Chapter 19
An OpenFOAM$^{®}$ Turbulent Flow Application

Abstract The OpenFOAM$^{®}$ solvers and boundary conditions developed in previous chapters are used here to solve a small scale turbulent incompressible flow problem. Details on the case structure, mesh, and solver setup are presented along with information on monitoring convergence and post processing results.

19.1 Introduction

The design of a car body is a very demanding task that necessitates finding a compromise between the stylistic constraint of the brand and consumer taste on one hand, and the engineering requirements of efficient aerodynamics for improved fuel economy on the other. The use of a CFD tool in this situation is critical, especially in investigating the aerodynamic forces, the interplay of viscous flow effects and turbulent boundary layer, the sensitivity of the flow around the vehicle to changes in its shape, and finally the drag coefficient of the car under various operating conditions.

In this chapter the simpleFoamTurbulent solver and the boundary conditions developed and implemented in previous chapters are used to analyze the flow around a well investigated test case, namely the Ahmed bluff body.

19.2 The Ahmed Bluff Body

In automotive applications, the major factor contributing to drag is the wake that develops behind the vehicle, with its prediction representing a difficult task in CFD. This is so because flow separation in turbulent flows is still a challenge to simulate numerically. Nonetheless determining the size of the flow separation zone greatly influences the predicted drag force acting on the body. Thus, accurate simulation of

F. Moukalled et al., *The Finite Volume Method in Computational Fluid Dynamics*,
Fluid Mechanics and Its Applications 113, DOI 10.1007/978-3-319-16874-6_19

the induced vortex flow and of the separation process is essential for correct predictions of aerodynamic efficiency.

Current automobile designs have many complex geometrical features that make them a challenge to model and/or investigate experimentally. Thus for the limited investigation considered here, the Ahmed bluff body [1, 2], which is schematically depicted in Fig. 19.1, is chosen. Figure 19.1a shows the side, front, and top views from which dimensions can be inferred, while Fig. 19.1b presents a three dimensional visualization of the body. Even though it represents simple geometry, Ahmed body allows for a three-dimensional region of separated flow to be developed and different flow phenomena to be studied and compared to experimental data.

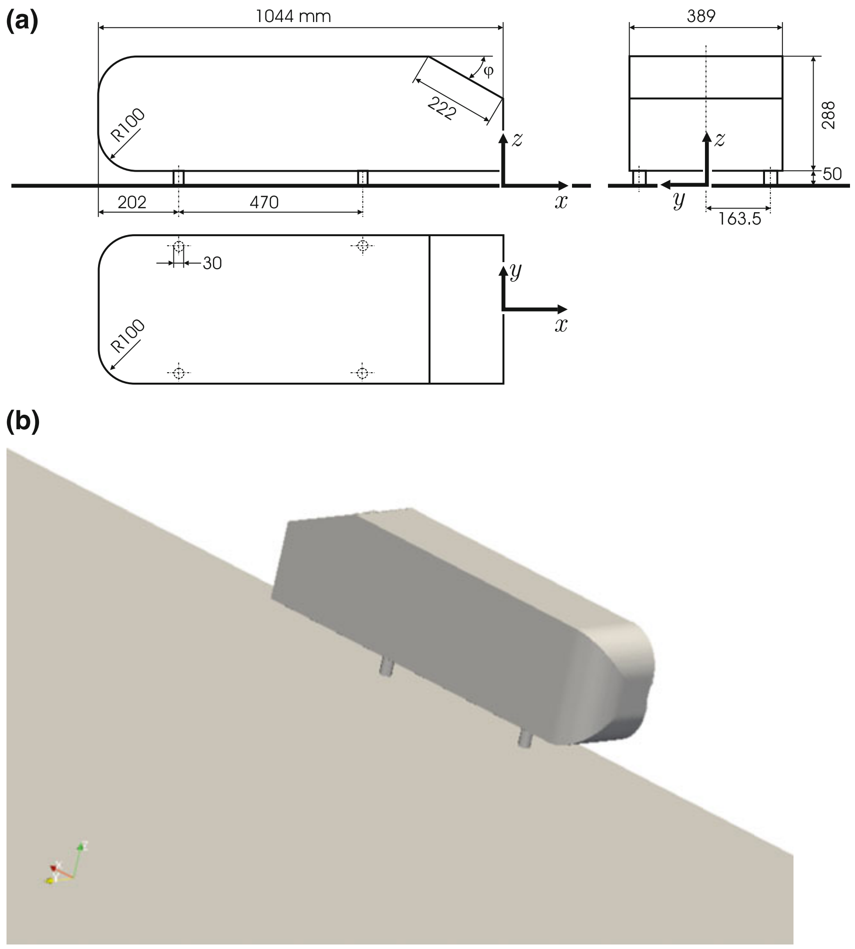

Fig. 19.1 **a** *Side*, *front*, and *top views* of Ahmed body, **b** a three dimensional visualization of Ahmed body geometry

The Ahmed body is a well known configuration and is widely used as benchmark. The slant geometry at its back end generates counter-rotating vortices at the side edges, whose strength is mainly determined by the base slant angle. It comes in two configurations differing in the slant angle value (25° and 35°). In this simulation the configuration with a 25° slant angle is considered.

19.3 Domain Discretization

The computational domain is depicted in Fig. 19.2 with a symmetry condition imposed along the mid section of the Ahmed body shape. Symmetry is exploited to reduce the computational domain and to mitigate the transient flow behavior expected to develop because of vortex shedding at the body's rear end. This should also enhance numerical stability.

Inflow and outflow boundary conditions are placed far from the body to minimize any unwanted interaction with the main flow region, especially between the outlet and the flow at the rear end of the body.

The mesh is generated by *snappyHexMesh*, a utility that is part of the OpenFOAM® [3] package. s*nappyHexMesh* generates three dimensional meshes containing hexahedra (hex) and split-hexahedra (split-hex) elements generated automatically from triangulated surface geometry in the Stereolithography (STL) format. The mesh gradually conforms to the surface by iteratively refining a starting mesh and morphing the resulting split-hex mesh to the surface. An optional phase will shrink back the resulting mesh and insert cell layers. The specification of mesh refinement level is very flexible and the surface handling is robust with a pre-specified final mesh quality. It runs in parallel with a load balancing step at every iteration [4].

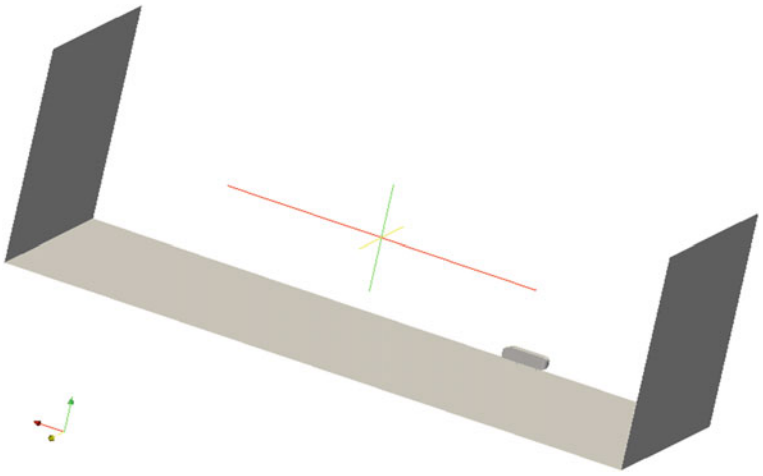

Fig. 19.2 Computational domain for Ahmed body

It is beyond the purpose of this book to give a detailed introduction of the *snappyHexMesh* application but a commented setup file can be found in the case directory "*Ahmed_body/system/snappyHexMeshDict*". Listing 19.1 shows an extract of the file.

```
// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       true;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    ahmed_body.stl
    {
        type triSurfaceMesh;
        name ahmed_body;
        appendRegionName false;
    }
…

// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 300000;

…
```

Listing 19.1 An extract of the snappyHexMeshDict file

The mesh is generated by executing, from the case directory, the following commands in the shown sequence:

blockMesh—dict system/blockMeshDict this defines the general computational domain within which the snappyHexMesh will operate, the domain should have within its volume the STL geometry.

snappyHexMesh **snappyHexMesh** generates the mesh in three phases with the result of each phase written to a folder (1/2/ and 3/).

mv 3/polyMesh constant/ this moves the final mesh generated by snappyHexMesh into the constant directory to be used for computing the solution.
createPatch—overwrite this deletes any empty patches.

The body fitted grid is composed of about 800,000 polyhedral cells mostly located in two regions around the car body. This allows for a good resolution of the wake region while keeping low the number of elements used in the computational domain. Details of the grid generated around the body is shown in Fig. 19.3.

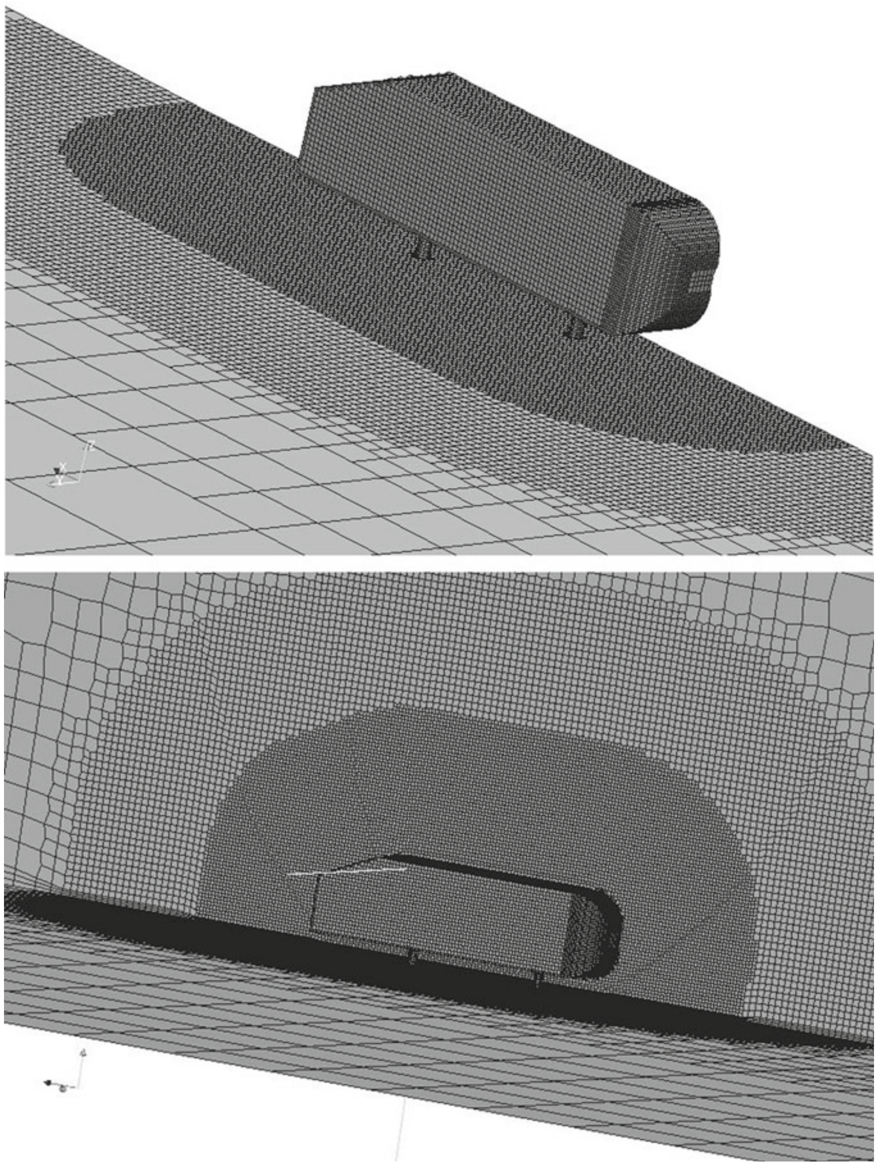

Fig. 19.3 Grid details around Ahmed body

Clustering of mesh elements in the region of interest is a common practice as it reduces the computational cost while properly resolving the important features of the flow.

The generated mesh is fully unstructured and uses polyhedral elements with up to 15 faces. The quality of the mesh is best evaluated using **checkMesh**, the output of such an operation is shown in Listing 19.2.

```
Mesh stats
    points:           806136
    faces:            2331063
    internal faces:   2292117
    cells:            762865
    faces per cell:   6.060285896
    boundary patches: 10
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     744610
    prisms:        1057
    wedges:        0
    pyramids:      0
    tet wedges:    1
    tetrahedra:    0
    polyhedra:     17197
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            4   82
            5   59
            6   4523
            7   1563
            8   410
            9   6961
           10   5
           11   3
           12   2782
           14   2
           15   807

…

Checking geometry...
    Overall domain bounding box (-5 -0.00189057368 0) (10 5 5)
    Mesh (non-empty, non-wedge) directions (1 1 1)
    Mesh (non-empty) directions (1 1 1)
    Boundary openness (-2.589889552e-17 -1.395011501e-15 -4.461826685e-14) OK.
    Max cell openness = 3.962970386e-16 OK.
    Max aspect ratio = 13.98146561 OK.
    Minimum face area = 1.211564662e-06. Maximum face area = 0.1568483398.  Face area
magnitudes OK.
    Min volume = 8.48251732e-09. Max volume = 0.05873313819.  Total volume = 374.944525.
Cell volumes OK.
    Mesh non-orthogonality Max: 62.12745867 average: 5.023035258
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 2.516852961 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

Listing 19.2 Grid quality check and details

The physical boundary conditions are set in the *boundary* file located in the "Ahmed_body/constant/polyMesh/" directory. This is displayed in Listing 19.3.

```
10
(
    inlet
    {
        type                patch;
        physicalType        inlet;
        nFaces              298;
        startFace           2292117;
    }
    outlet
    {
        type                patch;
        physicalType        outlet;
        nFaces              298;
        startFace           2292415;
    }
    ffmaxy
    {
        type                patch;
        nFaces              640;
        startFace           2292713;
    }
    floor
    {
        type                wall;
        inGroups            1(wall);
        nFaces              13779;
        startFace           2293353;
    }
    top
    {
        type                patch;
        nFaces              520;
        startFace           2307132;
    }
    ahmed_body_body
    {
        type                wall;
        inGroups            1(wall);
        nFaces              5750;
        startFace           2307652;
    }
    ahmed_body_body_front_h
    {
        type                wall;
        inGroups            1(wall);
        nFaces              864;
        startFace           2313402;
    }
    ahmed_body_body_front_v
    {
        type                wall;
        inGroups            1(wall);
        nFaces              1270;
        startFace           2314266;
    }
    ahmed_body_stilts
    {
        type                wall;
        inGroups            1(wall);
        nFaces              256;
        startFace           2315536;
    }
    cp
    {
        type                wall;
        inGroups            1(wall);
        nFaces              15271;
        startFace           2315792;
    }
)
```

Listing 19.3 Script used to set the physical boundary conditions

The patches *inlet* and *outlet* are defined as type *patch*, while *cp* is set as a *wall*, rather than a *symmetry plane*. In order to reduce the complexity of resolving the flow, a *slip* boundary condition will be used for the *cp* patch. The "*ahmed_body**" patches that represent the car body are defined as "*wall*", while the remaining patches define the external domain.

19.3.1 Initial and Boundary Conditions

The 0 directory must contain the following four basic files for a turbulent incompressible flow simulation: the velocity U, the pressure p, and for the kEpsilon turbulence model, the turbulent kinetic energy k, and the dissipation rate **ε**.

The file U defines the boundary conditions for the velocity (Listing 19.4).

```
dimensions      [0 1 -1 0 0 0 0];

internalField  uniform ( 40 0 0 );

boundaryField
{
    inlet
    {
        type            surfaceNormalFixedValue;
        refValue        uniform -40;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform ( 0 0 0 );
        value           uniform ( 0 0 0 );
    }
    ffmaxy
    {
        type            slip;
    }
    top
    {
        type            slip;
    }
    ahmed_body_body
    {
        type            noSlipWall;
        value           uniform ( 0 0 0 );
    }
    ahmed_body_body_front_h
    {
        type            noSlipWall;
        value           uniform ( 0 0 0 );
    }
    ahmed_body_body_front_v
    {
        type            noSlipWall;
        value           uniform ( 0 0 0 );
```

Listing 19.4 Script of the U file

```
    }
    ahmed_body_stilts
    {
        type            noSlipWall;
        value           uniform ( 0 0 0 );
    }
    floor
    {
        type            noSlipWall;
        value           uniform ( 0 0 0 );
    }
    cp
    {
        type            slip;

    }
}
```

Listing 19.4 (continued)

The new *noSlipWall* boundary condition is employed to impose the no slip condition at the walls including the body car and the floor. The fluid velocity at inlet is set at 40 m/s. Along the cp plane, a *slip* wall boundary condition is used where the viscous forces are set to zero to mimic a symmetry plane (the slight geometric error due to meshing on patch *cp* does not allow the direct use of a symmetryPlane boundary condition).

The *zeroGradient* pressure boundary condition uses a zero order extrapolation to compute the pressure at the boundary. At the outlet, a Dirichlet boundary condition is applied in order to set a reference pressure (Listing 19.5).

```
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    ffmaxy
    {
        type            zeroGradient;
    }
…
```

Listing 19.5 Script of the p file

For the kEpsilon model, the turbulent intensity and integral length scale are set at the inlet boundary, while the standard wall functions are employed along all the walls automatically (Chap. 17). The k file is shown in Listing (19.6) and the epsilon file in Listing (19.7).

```
boundaryField
{
    inlet
    {
        type            turbulentIntensityKineticEnergyInlet;
        intensity       0.01;
        value           uniform 0.002;
    }
    outlet
    {
        type            zeroGradient;
    }
…
```

Listing 19.6 Script of the k file

```
boundaryField
{
    inlet
    {
        type            turbulentMixingLengthFrequencyInlet;
        mixingLength    0.01;
        phi             phi;
        k               k;
        value           uniform 0.002;
    }
    outlet
    {
        type            zeroGradient;
    }
…
```

Listing 19.7 Script of the epsilon file

19.3.2 Systems Files

The *controlDict* file (Listing 19.8) is configured to perform 500 SIMPLE iterations. Further, in order to use the new *noSlipWall* boundary condition for the velocity, the corresponding library has to be included together with the proper libraries for inlet turbulence boundary conditions. These are set using the *libs* declaration. A specific run-time post processing configuration is also included to monitor iteratively variations in the lift and drag coefficients, which are set using the *functions* declaration. Usually for external aerodynamic applications it is more convenient to monitor loads or forces on the body, rather than the level of residuals, for checking convergence.

```
startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         500;

…

libs (
"libNoSlip.so"
"libincompressibleRASModels.so"
"libincompressibleRASModels.so"
);

functions
(

    forceCoeffs1
    {
        type        forceCoeffs;
        functionObjectLibs ( "libforces.so" );
        outputControl timeStep;
        outputInterval  1;
        log         yes;

        patches     ( "ahmed_body*" );
        pName       p;
        UName       U;
        rhoName     rhoInf;      // Indicates incompressible
        log         true;
        rhoInf      1;           // Redundant for incompressible
        liftDir     (0 0 1);
        dragDir     (1 0 0);
        CofR        (0.72 0 0);  // Axle midpoint on ground
        pitchAxis   (0 1 0);
        magUInf     40;
        lRef        1.45;        // Wheelbase length
        Aref        2.618;        // Estimated
    }

)
```

Listing 19.8 Script of the controlDict file

Linear solvers specifications and relaxation factors are configured in the *fvSolution* file displayed in Listing 19.9. Here preconditioned conjugate gradient methods with incomplete factorization pre-conditioners are chosen for all equations. For the pressure correction equation (*pp*), a symmetric matrix solver is specified due to the laplacian nature of the equation. It is worth mentioning that with a pressure correction formulation there is no need for non-orthogonal correction iterations.

```
solvers
{
    pp
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.01;
    }

    "(U|k|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-15;
        relTol          0.1;
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors 0;
}
relaxationFactors
{
        pp              0.3;
        U               0.7;
        k               0.7;
        epsilon         0.7;
}
```

Listing 19.9 Script of the fvSolution file

Spatial discretization settings are configured in the *fvSchemes* file depicted in Listing 19.10. For this simulation a first order upwind discretization for convection, a Gaussian method for gradient reconstruction, and a linear profile for variable face interpolation are used.

```
ddtSchemes
{
    default         steadyState;
}
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}
divSchemes
{
    default          Gauss upwind;
    div(phi,U)       Gauss upwind;
    div(phi,k)       Gauss upwind;
    div(phi,epsilon)  Gauss upwind;
    div(R)          Gauss linear;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear corrected;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}
snGradSchemes
{
    default         corrected;
}
fluxRequired
{
    default         no;
    pp               ;
}
```

Listing 19.10 fvSchemes file details

19.3.3 Running the Solver

At the start of the run, the solver provides information about equation residuals and force coefficients (Listing 19.11). Based on the controlDict setup, force coefficients are also written in the file Ahmed_body/postProcessing/forceCoeffs1/0/forceCoeffs.dat that can be used for checking the convergence history.

```
Starting time loop

Time = 1

DILUPBiCG:  Solving for Ux, Initial residual = 1, Final residual = 0.02308834211, No
Iterations 2
DILUPBiCG:  Solving for Uy, Initial residual = 1, Final residual = 0.07259752325, No
Iterations 1
DILUPBiCG:  Solving for Uz, Initial residual = 1, Final residual = 0.05706016776, No
Iterations 1
DICPCG:  Solving for pp, Initial residual = 1, Final residual = 0.009688743115, No
Iterations 140
DILUPBiCG:  Solving for epsilon, Initial residual = 0.999533875, Final residual =
2.962105429e-05, No Iterations 1
DILUPBiCG:  Solving for k, Initial residual = 1, Final residual = 0.000587089458, No
Iterations 1
ExecutionTime = 9.73 s  ClockTime = 9 s

forceCoeffs output:
    Cm    = 5.971060664
    Cd    = 18.59033988
    Cl    = 3.888152128
    Cl(f) = 7.915136728
    Cl(r) = -4.0269846
```

Listing 19.11 Solver verbosity

It is worth noting that for the pressure correction equation the residual at any iteration is always 1, which is due to the way OpenFOAM® normalizes residuals (Check the computational pointers in Chap. 10).

Figure 19.4a shows the convergence history plots of the residuals of the momentum equations, while Fig. 19.4b shows the changes in the lift and drag coefficients as the simulation progresses. Results clearly show that in 300 iterations the solution has completely converged with changes in the lift and drag coefficient values becoming negligible.

Figure 19.5 shows the static pressure contours around the body. High pressure values are evident on the front of the body due to the recovery of the dynamic pressure contribution. Contours also show a low pressure region immediately

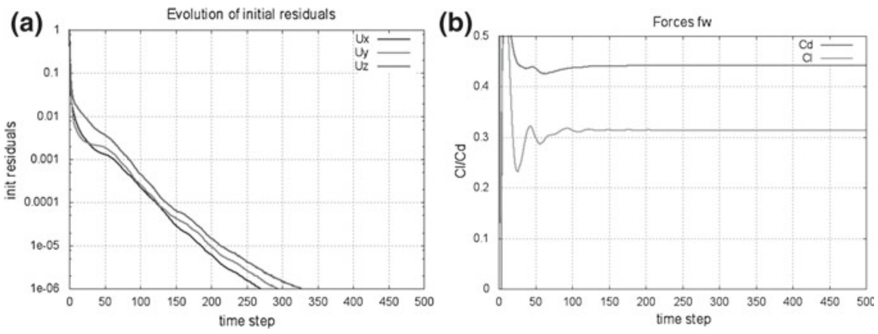

Fig. 19.4 **a** Convergence history plots of the residuals of the momentum equations, **b** variations of the drag and lift coefficients with time

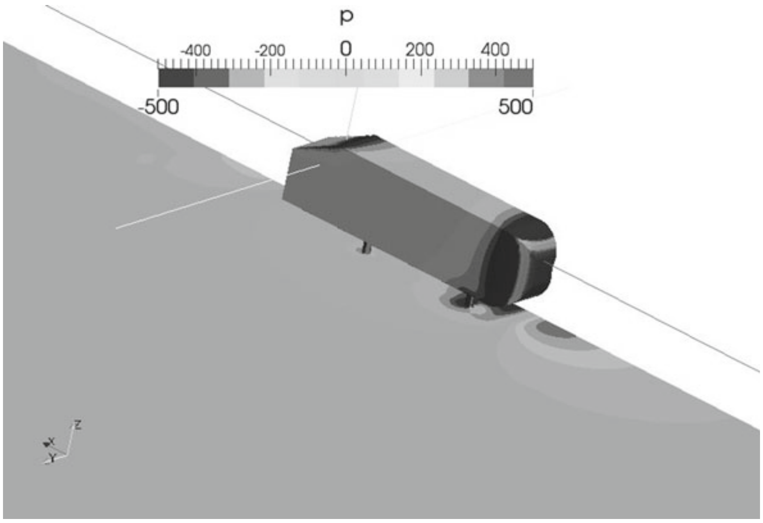

Fig. 19.5 Pressure contours around the body

downstream the front location of the body as a result of the acceleration of the fluid on a curved surface.

In Fig. 19.6 the recirculation region is evident on the back side of the body where a low speed region appears.

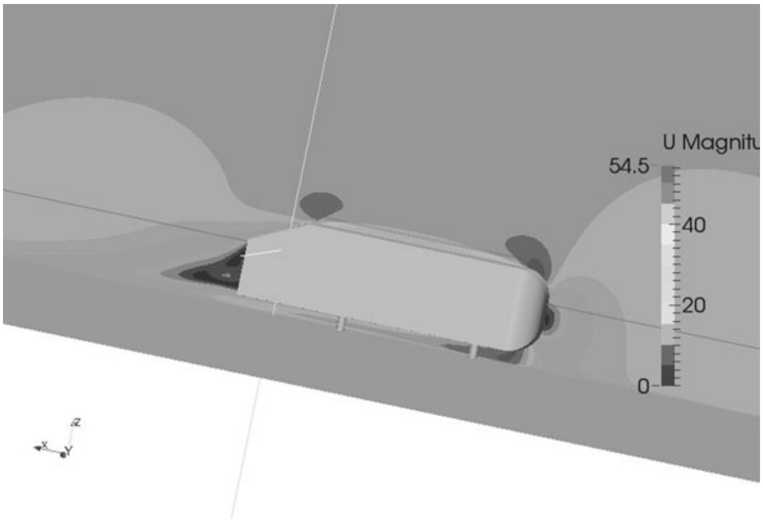

Fig. 19.6 Velocity contours around the body

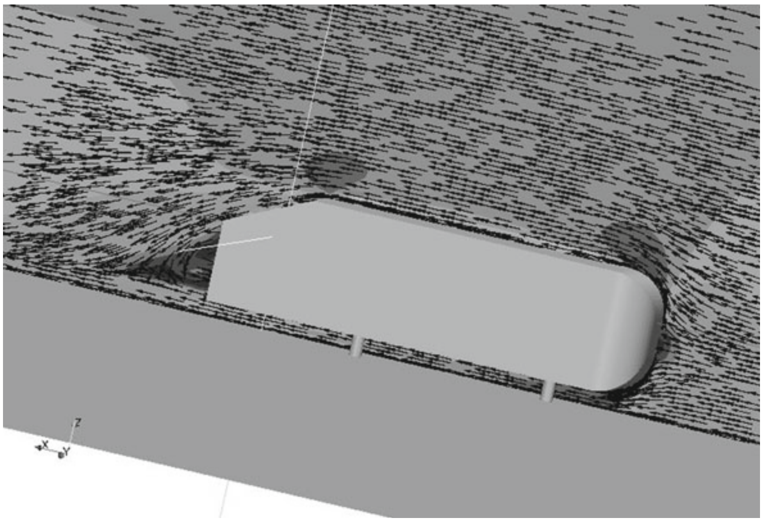

Fig. 19.7 Vector plots around body

The separation region is obvious in the vector plots shown in Fig. 19.7 where a wake is seen to form behind the body. The formation of this vortex region is responsible for pressure losses and is the main contribution to the drag on the body.

19.4 Conclusion

The simpleFoamTurbulent solver and boundary conditions developed in previous chapters were used to solve for the turbulent flow around the Ahmed bluff body. Despite the low order convection scheme and the relatively coarse mesh used, the important features of the flow were demonstrated.

References

1. Ahmed SR, Ramm G, Faltin G (1984) Some salient features of the time averaged ground vehicle wake. SAE Paper 840300
2. Lienhart H, Stoots C, Becker S (2000) Flow and turbulence structures in the wake of a simplified car model (Ahmed Model). In: DGLR Fach Symposium der AG STAB
3. OpenFOAM (2015) Version 2.3.x. http://www.openfoam.org
4. OpenFOAM® User Guide. http://www.openfoam.org/docs/user/snappyHexMesh.php. London 2012, 2014