	15.176635	66.9975	-6.75E-10	-4.33E-05	8.43E-09	-1.55E-05	7.76E-09	-5.89E-05
0.2	7.50E-03	15	19.317282	15.186025	66.9975	-6.93E-10	-4.53E-05	8.76E-09	-1.61E-05	8.07E-09	-6.14E-05
0.22	7.50E-03	15	19.315235	15.19495	66.9975	-6.84E-10	-4.72E-05	9.04E-09	-1.66E-05	8.36E-09	-6.38E-05
0.24	7.50E-03	15	19.312829	15.203473	66.9975	-6.73E-10	-4.89E-05	9.29E-09	-1.71E-05	8.61E-09	-6.60E-05
0.26	7.50E-03	15	19.310159	15.211645	66.9975	-6.92E-10	-5.05E-05	9.50E-09	-1.75E-05	8.81E-09	-6.80E-05
0.28	7.50E-03	15	19.307301	15.219505	66.9975	-6.97E-10	-5.20E-05	9.69E-09	-1.79E-05	8.99E-09	-6.99E-05

Figure B.80 – Excel sheet with the data from the Analysis Tab.

Export Selection

Export Selection takes the settings in the *Data Export DialogBox* (cf. section B3.9) and creates several data files, one per each time step. Files are saved in the subfolder *\Trans\Export* of the project folder and named *Export0.dat*, *Export1.dat*, etc. The files format is similar to that obtained for a simple export as described in section B3.9, with an additional (first) line containing the time instant information (cf. example in Figure B.81).

For unstructured meshes, export is also available in TectPlot format, If the selected region is the *Calculation Domain*. In this case, a single file is produced *\Trans\Export\ExportTecPlot.dat.*

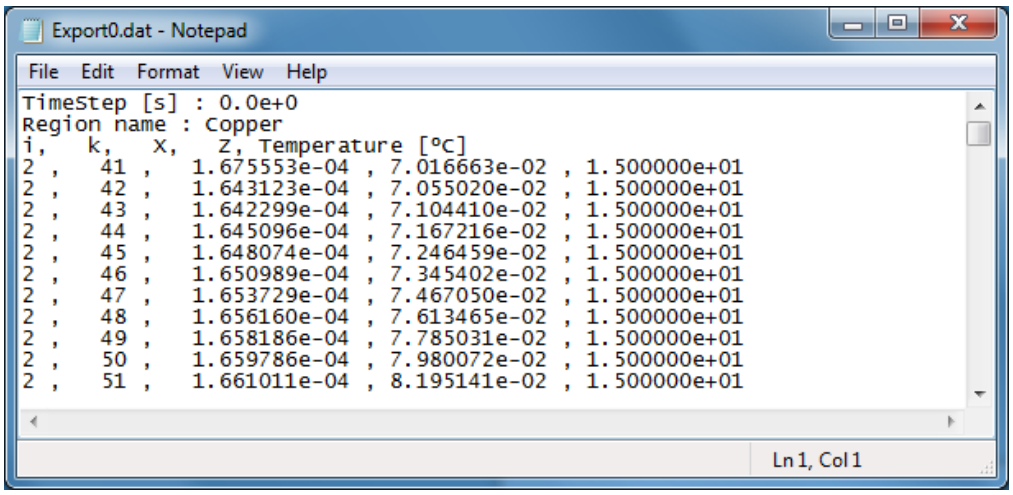


Figure B.81 – Data export file example for transient information (volume type region).

C – Test Cases

Note: The files corresponding to the following test cases are supplied with the software.

C1 – Laminar Flow between Two Parallel Plates

File: LaminarProfile.def

The analytical solution for the developed laminar flow between two parallel plates is given by:

$$V = -\frac{1}{2\mu}\frac{dp}{dx}\left(H^2 - z^2\right) \tag{1.1}$$

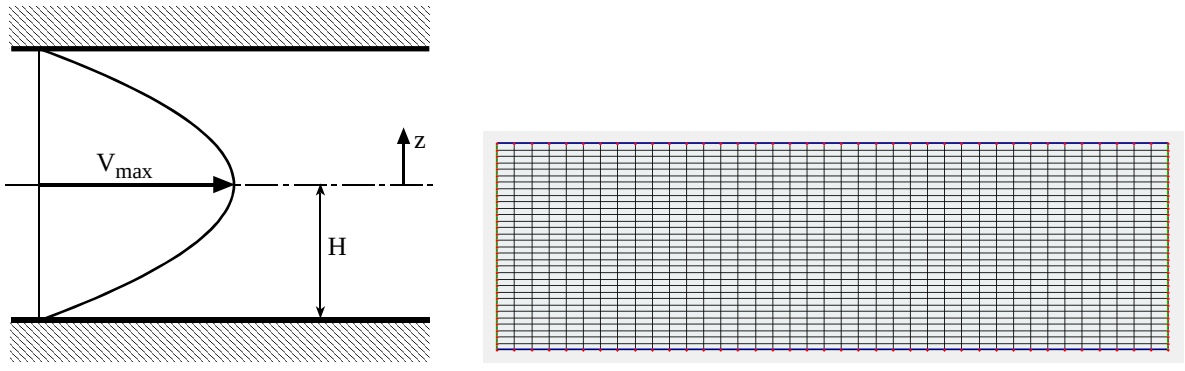
where H is half the distance between the plates and z is the coordinate normal to the plates, starting at the symmetry plane (cf. Figure C.1(a)). The longitudinal pressure gradient is:

$$\frac{dp}{dx} = -\frac{3\overline{V}}{H^2}\mu \tag{1.2}$$

where $\overline{V}$ is the average velocity. The maximum velocity, reached at the symmetry plane ($z=0$), may be obtained by substituting equation (1.2) into (1.1):

$$V_{max} = 1.5\overline{V}$$

Computations were carried out in half the domain, imposing a symmetry boundary condition and using a uniform grid spacing (Figure C.1). To obtain a developed velocity profile without using a very large length, the option *Profile* was activated when imposing the inlet boundary condition (cf. section B1.6.2).



(a) (b)

Figure C.1 – Problem definition. (a) schematic drawing; (b) EasyCFD calculation domain.

Figure C1.1 – Problem definition.

The following values where used:

$H = 0.04\,m$; $\overline{V} = 10^{-3}\,m/s$; fluid: water ($\mu = 10^{-3}\,N\,s/m^2$; $\rho = 10^3\,kg/m^3$)

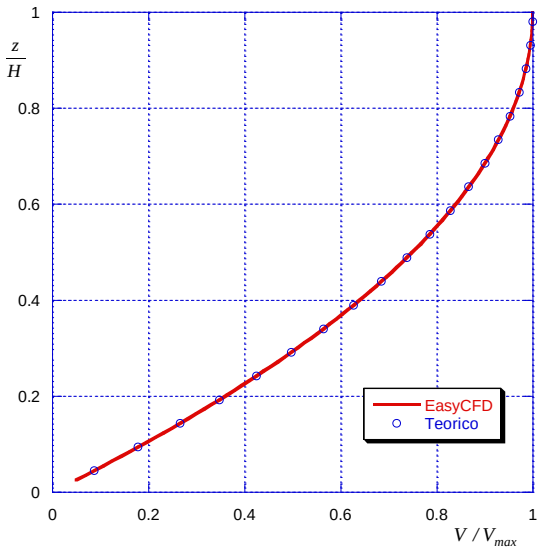


Figure C.2 – Velocity profile.

Figure C.2 shows the obtained velocity profile overimposed on the theoretical solution. As one may observe, the two curves are practically coincident. For this case, the theoretical pressure gradient is:

$$\left(\frac{dp}{dx}\right)_{theoretical} = -\frac{3\times10^{-3}}{0.04^2}\times10^{-3} = 1.875\times10^{-3}\ Pa/m$$

To compute the pressure gradient given by the **EASYCFD** results, the *Export* feature was used to open an Excel sheet for pressure along an horizontal line (cf. section B3.9.1). The obtained value is:

$$\left(\frac{dp}{dx}\right)_{EasyCFD} = 1.8736\times10^{-3}\ Pa/m$$

The maximum velocity, at the symmetry plane, given by the software, is $V_{max} = 1.497\,m/s$, while the theoretical value is *1.5 m/s*.

As a suggestion, you certainly may improve these results by using a more refined grid with a better clustering near the low boundary.

C2 – Turbulent Flow between Two Parallel Plates

File: TurbulentProfile.def

As for the previous case, computations where performed in half the domain, imposing a symmetry boundary condition. In order to resolve the flow near the wall, proper clustering was ensured at this boundary (cf. Figure C.3). To obtain a developed velocity profile without using a very

large length, the option *Profile* was activated when imposing the inlet boundary condition (cf. section B1.6.2).

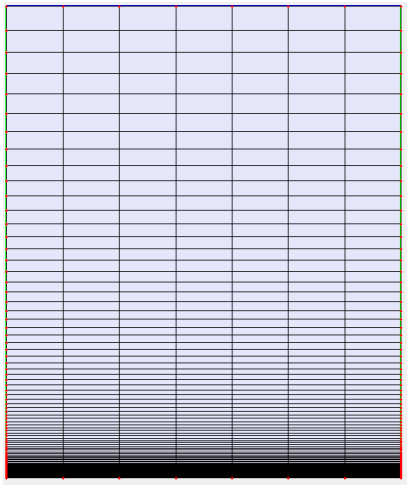


Figure C.3 – Mesh used.

According to the following values adopted for this problem,

$H = 0.6\,m$; $\overline{V} = 1\,m/s$; fluid: air ($\mu = 1.824 \times 10^{-5}\,N\,s/m^2$; $\rho = 1.1884\,kg/m^3$)

the flow Reynolds number is 39092. Computations where carried out using the *SST* k-ω turbulence model. Due to its hybrid characteristics, unlike the standard k-ε model, this model allows computations to be done in the viscous sublayer.

The obtained results were compared with the direct numerical simulation presented by Kim et al., 1987, which were done for a Reynolds number of 3300. Figure C.4 displays the boundary layer profile obtained with both simulations. Note that, due to the lower Reynolds number adopted by Kim et. al., their range of u^+ values is not as large as the present one. One may note a very good agreement, except for the transition between the viscous and the log layer.

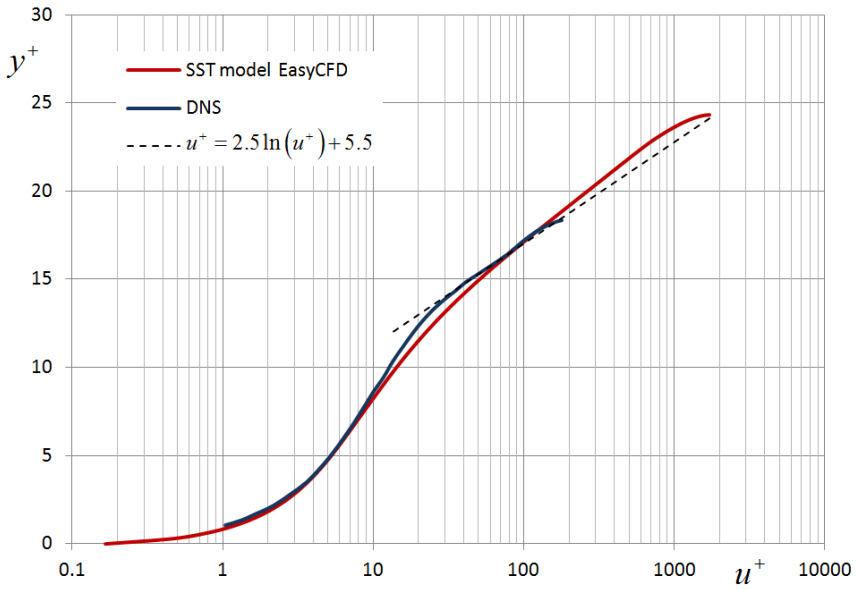


Figure C.4 – Boundary layer profile.

C3 – Turbulent Flow Over a Backward-Facing Step

File: Step.def

The flow over a backward-facing step (Figure C.5) is a typical problem for the validation of turbulence models and their implementation. This is a rather simple geometry, where the flow separation point is fixed at the step corner. The reattachment length depends on the Reynolds number and on the ratio between the inlet and the outlet sections. The prediction of the reattachment position is a demanding test for the turbulence model, as it is quite sensitive to the fluid effective viscosity.

In the present case, the flow Reynolds number, based on the inlet centreline velocity and on the height of the outlet section, is *132000*. Present results are compared with the ones described by Ting and Prakash (2005) and by Papageorgakis and Assanis (1999). The first work presents, among others, (1) results obtained with the k-ε model in a version implemented in the commercial software Fluent, using wall laws; (2) computations performed by Thangam and Hur (1991) with the k-ε model and (3) experimental results obtained by Kim et al. (1978). In these works, authors took, for the inlet boundary, a previously developed velocity profile. Papageorgakis and Assanis (1999) show calculations obtained with the *RNG* linear and the non-linear k-ε model, for *Re=135000*.

The hybrid discretisation scheme and the harmonic interpolation for viscosity were adopted. The grid has 179×102 nodes, with *15807* active nodes

In the present case, the inlet velocity profile was obtained with the feature *Profile* activated. Figure C.6 presents the inlet velocity profile thus obtained, along with the profile by Ting and Prakash (Fluent code) and by Papageorgakis and Assanis. As one may observe, the developed profile agrees well with the experimental data (another possible approach would be to use a *TableFunction* to assign the inlet profile according to the experiments).

Figure C.7 displays the streamlines in the recirculation region downstream of the step, together with the results computed by other authors. The reattachment point was calculated, by **EasyCFD**, at $x/H = 6.4$, while Ting and Prakash present the values *6.28*, *6.30* and *5.7* for other works. Papageorgakis and Assanis refer values in the range of *7.35* to *8.3*.

The graphics in Figure C.8 represent the velocity profile at different locations downstream of the step. The differences between the several contributions are certainly also due to the different inlet velocity profiles. The predictions of both Fluent and **EasyCFD** underestimate the length of the recirculation region, a well known characteristic of the predictions given by the k-ε model for this type of flows.

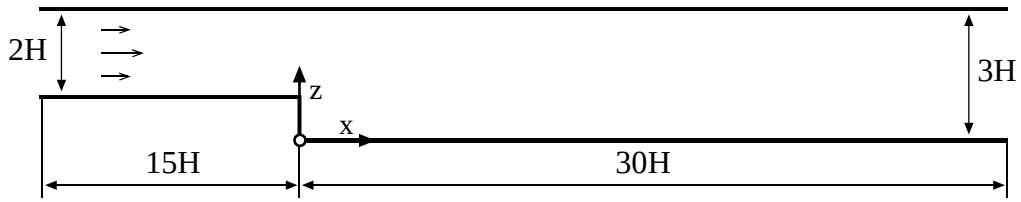


Figure C.5 – Geometrical configuration of the backward facing step (the drawing is not to scale).

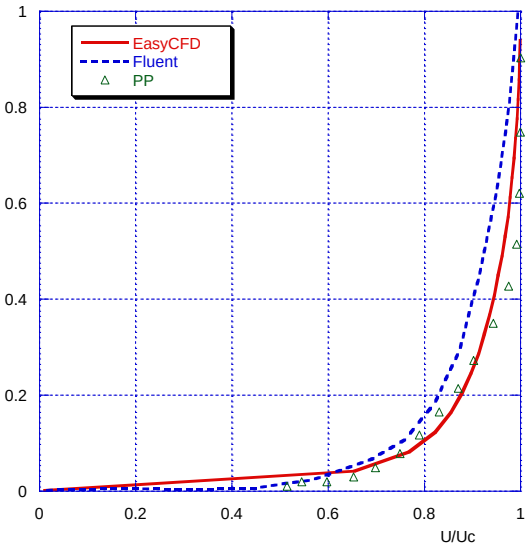
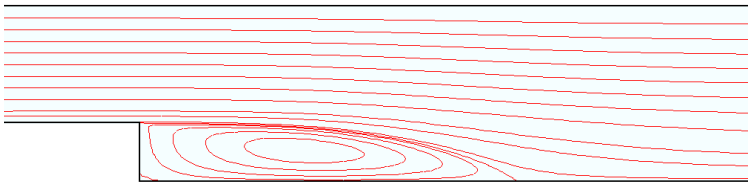
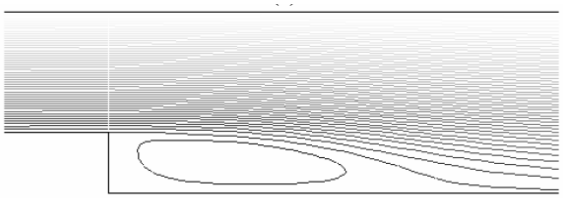


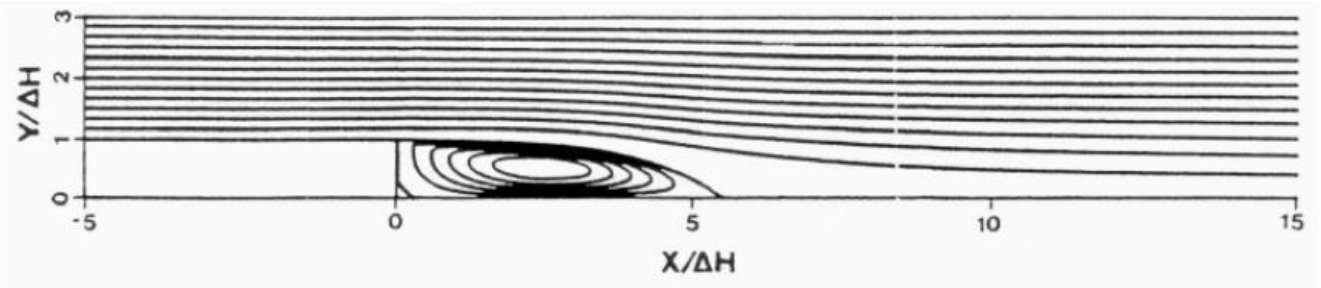
Figure C.6 – Inlet velocity profile.



(a) – EasyCFD



(b) - Fluent (in Ting and Prakash 2005)



(c) - Thangam and Hur (1991)

Figure C.7 – Streamlines in the recirculation region.

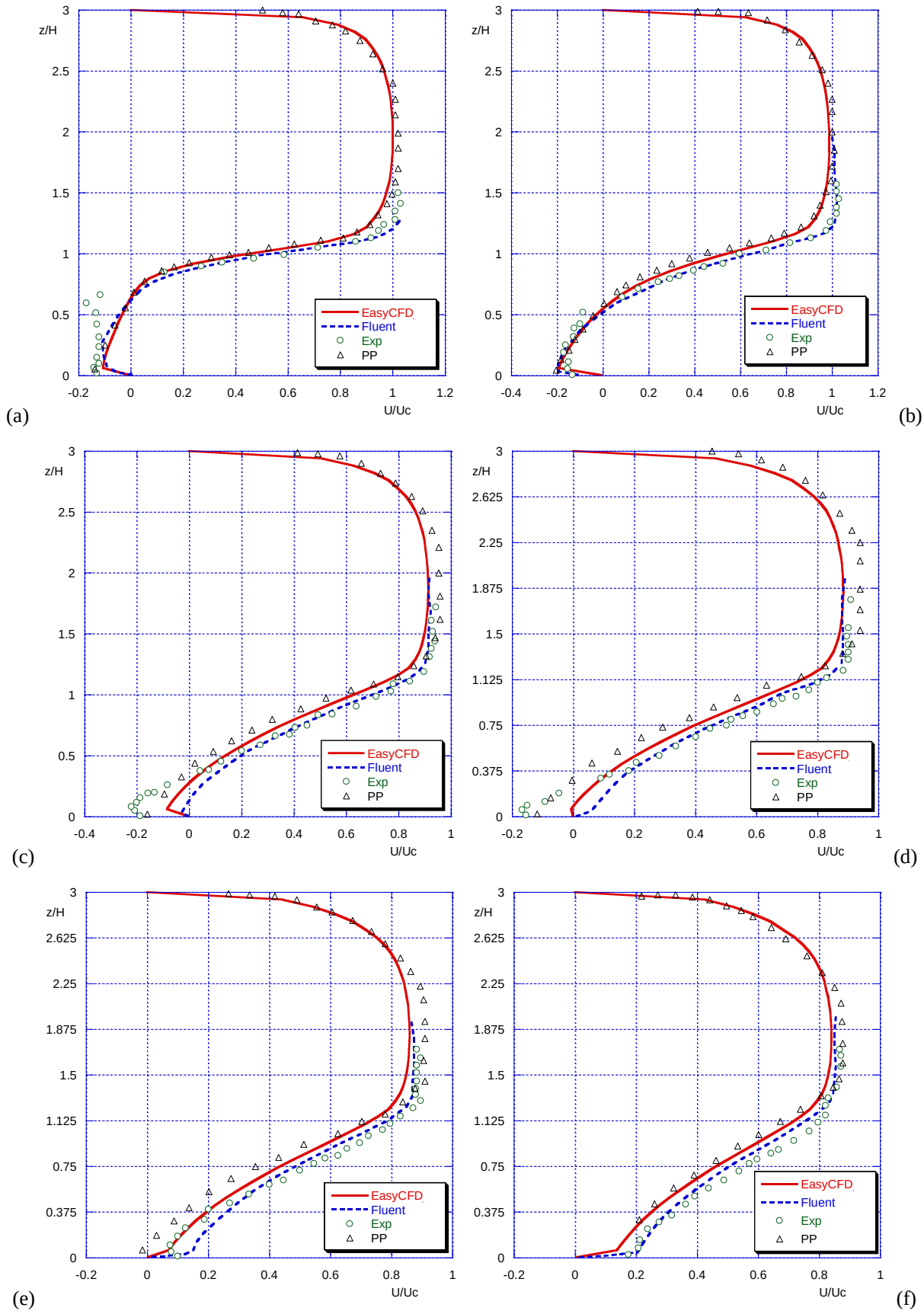


Figure C.8 – Velocity profiles several locations downstream of the step.

(a) - x/H=1.333; (b) - x/H=2.667; (c) - x/H=5.333; (d) - x/H=6.220; (e) - x/H=7.113; (f) - x/H=8.0

C4 – Natural Convection Inside a Cavity

File: ConvLam.def

The natural convection flow in a cavity is a classical test for the numerical methods in fluid dynamics. In this problem, constant temperature is imposed in two vertical walls. The horizontal walls are adiabatic (cf. Figure C.9).

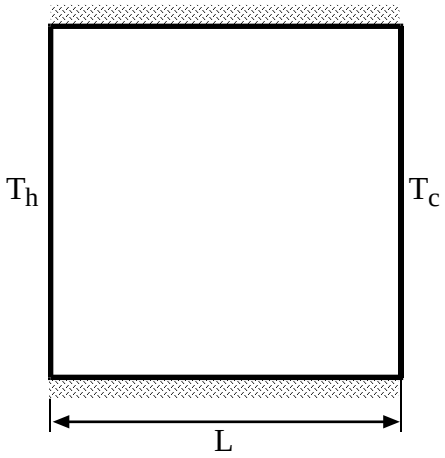


Figure C.9 – Problem definition.

The problem is governed by the following non-dimensional parameters:

$$\text{Prandtl number, } Pr = \frac{\nu}{\alpha}$$

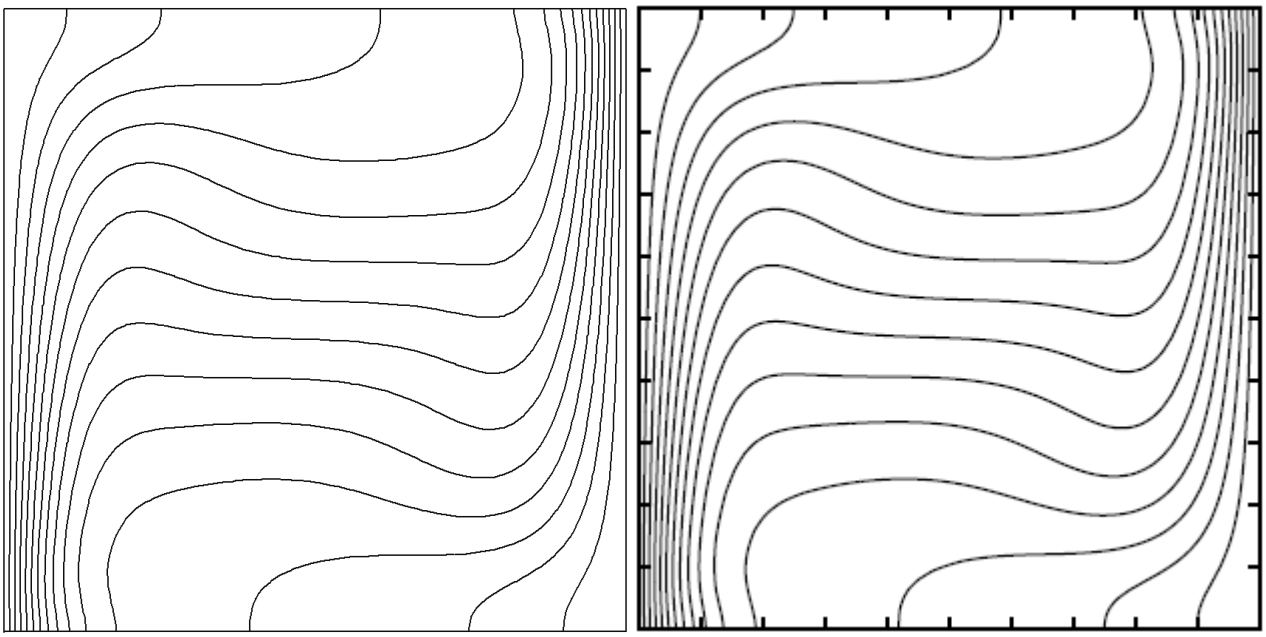
where the thermal diffusivity is $\alpha = \lambda / \left(\rho c_p \right)$, ν is the kinematic viscosity, λ is the thermal conductivity, ρ is the density and c_p is the heat capacity.

$$\text{Rayleigh number, } Ra = \frac{g \beta \Delta T L^3 Pr}{\nu^2}$$

where β is the thermal expansion coefficient, g is the gravity acceleration and $\Delta T = T_h - T_c$ is the temperature difference between the vertical walls.

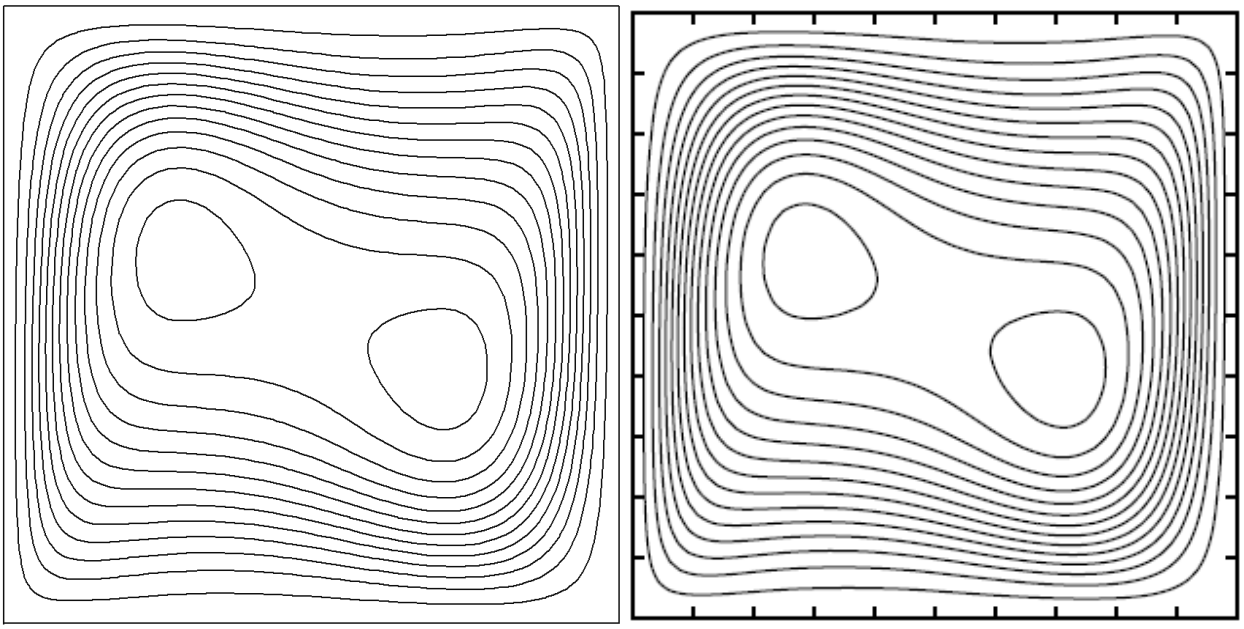
The transition between laminar and turbulent flow takes place for $Ra = 10^7$, approximately. Results are present Rayleigh number $Ra = 10^5$ (laminar regime). Air is the operating fluid, for which $Pr = 0.71$.The hybrid advection scheme was used and the Boussinesq approximation was adopted.

Computations were performed in a non-uniform grid, with $82 \times 82 = 6400$ nodes. Vahl Davis (1983) and Vahl Davis et al. (1983) report reference results for several Rayleigh numbers in laminar regime and compare solutions given by several authors. Dixit and Babu (2006) present results for laminar and turbulent flow.



(a) – EasyCFD (b) – Dixit and Babu (2006)

Figure C.10 – Isothermal lines. $Ra=10^5$.



(a) – EasyCFD (b) – Dixit and Babu (2006)

Figure C.11 – Streamlines. $Ra=10^5$

Figure C.10(a) and (b) display the isothermal lines, generated using uniform value spacing between the minimum and the maximum verified within the domain. Figure C.11(a) and (b) represent the flow streamlines. For this case, the minimum and the maximum values used do not correspond to the amplitude of the stream function within the domain. These values were adjusted in **EasyCFD** to correspond to those employed in Dixit and Babu (2006). The agreement between the calculations is quite good, for both types of results.

Vahl Davis and Jones (1983) present normalised maximum values for the *u* and for the *w* components of velocity:

$$u^* = \frac{u\,L}{\alpha} \quad ; \quad w^* = \frac{w\,L}{\alpha}$$

occurring in the vertical and in the horizontal symmetry plane, respectively. Table 2 shows the results obtained with **EasyCFD**, the reference values of Vahl Davis and the range of variation for the *37* contributions reported by Vahl Davis and Jones (1983). This range does not include the minimum and the maximum reported values, since these clearly fall outside the general trend of the remaining contributions.

Table 2

	EasyCFD	Vahl Davis	Interval
u^*_{max}	*34.44*	*34.81*	*34.2-41.2*
w^*_{max}	*68.21*	*68.68*	*65.8-70.45*

C5 – Pulsatile Flow Inside a Symmetric Channel with Wavy Walls

File: PulsatileFlow.def

This problem concerns a pulsatile laminar flow inside a channel symmetric with sinusoidal walls. The geometrical configuration is presented in Figure C.12.

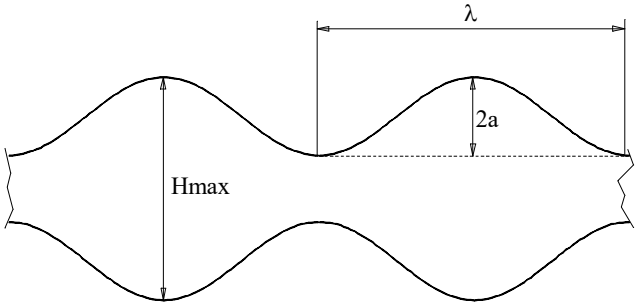


Figure C.12 – Geometry for the wavy channel.

The instantaneous inlet flow is defined by:

$$Re_i(t) = Re_m\left(1 + A\sin\left(2\pi t / T\right)\right)$$

where t is time, T is the oscillation period, A is the oscilation amplitude and Re_m is the average Reynolds number:

$$Re_m = \frac{Q_m}{\nu}$$

with Q_m as the volumetric flowrate per unit depth.

The temporal scale is given by the Womersley number, α^2:

$$\alpha^2 = \frac{H_{max}^2 2\pi / T}{\nu}$$

The dimensionless time t^* is defined as:

$$t^* = \frac{t}{T}$$

For the present case, the following values where considered:

Geometrical parameters: $2a = 3.5mm \quad H_{max} = 10mm \quad \lambda = 14mm$

Flow parameters: $\alpha^2 = 650 \quad A = 2.5 \quad Re_m = 50$

Figure displays the instantaneous Reynolds number as function of the dimensionless time, for an entire cycle.

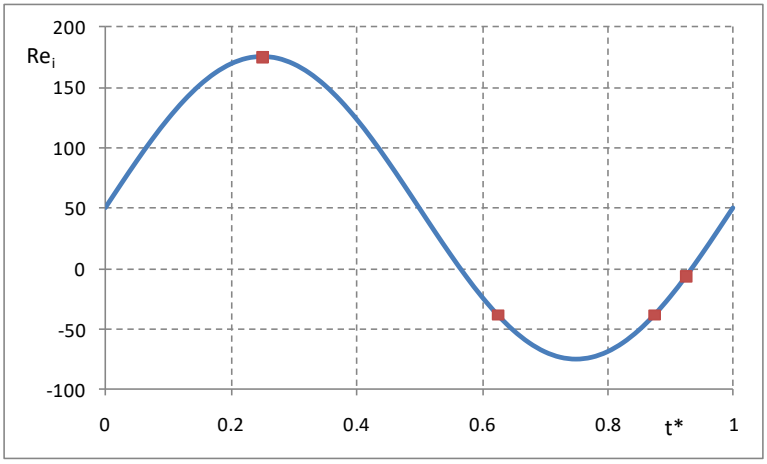


Figure C.13 – Instantaneous Reynolds number as function of the dimensionless time. Red dots represent time instants for Figure C.15.

The channel is periodic. In the present implementation, 3 waves were considered, and periodic boundary conditions were implemented by using the *Profile* option for the inlets (cf. Figure C.14). The velocity value is imposed at the two inlets, using *TableFunctions* with opposite sign as function of time (thus, at a certain time instant, there will be one inlet and one outlet). Although two entire temporal cycles were simulated (in order to achieve a developed flow in temporal terms), only one cycled needs to be specified in the *TableFunction*; since the first and the last velocity values are similar, periodic conditions are automatically assumed. Comparison with data by Nishimura and Matsune (1998) is presented in Figure C.15, at the central wave, for different values of the adimensional time t^* (cf. red dots in Figure C.13; the streamlines values used in the comparison is not the same, as these authors don´t mention the criterion followed for the choice of the streamlines).

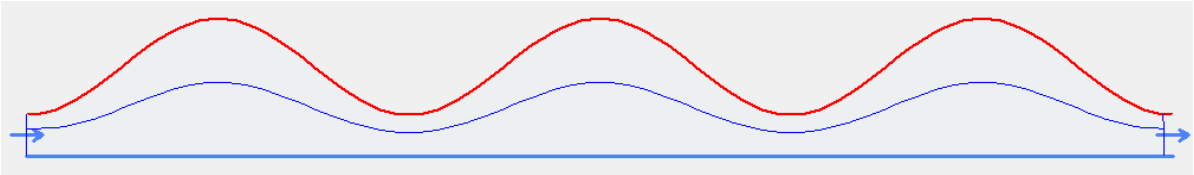
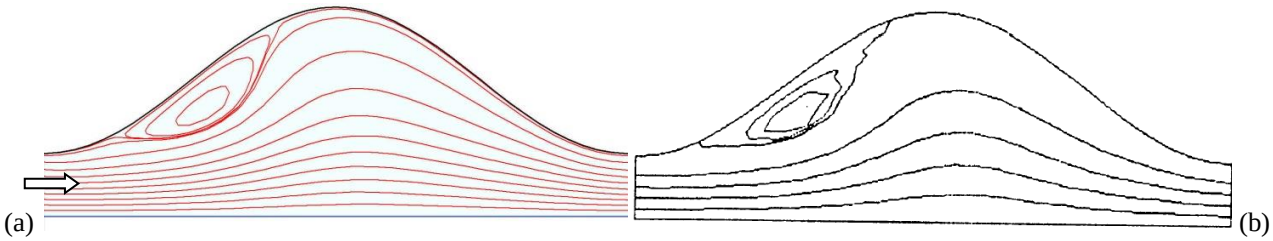
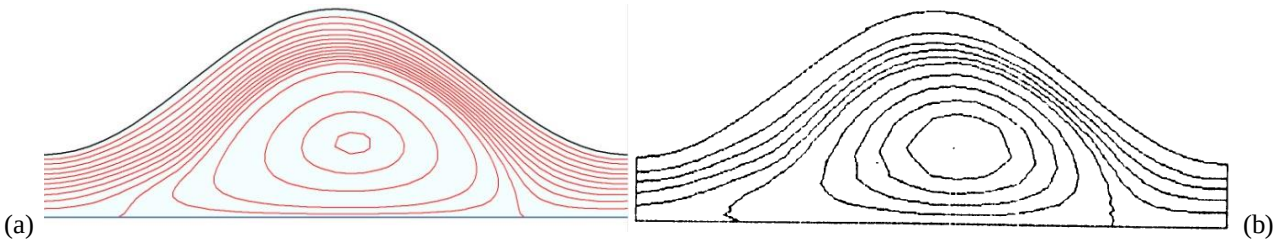


Figure C.14 – Geometric configuration for the implementation in *EasyCFD* (sinusoidal blue line represents the profile option for the inlets).



t = 0.25*



t = 0.625*

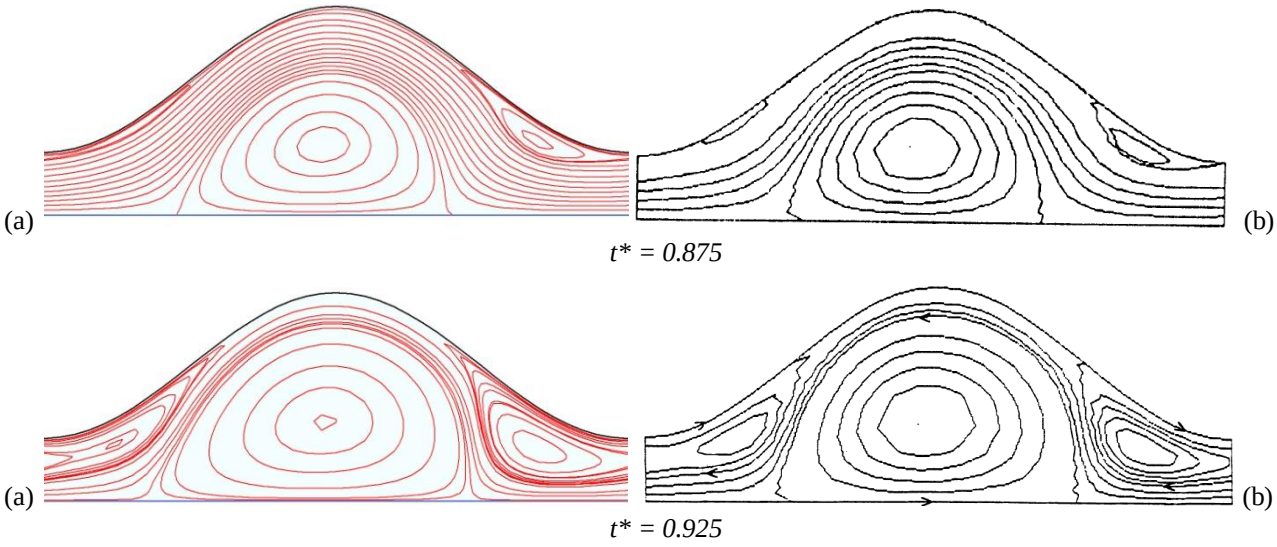


Figure C.15 – Streamlines visualization for different values for the adimensional time. (a): *EasyCFD*; (b): Nishimura e Matsune (1998).

C6 – Drag Coefficient for a sphere

Files: Sphere.def

This problem consists on the determination of drag coefficient of a sphere for a wide range of Reynolds numbers, defined as:

$$Re_D = \frac{\rho V_\infty D}{\mu}$$

where D is the sphere diameter. The drag coefficient is:

$$C_D = \frac{F_x}{0.5\,\rho V_\infty^2 A}$$

where A if the frontal area and F_x is the total force along the streamwise direction. The flow is laminar up to $Re = 3.7E5$, approximately. Flow prediction around the transition regime is particularly difficult due to the large separation region, large scale vortex shedding, etc. Several studies were devoted to this subject, such as, for example, Constantinescua and Squires, 2004.

The present simulations were carried out using the domain presented in Figure C.16. Note that, due to the software limitations, the flow was considered as axisymmetric, though in reality, despite the axisymmetric geometry and boundary conditions, the alternate vortex shedding breaks the symmetry. Due to its tridimensional intrinsic character, this type if phenomena cannot, for the present case, be predicted with EasyCFD.

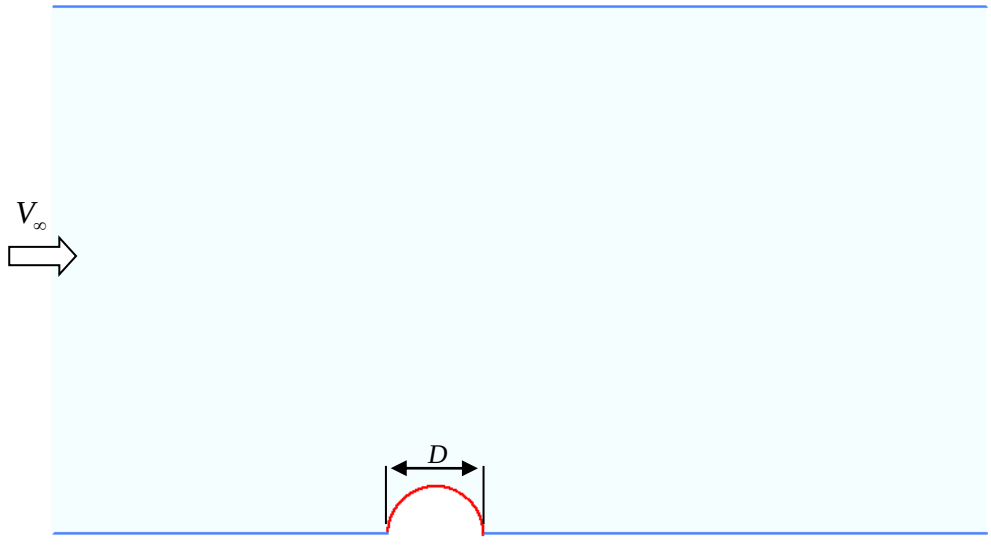


Figure C.16 – Problem definition for the flow around a sphere.

For the turbulent region, both the k-ε and the *SST* k-ω turbulence models were employed. The supplied file *Sphere.def* can be used to solve this problem. The results hereafter presented were computed using the multigrid capability, after doubling twice the unstructured mesh size, from the initial 2063 nodes, up to 8252 nodes and, finally, 33008 nodes (cf. section B1.4.5).

The obtained results are presented in Figure C.17. One may see that results in the laminar region agree very well with experimental data. The turbulent region is not so well predicted, certainly due to the limitations in the turbulence models. The reader is suggested to check for dependency of results with mesh clustering near the sphere, namely with the y^+ values, for the turbulent region.

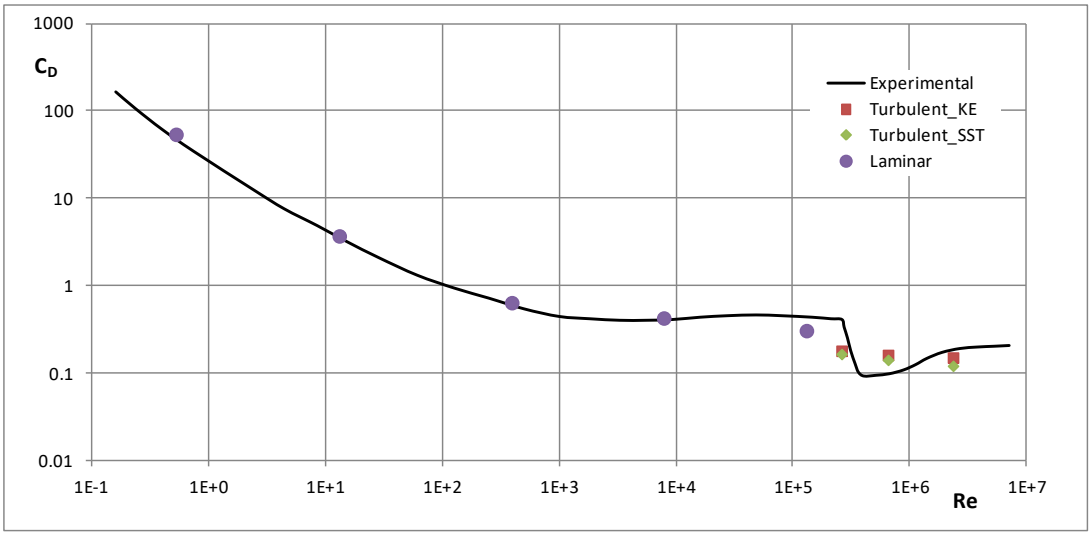


Figure C.17 – Drag coefficient versus Reynolds number, for the flow around a sphere.

C7 – Lift Coefficient for the NACA 2412 Airfoil

Files: Naca2412_NST.dat

This problem consists on the calculation of the lift coefficient for the turbulent flow around a NACA 2412 airfoil. The airfoil profile is given by a point data file, which is imported into **EasyCFD** (cf. section B1.2.8). The lift coefficient C_l is defined as:

$$C_l = \frac{F_z}{0.5\,\rho V_\infty^2 A}$$

where F_z is the total vertical force acting on the profile, V_∞ is the incident wind speed, A is the planform airfoil area (Figure C.18) and ρ is the fluid density. The problem is governed by the flow Reynolds number (assuming that compressible effects are negligible) and by the incidence angle θ.

Calculations were done for $Re = 3.2 \times 10^6$. The results presented in Figure C.19 were obtained with an ustructured mesh of about 20000 nodes. Experimental data was published by Anderson (1989). The agreement is quite reasonable up to the stall region, after which the predictions underestimate the lift coefficient. It must be emphasised that drag and lift values are quite dependent on mesh resolution, so that further calculations should be done with higher resolution meshes. The stall region, which occurs for the present case for angles above 16º is very demanding for turbulence models. It is recognized that most likely the Reynolds average approach, such the one followed by most commercial codes, is not enough for the prediction of such flows.

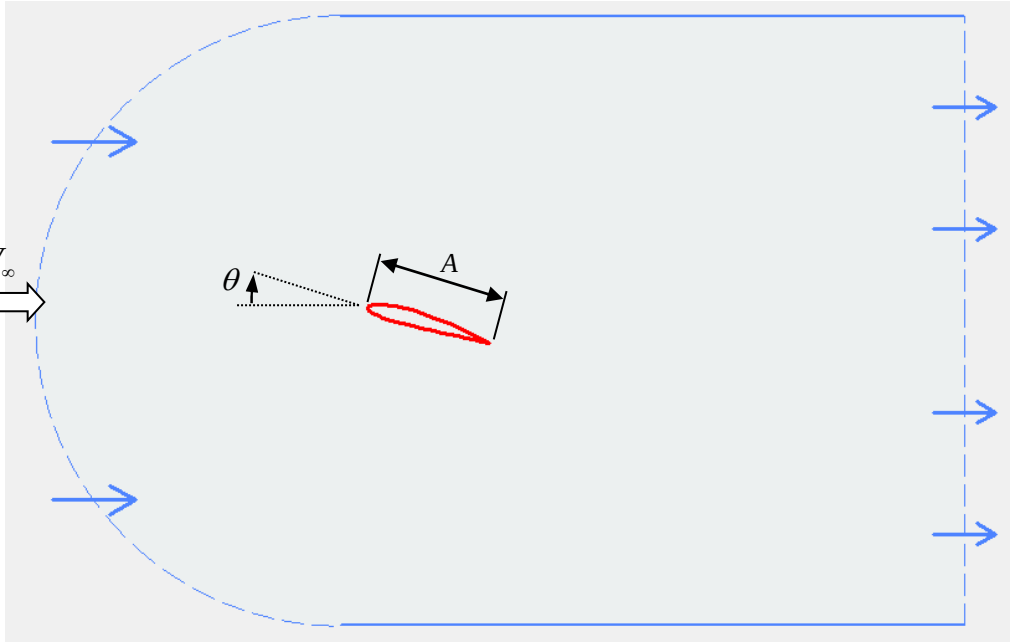


Figure C.18 – Problem definition.

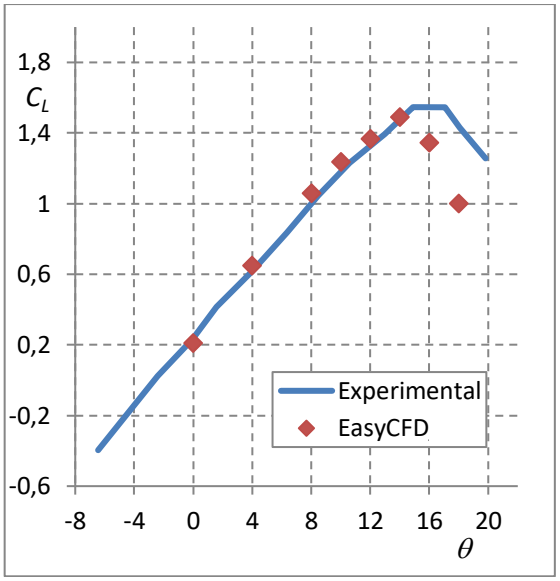


Figure C.19 – Lift coefficient as function of the incidence angle..

The validation cases here presented are, purposely, a starting point for other studies that may contemplate the influence of grid characteristics, advection schemes, inlet conditions for turbulence, etc. Wishing you a good work!!

D – References

- Anderson, J.D. Jr., *Introduction to Flight*, McGraw-Hill Book Company, New York, 1989.

- Blacker, T.D. and Stephenson, M.B., "Paving: a New Approach to Automated Quadrilateral Mesh Generation", International Journal for Numerical Methods in Fluids, Vol. 32, pp. 811-847, 1991.

- Constantinescua, G. And Squires, K., "Numerical investigations of flow over a sphere in the subcritical and supercritical regimes", Physics of Fluids, vol. 16, N. 5, 2004.

- Dixit, H.N and Babu, V., "Simulation of High Rayleigh Number Natural Convection in a Square Cavity Using the Lattice Boltzmann Method", International Journal of Heat and Mass Transfer, Vol. 49, pp. 727-739, 2006.

- Djilali, N., Gartshore, I. and Salcudean, M., "Calculation of Convective Heat Transfer in Recirculating Turbulent Flows Using Various Near-Wall Turbulence Models", Numerical Heat Transfer, Part A, vol. 16, pp. 189-212, 1989

- Ehrlich, L. W., "An Ad Hoc SOR Method", Journal of Computational Physics, Vol. 44, pp. 31-45, 1981.

- Hayase T., Humphrey, J.A.C., and Greif, R., "A consistently formulated quick scheme for fast and stable convergence using finite-volume iterative calculation procedures. Journal of Computational Physics, Vol. 98, pp. 108–118, 1992.

- Kim, J., Kline, S. J. and Johnston, J. P. 1978, "Investigation of separation of a turbulent shear layer: Flow over a backward-facing step", Thermosciences Division, Dept. of Mech. Engg, Stanford University.

- Kim, J., Moin, P. And Mose, R., "Turbulence statistics in fully developed channel flow at low Reynolds Number", J. Fluid. Mechanics, vol. 177, pp. 133-166, 1987.

- Launder, B.E. and Spalding, D.B., "The Numerical Computation of Turbulent Flows" Computer Methods in Applied Mechanics and Engineering, vol. 3, pp. 269-289, 1974.

- Launder, B.E. and Spalding, D.B., *Mathematical Models of Turbulence*, Academic Press London and New York, ISBN 0-12-438050-6, 1972

- Majumdar, S., "Role of Underrelaxation in Momentum Interpolation for Calculation of Flow with Nonstaggered Grids", Numerical Heat Transfer, Vol. 13, N. 1, pp. 125-132, 1988.

- Menter, F. R., Kuntz, M., and Langtry, R., "Ten Years of Industrial Experience with the SST Turbulence Model," Turbulence, Heat and Mass Transfer 4, ed: K. Hanjalic, Y. Nagano, and M. Tummers, Begell House, Inc., pp. 625 – 632, 2003.

- Menter, F.R. Zonal two-equation k-ω turbulence model for aerodynamic flows. *AIAA Paper 1993-2906*, 1993.

- Menter, F.R., Ferreira, J. C., Esch, T and Konno, B., "The SST Turbulence Model with Improved Wall Treatment for Heat Transfer Predictions in Gas Turbines", Proceedings of the International Gas Turbine Congress 2003 Tokyo, November 2-7, 2003.

- Nishimura, T. and Matsune, S., "Vortices and Wall Shear Stresses in Asymmetric and Symmetric Channels with Sinusoidal Wavy Walls for Pulsatile Flow at Low Reynolds Numbers", International Journal of Heat and Fluid Flow, 19, pp. 583-593, 1998.

- Papageorgakis, G.C. and Assanis, D.N., "Comparison of Linear and Nonlinear RNG-Based k-ε Models for Incompressible Turbulent Flows", Numerical Heat Transfer, Part B, vol. 35, pp1-22, 1999.

- Patankar, S.V. (1980), *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corporation, Washington, D.C., ISBN 0-89116-522-3.

- Rhie, C.M. and Chow, W.L., "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation ", AIAA Journal, Vol. 21, N. 11, pp. 1525-1532, 1983.

- Shen, W.Z., Michelsen, J.A., Sorensen, N.N and Sorensen, J.N., "An Improved SIMPLEC Method on Collocated Grids for Steady and Unsteady Flow Computations", Numerical Heat Transfer, Part B, Vol. 43, pp. 221-239, 2003.

- Steger, J.L. and Sorenson, R.L., "Automatic Mesh-Point Clustering Near a Boundary in Grid Generation with Elliptic Partial Differential Equations", Journal of Computational Physics, Vol. 33, pp. 405-410, 1979.

- Thangam, S and Hur, N. 1991, "A Highly-Resolved Numerical Study of Turbulent Separated Flow Past a Backward-Facing Step", International Journal of Engineering Science', vol. 29, no. 5, pp. 607-615.

- Thomas, P.D. and Middlecoff, J.F., "Direct Control of the Grid Point Distribution in Meshes Generated by Elliptic Equations", AIAA Journal, Vol. 18, N. 6, pp. 652-656, 1979.

- Ting, S and Prakash, M. 2005, "Simulation of High Reynolds Number Flow Over a Backward Facing Step Using SPH", Mathematical and Information Sciences, CSIRO, Technical Report CMIS 05/191.

- Vahl Davis, G.D. and Jones, I.P, "Natural Convection in a Square Cavity: A Comparison Exercise", International Journal for Numerical Methods in Fluids, vol. 3, pp. 227-248, 1983.

- Vahl Davis, G.D., "Natural Convection in a Square Cavity: A Benchmark Numerical Solution", International Journal for Numerical Methods in Fluids, vol. 3, pp. 249-264, 1983.

– Van Doormaal, J.P. and Raithby, G.D., "Enhancements of the Simple Method for Predicting Incompressible Fluid Flows", Numerical Heat Transfer, Vol. 7, pp. 147-163, 1984.

– Vertsteeg, H.K. and Malalasekera, W., “An Introduction to Computational Fluid Dynamics”, Pearson Education Limited, 2007.