Technical Report

Cylinder Flow Benchmark with Commercial Software Packages – A Comparative Study

Xiaoqun Feng
Michael Köster
Li Zhang

Lecture "Mathematische Softwarewerkzeuge",
Winter Semester 2004/2005

University of Dortmund, Germany

Contents

1 Introduction

The exercise in this course is to analyse a set of mathematical software tools concerning their speed, accuracy and flexibility. For this purpose a known benchmark problem (cf. [1]) of engineering type is simulated with each software tool and the results are compared with reference values. Our object of investigation is the so-called "flow around a cylinder"-problem, which is widely accepted as standard test for accuracy and performance of mathematical software tools. This problem is a typical test problem for analyzing the behavior of flow behind a rigid body: A circular obstacle is fixed at a special position in a long channel. The obstacle is surrounded by a fluid that is passing through the cylinder.

Depending on the speed of the inflow (or the Reynolds-number, resp.) the flow behind the obstacle shows one of the three prototypical forms of flows: Using a "low" Reynolds-number the flow becomes steady. Using a "mid-range" Reynolds-number the flow becomes unsteady but shows a periodic behavior (the so called "von Karman vortex street"), whereas using "high" Reynolds-number the flow may become completely chaotic. The phenomena of such flow problems are visible everywhere around our living environments such as: flow around high-rise building, the drag force induced by driving car accelerating in the wind, ocean current interaction with the offshore structures, etc. As viewed from engineering, it is important to predict for example the drag- and lift-forces on such an obstacle or the frequency of a vibration at various fluid speeds in order to avoid undesirable resonances in the vibration of the solid structures.

Here we concentrate our investigation on three types of commercial mathematical software tools: FEMLAB, CFX and FLUENT. All three types of software use different mathematical approaches to simulate flow and every software should be able to calculate a more or less accurate solution to the model problem. To be more precise we will concentrate on the following questions:

- How accurate is the solution that can be obtained by a simple modeling?
- How much time will the computation need?
- Is it possible to optimize the results in terms of speed and/or accuracy?
  What possibilities are offered by the software for doing that?
- Are there any major restrictions or drawbacks in the software?
  (Only 2D, only 3D, …)
- How easy is it to set up the benchmark configuration in each software product?
  Is the software easy to handle?

1.1 The Cylinder Flow Benchmark

The cylinder flow benchmark examines incompressible flow past a circular cylinder placed in a channel at right angle to the oncoming fluid. The channel height is H=0.41m, the cylinder diameter D=0.1m (see Figure 1). The fluid properties are defined as follows (cf. [1], [2]):

- viscosity: $\eta = 10^{-3}\ m^2/s$
- density: $\rho = 1\ kg/m^2$

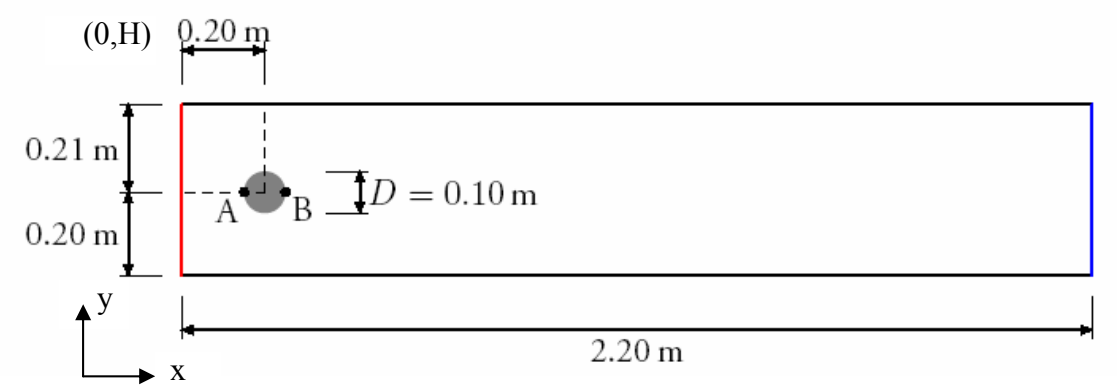

Figure 1: Benchmark geometry

The upper and lower part of the boundary and the boundary of the obstacle itself is fixed as „rigid wall“, i.e. no-slip boundary conditions are used here (u=v=0). The leftmost boundary introduces an inflow with a parabolic inflow profile: Taking U_{max} as the maximum velocity the inflow calculates (depending on the y-coordinate) as

$$u = 4U_{\max} y(H-y)/H^2$$

The rightmost boundary is taken as the outlet and here appropriate boundary conditions have to be prescribed, e.g. by setting the pressure to P=0 together with natural boundary conditions.

It is known that the type of the flow in this configuration is depending on the inflow speed. The crucial number determining the type of the flow is the *Reynolds number*, defined as

$$\mathrm{Re} = \frac{U_{mean} D}{\eta}$$

for the mean velocity

$$U_{mean} = \frac{2}{3} U_{\max}$$

of the parabolic profile. A maximum velocity of U_{max}=0.3 m/s corresponds to Re=20, which will result in a stationary flow, whereas U_{max}= 1.5 m/s corresponds to Re=100 which results in a time dependent flow with vortex shedding.

Quantities to be calculated

The measurement of the accuracy of the benchmark is determined by a comparison of the following values to reference values: We define the drag coefficient C_D and the lift coefficient C_L as

$$c_D = \frac{2F_D}{\rho U_{mean}^2 D} \quad , \quad c_L = \frac{2F_L}{\rho U_{mean}^2 D}$$

Here F_D and F_L are the total forces in x- and y-directions, respectively. Furthermore the **pressure-drop** ΔP (or $\Delta P(t)$ in case of a time-dependent simulation at time t, respectively) is defined as the difference of the pressures in the leftmost and rightmost point of the cylinder (i.e. A=(0.15,0.2), B=(0.25,0.2)):

$$\Delta P = P_A - P_B$$

1.1.1 The stationary case

For the stationary case, the maximal inflow velocity is set to U_{max}=0.3 m/s, which results in a steady flow at Reynolds numbers Re=20. Here c_D, c_L and ΔP are calculated and compared with the reference values. In order to estimate the accuracy of the simulation we have defined the **relative error**

$$E = \frac{\left| V_{simulation} - V^{ref} \right|}{V^{ref}}$$

with $V = c_D$, c_L or ΔP, respectively. The reference values for the stationary simulation are (cf. [5], [3]):

$$c_D^{ref} = 5.57953523384$$

$$c_L^{ref} = 0.010618937712$$

$$\Delta P^{ref} = 0.11752016697$$

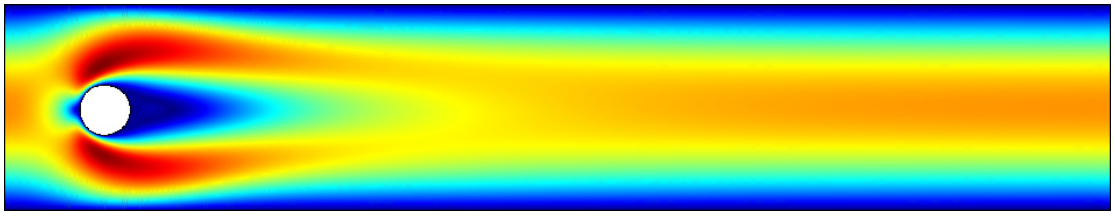

Figure 2: A stationary solution at Re=20

1.1.2 The time-dependent case

The time-dependent simulation is performed in the time-interval I=0-8s with U_{max}=1.5 m/s, which results in a non-steady flow at Reynolds numbers Re=100. Because all the above quantities are time-dependent now, we have to fix a point in time where to calculate the values of interest. It is known that the flow forms a periodic behavior. Therefore it is natural to analyse one *period* of the flow. As guiding variable we fix the lift coefficient. Taking t_0 as the second but last maximum of the lift coefficient in the time interval 0-8s and f as the frequency of the lift, we analyse the time interval $I=[t_0, t_0+1/f]$. More exactly we calculate:

- the maximum drag coefficient $c_{D,max}$ in this time interval
- the maximum lift coefficient $c_{L,max}=c_L(t_0)$
- the pressure difference at the time when the lift coefficient is minimal: $\Delta P := \Delta P(t_0 + 1/(2f))$
- the Strouhal number, defined as

$$St = \frac{D \cdot f}{U_{mean}}$$

- the relative errors of all quantities.

Furthermore the drag- and lift-coefficients are plotted in the time interval I. The reference values for the time-dependent case are (cf. [4]):

$$c_{D,\max}^{ref} = 3.2300$$

$$c_{L,\max}^{ref} = 1.0000$$

$$St^{ref} = 0.3000$$

$$\Delta P^{ref} = 2.4800$$

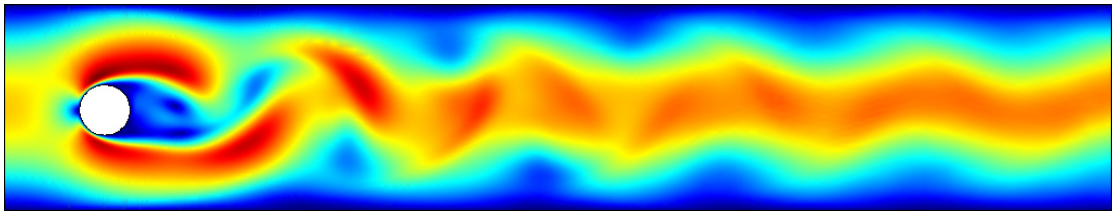

Figure 3: Vortex-shedding in the time-dependent case

1.2 Realization of the benchmark with different simulation software

1.2.1 Femlab

For out simulation we are using Femlab in classical 2D-mode on a Pentium-III PC with 500 MHz. The basic grids that we use are shown in Figure 4. These meshes shown in this Figure are denoted as meshes on level 1. Each of the three grid is regularly refined 2 times, giving level 2 and 3 of each grid.

As Femlab offers the possibility to work with finite elements, we have to prescribe an element type, a finite element formulation and appropriate boundary conditions for this domain. We are using here the standard formulation of the Navier-Strokes equation with the P2/P1 element. On the inlet we prescribe the parabolic profile as mentioned earlier, on the outlet we prescribed P=0. All other boundary components including the obstacle are set to *no-slip* conditions.

Setting up a computation is an interactive step-by-step work in Femlab. In a first wizard one has to define the basic problem definition, which can be extended later. Then after creating the geometry in the geometry generator, one can define the properties of the material in each domain, generate a grid in the grid generator, set up solver parameters, solve the problem and analyse the results in the postprocessor. If something went wrong, one can switch back to any previous step, make corrections and repeat the computation if necessary. All edit fields in Fluent are *interactive* edit fields, i.e. one can not only enter numbers but also expressions. This e.g. allows direct definition of the inflow profile as an expression.

To obtain a value from the solution in the postprocessing, Femlab offers appropriate dialog windows. This allows to print out the solution at any point and even to compute an expression with the help of the solution. Obtaining the values in the instationary case is a little bit more hidden. Either one can select a time step and print out any value of interest in that time step, or one can plot the values of interest as a function in time. The dialog window showing the plot then allows to export the data of all time steps into a text file that can be processed by other programs.

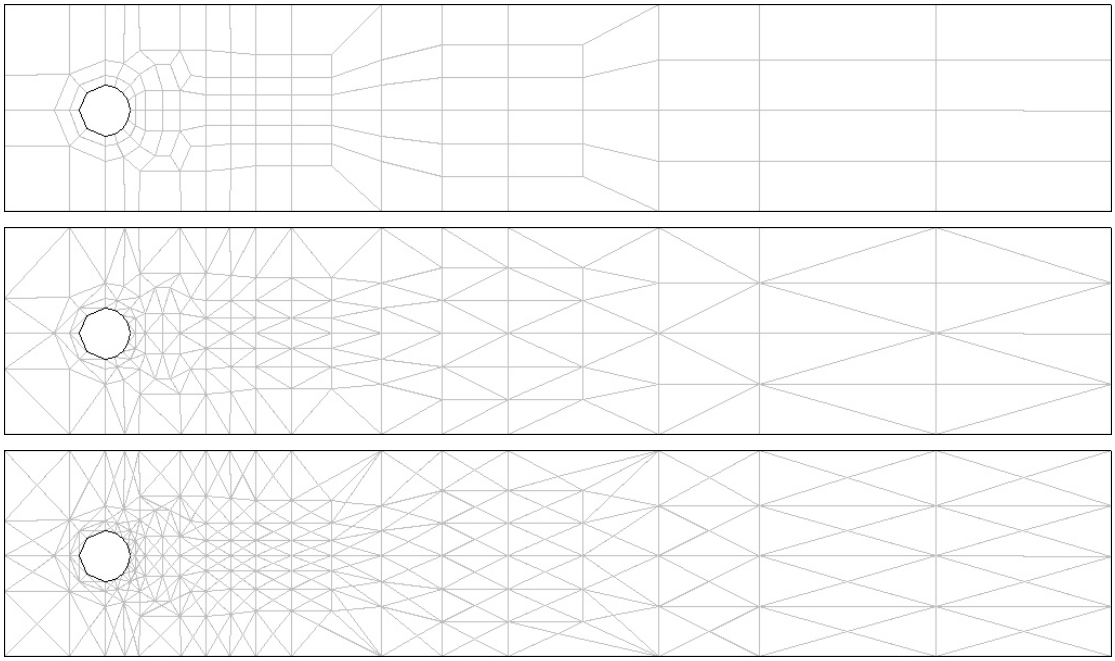

Figure 4: Coarse level 1 grids. From top to bottom: quadrilateral, triangular v.1, and triangular v.2

1.2.2 Fluent

Fluent is a Finite-Volume based software package that supports classical 2D-modelling as well as 3D-modelling. Our computations with this software package are performed on an SUN Ultrasparc 3 V880 with 950 MHz. The software package itself decomposes into two parts: A grid generator named “Gambit” and the solver package “Fluent” itself. Gambit is a standalone mesh generator which can also be used independent to the solver package Fluent. The solver package then combines the solver and postprocessing tool as well as remeshing and boundary-/mesh-adaption tools.

We had not been able to import any of the Femlab 2D reference grids into the Fluent package, as Fluent does not support the Femlab file format. Therefore it was necessary to create grids with Gambit on our own for testing. As Gambit does not support regular refinement of the mesh, we created for the triangle as well as for the quadrilateral case three different grids with different mesh resolutions; in the following we denote these meshes as “Coarse”, “Med” and “Fine”.

The *Coarse* grid was in all cases created with the “Pave” scheme of Gambit using a spacing parameter of 0.01. For the *Med* and the *Fine* grid we used spacing parameters of 0.01 and 0.005. Both the *Med* and the *Fine* grid differ from the *Coarse* grid in that way, that there was a boundary layer created around the circle with different parameters to resolve boundary effects. The different parameters used for the creation of these grids can be seen in Table 1. The triangle grids themselves are depicted in Figure 5, the quadrilateral grids in Figure 6.

To set up a computation in Fluent, one has to create a new project in the solver/postprocessor and load in the desired mesh. Then all configurations about the solution process can be made via the menus. What is a little bit special with the fluent solver is the definition of the inflow profile. It is not possible to directly specify a parabolic inflow profile, only a constant inflow along an edge in a specific direction is realized by default. To specify the desired profile it is necessary to write a small C++-procedure in a predefined plugin-syntax. Fluent compiles this with the system-internal C++-compiler before it can be used.

	Triangle			Quadrilateral		
Parameter	Coarse	Med	Fine	Coarse	Med	Fine
Spacing parameter	0,01	0,01	0,0005	0,01	0,005	0,005
Number of circle segments	-	64	96	-	64	128
Number of boundary layers	-	10	15	-	8	12
Growing factor	-	1,2	1,2	-	1,385	1,2
Number of cells	5131	7150	15180	2213	9591	12284
Number of faces	7846	11208	23730	4565	19475	24893
Number of nodes	2708	4058	8550	2352	9884	12609

Table 1: Data of the triangle and quadrilateral grids in Fluent

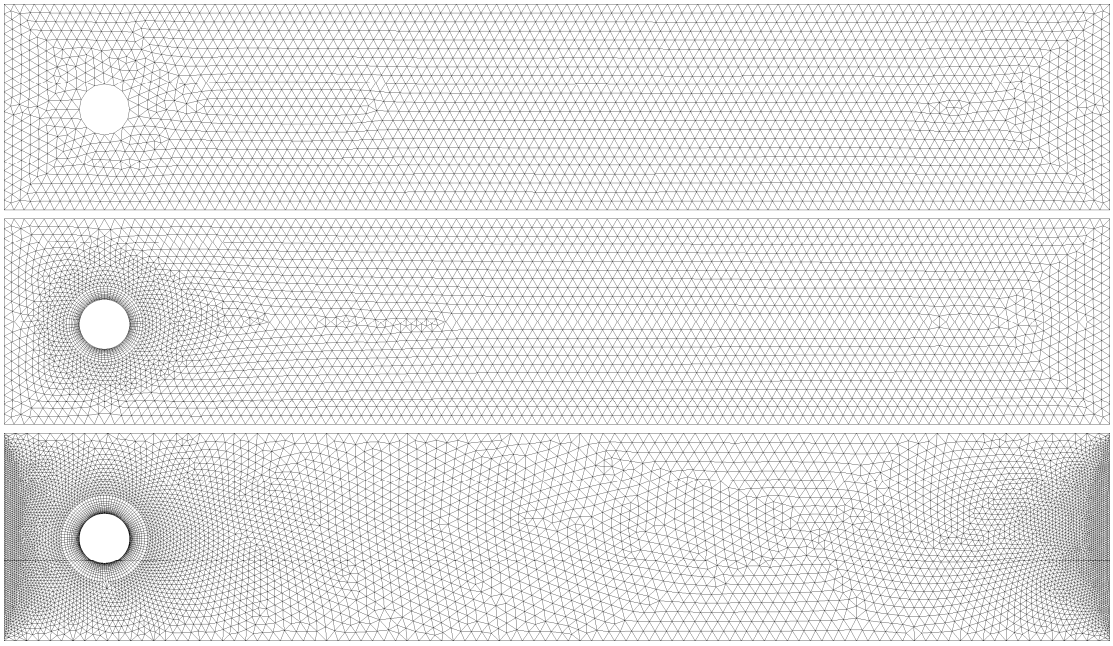

Figure 5: Coarse, Med and Fine triangular grids for Fluent

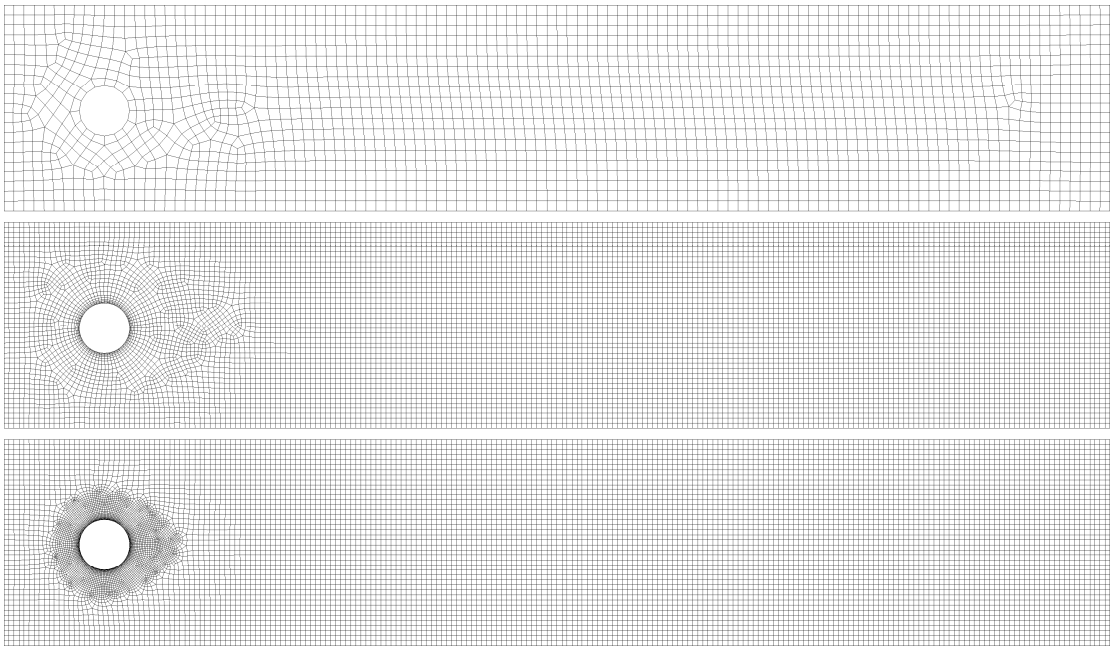

Figure 6: Coarse, Med and Fine quadrilateral grids for Fluent

1.2.3 CFX

CFX can only solve the problem in 3D, it consists four main modules: geometry and mesh generator, preprocessor, solver and postprocessing. The transfer among the modules is through files on the hard disc. So it has the advantage that it is convenient for dividing the work among different people to work in cooperation.

In our tests we restrict to analyze the last three components which are mainly responsible for the computation. Like Femlab our computations are performed on a Pentium-III-PC with 500 MHz. As CFX is not able to work in 2D-mode it has not been possible to include one of the reference grids in here which are used for Femlab. Instead in our tests we restrict to analyze one hexahedral grid which was created with the geometry/mesh generator ANSYS, we sincerely thank Jianjun Feng from the Institute of Energy and Enviromental Engineering of the university Duisburg-Essen for supporting us with that grid. The grid itself is a thin extrusion of a 2D grid into 3D by a depth of 0.001; in Figure 7 a 2D-projection is depicted. As CFX itself (without the grid generator) is not able to do regular refinement to a loaded grid, we have not been able to work on finer versions of this grid.

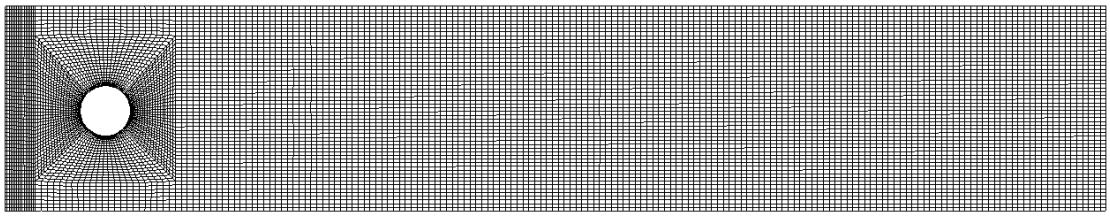

Figure 7: 2D-view of the hexahedral grid used in CFX, size (w/h/d): 2.2 / 0.41 / 0.001

Because this is a 3D-grid, there are some specialties to be considered for a successful computation; these can shortly be summarized by the following:

- A 3D-version contains an additional front and a back wall. On these walls Neumann boundary conditions have to be prescribed to emulate the 2D simulation.
- The definition of the drag- and lift-coefficient changes. The denominator not only has to contain the height of the cylinder but also its depth. The formulas that have to be used here are:

$$c_D = \frac{2F_D}{\rho U_{mean}^2 A} \quad , \quad c_L = \frac{2F_L}{\rho U_{mean}^2 A}$$

  with A = D*depth = 0.1*0.001 = 0.0001 the vertical *face area* of the cylinder.

To set up the computation in CFX, the preprocessor module CFX-Pre has to be used. Here the geometry and the mesh is loaded, the solver-parameters like inflow, outflow, fluid properties and so on are defined. CFX-Pre offers different wizard modules to simplify this process. For our simulation we choose the "Quick-Setup" wizard. This leads step-by-step through loading/importing the mesh, defining one fluid flow, defining inflow, outflow and walls. After the wizard has finished there is some more fine-tuning necessary to adapt the created configuration to our desires:

- In the "Output Control" branch of the problem definition tree we define defined two points (0.15, 0.2, 0) and (0.25, 0.2, 0) where we monitor the pressure.
- We switch off the "Heat Transfer Model" which is activated by default.

- We defined an expression for the inlet profile. By assigning this expression to the inlet the inflow profile was is up.

All parameters are saved in a definition file. Afterwards the solver module has to be started to solve the problem according to the definition file. Here the user can monitor the behavior of the solving process as well as defined monitor functions or behavior of drag/lift-forces in the case of an instationary simulation. The output of the solver can in the final step be analyzed in the postprocessor.

The calculation of drag- and lift-coefficients is a bit tricky. CFX is able to compute the drag- and lift-forces, but not the coefficients. To calculate the coefficients one can either set up an expression or calculate them from the forces "by hand" afterwards. The forces of a steady calculation as well as in one time step of the instationary simulation can be obtained via a menu in the postprocessor. The results in the instationary case can even be exported as set from the solver into a text file.

2 Results in the stationary case

2.1 Femlab

For the analysis of stationary case, we need a nonlinear solver, as the underlying FE-system is a nonlinear one. Such a nonlinear solver typically solves the system by linearising the differential equation. These linear equations are solved by a so called *inner solution engine* which normally is a standard linear solver like Gauss-elimination, CG-algorithm or others. Another so called *outer solution engine*, in which the linear solver is embedded, then treats the nonlinearity. The inner linear solver is by default selected to be the direct Gauss elimination (UMFPACK), but also iterative linear solvers like GMRES will be tested. Femlab allows two different formulations to implement boundary conditions: In a *traditional* way and with the help of a weak formulation, which uses Lagrangian multipliers. We test both formulations as the second approach promises higher accuracy. For all tests we use the stopping criterion "absolute error < 10^{-4}" for the residuals.

2.1.1 Traditional analysis

The traditional analysis implements the boundary conditions in a direct way. The results are listed in table 1. With the higher levels of refinements using the quadrilateral element, the errors of c_D and c_L decrease greatly. The behavior in case of triangular elements is mostly comparable, although using the Tri.v2-grid Femlab calculates wrong results. We have not been able to find out the reason for this.

Grid	Level	Dof	Nel	Solver Time(s)	c_D	c_L	Err_c_D	Err_c_L
Quad	1	1300	130	7	5,951305	0,006035	0,066631	0,431676
Tri.v1	1	1300	260	5	6,756983	0,008867	0,211030	0,165001
Tri.v2	1	2470	520	8	7,317217	0,025822	0,311438	1,431693
Quad	2	4940	520	20	5,640289	0,010774	0,010889	0,014602
Tri.v1	2	4940	1040	18	5,932095	0,010951	0,063188	0,031271
Tri.v2	2	9620	2080	28	4,885225	0,000065	0,124433	0,993879
Quad	3	19240	2080	80	5,592235	0,010931	0,002276	0,029387
Tri.v1	3	19240	4160	66	5,786197	0,012179	0,037039	0,146913
Tri.v2	3	37960	8320	166	5,997457	0,110550	0,074903	9,410646

Table 2: Analysis of stationary case, coefficient of drag and lift. Settings: UMFPACK, weak constraints: off; shape function P2/P1.

2.1.2 Non-ideal weak constraints analysis

In the second step we use non-ideal weak constraints for calculating the drag- and lift-coefficients. The results are depicted in Table 3 and Table 4. From the two tables it can be found that with increasing the refinement the relative errors of c_D, c_L and Delta(p) have greatly decreased, also for the triangular element compared with above traditional analysis. To improve the accuracy, using a higher level of refinement is of course an effective way, but it is at the price of more time for calculating the solution, because theoretically the time of computation using UMFPACK is proportional to the third order of number of unknowns (i.e. DOF).

Grid	Level	Dof	Nel	Solver time	c_D	c_L	Err_c_D	Err_c_L
Quad	1	1348	130	7	5,716438	0,007627	0,024537	0,281755
Tri.v1	1	1348	260	6	5,996847	0,008529	0,074793	0,196812
Tri.v2	1	2518	520	9	5,741367	0,008291	0,029005	0,219225
Quad	2	5036	520	22	5,597229	0,010229	0,003171	0,036721
Tri.v1	2	5036	1040	18	5,599412	0,010398	0,003562	0,020806
Tri.v2	2	9718	2080	30	5,600179	0,010409	0,003700	0,019770
Quad	3	19432	2080	89	5,580811	0,010689	0,000229	0,006598
Tri.v1	3	19432	4160	68	5,580337	0,01074	0,000144	0,011401
Tri.v2	3	38153	8320	156	5,578489	0,010671	0,000188	0,04903

Table 3: Analysis of the stationary case, coefficient of drag and lift. settings: solver: UMFPACK; weak constraints: non-ideal; shape function: P2/P1

Grid	Level	p_Left	p_Right	Delta(p)	Err_Delta(p)
Quad	1	0,145876	0,015361	0,130515	0,110575
Tri.v1	1	0,150716	0,014878	0,135838	0,155872
Tri.v2	1	0,138980	0,014737	0,124243	0,057206
Quad	2	0,136620	0,014856	0,121763	0,036107
Tri.v1	2	0,137828	0,014932	0,122896	0,045742
Tri.v2	2	0,135826	0,014745	0,121081	0,030300
Quad	3	0,133263	0,014689	0,118574	0,008969
Tri.v1	3	0,133848	0,014748	0,119100	0,013443
Tri.v2	3	0,132571	0,014673	0,117898	0,003215

Table 4: Analysis of the stationary case, Delta(p). settings: solver: UMFPACK; weak constraints: non-ideal; shape function: P2/P12.1.3 GMRES Solver

2.1.3 Optimization of computational time

We try to optimize the computational time without much loss of accuracy in the solution. As the computation using non-ideal weak constraints shows the most accurate results, we are using this case as our basic configuration. In order to reduce the solver time, we try different setting of the solver. Normally Femlab uses UMFPACK for solving the linearised equations. Here we try the iterative methods, i.e. we selected GMRES method with ILU-preconditioner. The drop tolerance of ILU-preconditioner is to balance the memory efficiency and preconditioner quality. Table 5 summarizes the solver time in conjunction with different drop tolerance of ILU-preconditioner. We can see that at level 2 of Tri.v1 using the factor of 0,01 and 0,02 the computation can be solved in shorter time compared with UMFPACK, whereas the results of the other tests are not as good as those using UMFPACK as linear solver.

Grid	Level	Drop tolerance (ILU-preconditioner)	Solver Time (s)	Reference time(s) from UMFPACK	Evaluation
Tri.v1	2	0,001	26	18	slower
Tri.v1	2	0,01	17	18	faster
Tri.v1	2	0,02	17	18	faster
Tri.v1	2	0,05	22	18	slower
Tri.v1	2	0,1	21	18	slower
Tri.v1	3	0,001	165	68	slower
Tri.v1	3	0,005	114	68	slower
Tri.v1	3	0,01	132	68	slower
Quad	1	0,01	7	7	no difference
Quad	2	0,01	26	22	slower
Quad	2	0,02	25	22	slower

Table 5: GMRES-Solver with different drop tolerance in the ILU-preconditioner

Other iterative methods such as CG, SSOR or Multigrid are not possible to use because there are 0-blocks / zero on diagonal of the matrix which makes these solvers to fail.

2.1.4 Higher-order shape functions

In the next step we try to use different types of finite elements and finite element pairs, especially with higher order shape functions. Our hope is to analyse if it is possible to obtain the same accuracy with less unknowns and less computational time. Table 6 summarizes the results with the help of weak constraints. It is worthwhile to notice that the accuracy of c_D using P3/P2 at level 2 is roughly the same as Level 3, meanwhile c_L is much better than level 3, even with less unknowns!

Shape function	level	Dof	Solver Time (s)	c_D	c_L	Err_c_D	Err_c_L
P3/P2	1	3140	17	5,599793	0,011067	0,003631	0,042195
P4/P3	1	5712	25	5,573696	0,011015	0,001047	0,037298
P3/P2	2	12000	47	5,580515	0,010580	0,000176	0,003670
P4/P3	2	22084	120	5,580693	0,010460	0,000208	0,014967

Table 6: Different shape functions in Tri.v1, weak constraints: on.

One might wonder why we did not calculate the pressure values using weak constraints. Unfortunately we have not been able to obtain correct pressure values here because of errors in Femlab. To overcome this problem we tried the calculation again without weak constraints, cf. Table 7. Here the accuracy of Delta(p) at level 2(P3/P2) is better than level 2 but not as good as level 3.

Shape function	Level	Dof	Solver Time (s)	p_Left	p_Right	Delta(p)	Err_Delta(p)
P3/P2	1	3068	14	0,150596	0,015054	0,135542	0,153351
P4/P3	1	5616	24	0,188101	0,014760	0,173341	0,474990
P3/P2	2	11856	44	0,135207	0,014735	0,120472	0,025117
P4/P3	2	21892	115	0,135762	0,014653	0,121110	0,030546

Table 7: Different shape functions of Tri.v1, weak constraints: off.

2.2 Mixing different Finite-Element spaces

One possible approach to obtain higher accuracy in less time is the mixing of different finite element spaces in the computational domain. In regions where the mathematical

solution is expected to be smooth, high order finite elements on large cells can be used. This approach approximates the solution very accurately with only *very few* unknowns. In regions where the solution is expected to be non-smooth, low order finite elements with very small cells are preferable to capture fine physical details well.

Femlab does not support the mixing of different Finite-Element spaces in one domain, but it supports the use of different Finite-Element spaces on different domains which can be connected to each other. We try to exploit this fact in that way, that we divide the computational domain into two parts: One square containing the circle is used with a very fine grid and the low-order FE-space P2/P1. The rest of the domain is discretised by large triangles and the high-order FE-space P4/P3. Both subdomains share one common boundary edge, what is automatically detected by Femlab: The user is not able to prescribe anything on the common edge.

As can be seen in Figure 8, although Femlab theoretically supports this type of FE-space-mixing, the computational result of the flow is obviously completely wrong (upper solution). However using the same FE-space in both domains gives the correct result (lower solution). There might be different aspects or details that the user has to take care of when performing this kind of simulation with Femlab. Unfortunately it is not obvious how else such a simulation has to be performed, and so a successful improvement of the results has not been possible for us this way.

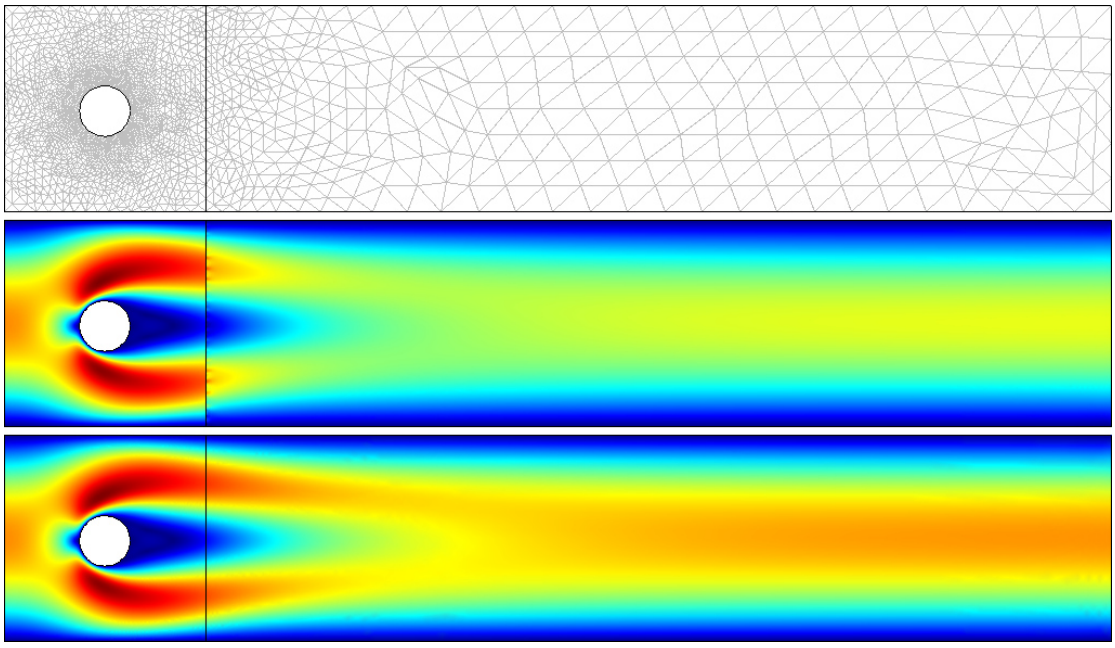

Figure 8: The computational domain, splitted into two sub-domains. The upper solution is calculated with a discretization using different FE-spaces in both sub-domains (P2/P1 in the left, P4/P3 in the right domain). The lower solution is calculated by using P2/P1 in both subdomains.

2.3 Fluent

Fluent basically offers two types of solvers for the stationary as well as for the instationary case: a segregated solver, which solves the governing equations for continuity, momentum and energy separately from each other, and a coupled solver, which solves them in a coupled way. Our tests are all computed with the segregated solver – the standard in Fluent – which is a defect-correction method for solving the nonlinear equations. The linearised equations are solved by an Algebraic Multigrid Method which uses Point-Gauss-Seidel as a smoother.

When using the coupled solver it is also possible to use a geometric Multigrid approach, which obtains the coarse grids by coarsening the fine grid (ignoring the irregular shape of coarse grid cells which arise from the coalescening of groups of fine grid cells), but we don't test this approach here.

For the defect-correction-method used for solving the nonlinear equations, the user has to specify a set of under-relaxation factors for pressure, density, body-forces and momentum. Furthermore parameters for determining the discretization of pressure, pressure-/velocity coupling and momentum equation have to be specified. The exact parameters we used are depicted in Table 8. The embraced numbers in this table are the predefined parameters of Fluent. We changed them after some tests with the Triangle-grids as we figured out that the predefined parameters had been very inaccurate for our problem, as this can be seen later in our tests.

Underrelaxation		**Discretization**		**Stopping criteria**	
Pressure	0,2 (0,3)	Pressure	Standard	Momentum	$< 10^{-4}$ (10^{-1})
Density	1,0	P/V coupling	SIMPLE	X-Velocity	$< 10^{-4}$ (10^{-1})
Body forces	1,0	Momentum	2^{nd}-order Upwind	Y-Velocity	$< 10^{-4}$ (10^{-1})
Momentum	0,5 (0,7)		(1^{st}-order Upwind)		

Table 8: Solver properties for Fluent

Table 9 and Table 10 now depict the results of our stationary tests for the six grids. For comparison reasons the first three lines of these tables show the results Fluent computes using the standard-parameter for the solver. It can clearly be seen that these results are rather inaccurate and can be avoided by a little tuning on the parameters. The results with the tuned parameters are much better. The errors for drag and pressure are < 1% using the *Med* and *Fine*-grids in both the triangle and the quadrilateral case. In contrast to this the error for the lift is much higher in all cases. In the case of triangle grids it has to be noted that using the *Fine*-grid the simulation showed an instationary/oscillating behavior, although the inflow-profile/-velocity only realizes a Reynolds number of 20! This is of course the reason for the large error in the lift in this case.

Grid	**Type**	**Solver time (s)**	**c_D**	**c_L**	**Err_c_D**	**Err_c_L**	**Remark**
Tri	Coarse	8	6.610400	-0.081422	0.184758	8.667622	
Tri	Med	38	6.009600	0.007318	0.077079	0.310844	
Tri	Fine	78	5.802400	0.022550	0.039943	1.123565	
Tri	Coarse	57	5.320200	0.008847	0.046480	0.166875	
Tri	Med	75	5.571300	0.017455	0.001476	0.643761	
Tri	Fine	240	5.571600	-0.020436	0.001422	2.924486	oscillating
Quad	Coarse	20	5.729800	0.000302	0.026931	0.971532	
Quad	Med	63	5.572300	0.008069	0.001297	0.240141	
Quad	Fine	160	5.581700	0.028796	0.000388	1.711759	

Table 9: Drag- and Lift-coefficients; the first three lines show the result using the standard solver parameters, the next 6 lines using the tuned parameter settings.

Grid	Type	p_Left	p_Right	Delta(p)	Err_Delta(p)	Remark
Tri	Coarse	0.144400	0.127840	0.016560	0.859088	
Tri	Med	0.154090	0.017631	0.136459	0.161154	
Tri	Fine	0.135700	0.016470	0.119230	0.014549	
Tri	Coarse	0.110470	0.013731	0.096739	0.176831	
Tri	Med	0.131190	0.014712	0.116478	0.008868	
Tri	Fine	0.131340	0.014523	0.116817	0.005983	oscillating
Quad	Coarse	0.120980	0.013406	0.107574	0.084634	
Quad	Med	0.132090	0.015303	0.116787	0.006239	
Quad	Fine	0.132870	0.015252	0.117618	0.000832	

Table 10: Pressure-results; the first three lines show the result using the standard solver parameters, the next 6 lines using the tuned parameter settings.

2.4 CFX

When creating a new simulation with the Quick-Setup wizard, CFX offers the *Laminar*, *k-Epsilon*, *k-Omega* and *Shear Stress Transport* turbulence model, where k-Epsilon is the predefined one. Switching this to *Laminar* activates the Stationary solver of CFX. CFX uses a coupled solver for the nonlinear terms with an Algebraic Multigrid solver for the linearised terms. We use the standard parameters here, but there is a slight problem when we try to start the solver: CFX complains that our grid does not have any inner points – and rejects to solve the problem. The reason is that our grid was created by an extrusion of a 2-dimensional grid by one cell into the Z-direction, and so it does not have points in the inner of the domain. In the log-file of the solver where the error message appears the solver tells us to modify a special "expert-parameter" of the solver in order to switch of this test for inner points. Such *expert-parameters* can normally be defined in a special menu in the CFX-pre. Unfortunately this one cannot be defined there, we have to manually add this to the solver-definition file with the help of a special menu in the CFX-solver that allows to modify the solver definition file directly!

After these small difficulties CFX successfully solves our problem, the results are depicted in Table 11 and Table 12. The convergence-criterion in this test was the maximum-norm of the residual being $< 10^{-3}$. Obviously the pressure difference matches very well. Also the lift coefficient is not too bad (2 digits after the comma). Unfortunately the drag coefficient does not match very well although the grid is rather fine in our eyes – we have not been able to find the reason for this.

Grid type	Solver time (s)	c_D	c_L	Err_c_D	Err_c_L
Hexahedral	374	6.708700	0.018607	0.202376	0.752209

Table 11: Drag and lift coefficient in CFX

Grid type	p_Left	p_Right	Delta(p)	Err_Delta(p)
Hexahedral	0.128150	0.015216	0.112934	0.039025

Table 12: Pressure difference in CFX.

3 Results in the time-dependent case

3.1 Femlab

The time-dependent simulation is realized in Femlab with a nonlinear, time-dependent solver. For the time-discretization we choose a resolution of 0.01 seconds per time-step. We remark that this is not the real time-step Femlab is using! Femlab uses adaptive time-stepping and interpolates between the calculated solutions to obtain the results in the prescribed time-steps.

3.1.1 Traditional analysis

In the first set of tests we want to analyse the accuracy and speed of the Femlab Finite Element simulation tool using the traditional formulation without Lagrangian multipliers. Table 13 shows the solver-time for the whole simulation in conjunction with the drag- and lift-coefficients as well as the relative errors. Furthermore Table 14 shows the appropriate frequency, Strouhal numbers and pressure difference.

Grid	Level	Solver time (s)	c_{Dmax}	c_{Lmax}	Err_c_{Dmax}	Err_c_{Lmax}
Quad	1	353	3,087071	0,382241	0,044250	0,617759
Tri.v1	1	358	4,167796	0,919739	0,290339	0,080261
Tri.v2	1	610	4,819648	0,769095	0,492151	0,230905
Quad	2	1320	3,518426	1,080904	0,089296	0,080904
Tri.v1	2	1336	3,518159	1,080510	0,089213	0,080510
Tri.v2	2	2825	2,793545	0,955652	0,135126	0,044348
Quad	3	6252	3,338827	0,993574	0,033693	0,006426
Tri.v1	3	3157	3,338827	0,955652	0,033693	0,044348
Tri.v2	3	14961	4,104357	0,906472	0,270699	0,093528

Table 13: Solver-Time and Drag-/Lift-coefficients in the time-dependent case. Traditional formulation.

Grid	Level	f	St	Delta(p)	err_St	err_p(t)
Quad	1	3,030303	0,303030	2,340160	0,010101	0,056387
Tri.v1	1	2,857143	0,285714	2,666303	0,047619	0,075122
Tri.v2	1	3,076923	0,307692	2,544708	0,025641	0,026092
Quad	2	2,941176	0,294118	2,605523	0,019608	0,050614
Tri.v1	2	2,941176	0,294118	2,604910	0,019608	0,050367
Tri.v2	2	3,030303	0,303030	2,551084	0,010101	0,028663
Quad	3	3,030303	0,303030	2,511036	0,010101	0,012515
Tri.v1	3	3,030303	0,303030	2,524167	0,010101	0,017809
Tri.v2	3	3,030303	0,303030	2,483642	0,010101	0,001468

Table 14: Frequency, Strouhal number and pressure difference in the time-dependent case. Traditional formulation.

Obviously the computed values on level 1 are far away from the reference values, but increasing the refinement level quickly reduces the error in most cases. On level 2 the relative error is mostly <0.10 for all computed numbers, on level 3 it is even <0.05. The best approximation properties concerning the drag-/lift-coefficients can be obtained here using a triangulation with quadrilaterals – the relative error in the lift coefficient here is even <0.01 on level 3. Concerning the pressure, the Tri.v2-grid seems to produce the best results.

One remark to the above numbers must not be concealed here: It seems that there are problems evaluating the drag- and lift-coefficients with the Tri.v2-mesh, as the error in these numbers grow dramatically when increasing the level from 2 to 3, what is not expected to be. We have not been able to decide whether this is a computational error or a program error. At least we have to note that Femlab has program errors evaluating the drag-/lift-coefficients using standard boundary integration. We had to manually change the sign of the integral values on some parts of the circle domain before adding them. Femlab seems to be confused about the orientation of the inner domain boundary parts, which are partially clockwise and anticlockwise. Therefore the drag- and lift-values calculated here have to be taken with care!

3.1.2 Non-ideal weak constraints analysis

In the next step we try to enhance the accuracy of the solver by using (non-ideal) weak constraints and Lagrangian multipliers. Table 15 depicts the solver time and the drag-/lift coefficients in the time-dependent simulation as well as the relative errors of the coefficients.

It can easily be seen that the accuracy using this formulation is much better than the traditional formulation, although the time that was necessary to perform the simulation is only slightly higher than in the traditional case. The simulation on level 1 is still somehow inaccurate; this can e.g. be seen in the relative error of the lift, which is about 0.61. On level 2 the calculated values are much better, having a relative error of <0.03 in the drag and <0.07 in the lift. The tri.v2-grid here offers the best accuracy (especially for the lift), but at the price of a much higher computational time.

The calculations on level 3 offer even a high accuracy with relative errors mostly <0.01. The price for this is the high computational time of about 4 hours!

Grid	Level	Solver time (s)	c_{Dmax}	c_{Lmax}	Err_c_{Dmax}	Err_c_{Lmax}
Quad	1	380	3,001042	0,386391	0,070885	0,613609
Tri.v1	1	416	3,806156	0,894009	0,178376	0,105991
Tri.v2	1	702	3,377294	0,722504	0,045602	0,277496
Quad	2	1374	3,298724	1,062995	0,021277	0,062995
Tri.v1	2	1412	3,298538	1,062279	0,021219	0,062278
Tri.v2	2	2490	3,267366	0,992351	0,011568	0,007649
Quad	3	7258	3,230735	0,989881	0,000228	0,010119
Tri.v1	3	6519	3,230340	0,987797	0,000105	0,012203
Tri.v2	3	14434	3,242434	1,004005	0,003850	0,004005

Table 15: Solver-Time and Drag-/Lift-coefficients in the time-dependent case. Lagrangian multipliers used.

Grid	Level	f	St	Delta(p)	err_St	err_p(t)
Quad	1	3,030303	0,303030	2,339557	0,010101	0,056630
Tri.v1	1	2,898551	0,289855	2,645322	0,033816	0,066662
Tri.v2	1	3,076923	0,307692	2,544952	0,025641	0,026190
Quad	2	2,941176	0,294118	2,603325	0,019608	0,049728
Tri.v1	2	2,941176	0,294118	2,602205	0,019608	0,049276
Tri.v2	2	2,985075	0,298507	2,556127	0,004975	0,030696
Quad	3	3,030303	0,303030	2,509173	0,010101	0,011763
Tri.v1	3	3,030303	0,303030	2,508964	0,010101	0,011679
Tri.v2	3	2,985075	0,298507	2,500751	0,004975	0,008367

Table 16: Frequency, Strouhal number and pressure difference in the time-dependent case. Lagrangian multipliers used.

Table 16 now shows the frequency, Strouhal number and pressure difference that are calculated with the grids on the different levels. In all cases (even at level 1) the relative errors are < 0.07. Again the tri.v2-grid gives the best overall results at level 3 as the relative error here is even < 0.01, but again at the highest price.

We also show the behavior of the drag, the lift and the pressure difference as a function of time. For this purpose we plot this information in the time interval of the last simulated period of the lift on level 3, for each grid on level 1-3, for the traditional calculation method as well as for the case of weak constraints. The diagrams in Figure 9 to Figure 11 show two things: At first the formulation with weak constraints captures the behavior of the system much better than the traditional calculation. At second the Tri.v2-grid seems to capture the behavior best. The curves of level 2 and 3 concerning drag/lift are nearly identical without any considerable difference in their phase. Also the curves of the pressure difference are most close to each other here, although in the case of the pressure there can hardly be seen and difference between the traditional calculation and calculation with weak constraints. Altogether this shows that Tri.v2-grid captures most information of the dynamical behavior already at level 2, whereas the other grids need one more level of refinement. Taking the number of degrees of freedom into account, one can see that the number of degrees of freedom of Tri.v2 on level 2 is about the average of DOF of Tri.v1/quad between level 2 and 3, so the use of this grid seems preferable.

3.1.3 Optimization of computational time and accuracy

In a last test we try to apply our knowledge about possible better choices from the stationary case to the time-dependent case. We have seen that the computation on level 2 with the element pair P3/P2 gave about the same accuracy in the solution as level 3. Furthermore the computational time could be decreased by using a GMRES-solver with ILU-preconditioner. Unfortunately using the P3/P2 element and/or the GMRES solver with ILU preconditioner (dropping tolerance 0.2, 0.1, 0.005 tested) lead to stability problems in the time-dependent case. Already when using the P3/P2 element type with the tri.v2-grid on level 1, the solver is not able to solve for more than 0.074 seconds of simulation time. Therefore a proper optimisation with different settings like in the stationary case is not possible here.

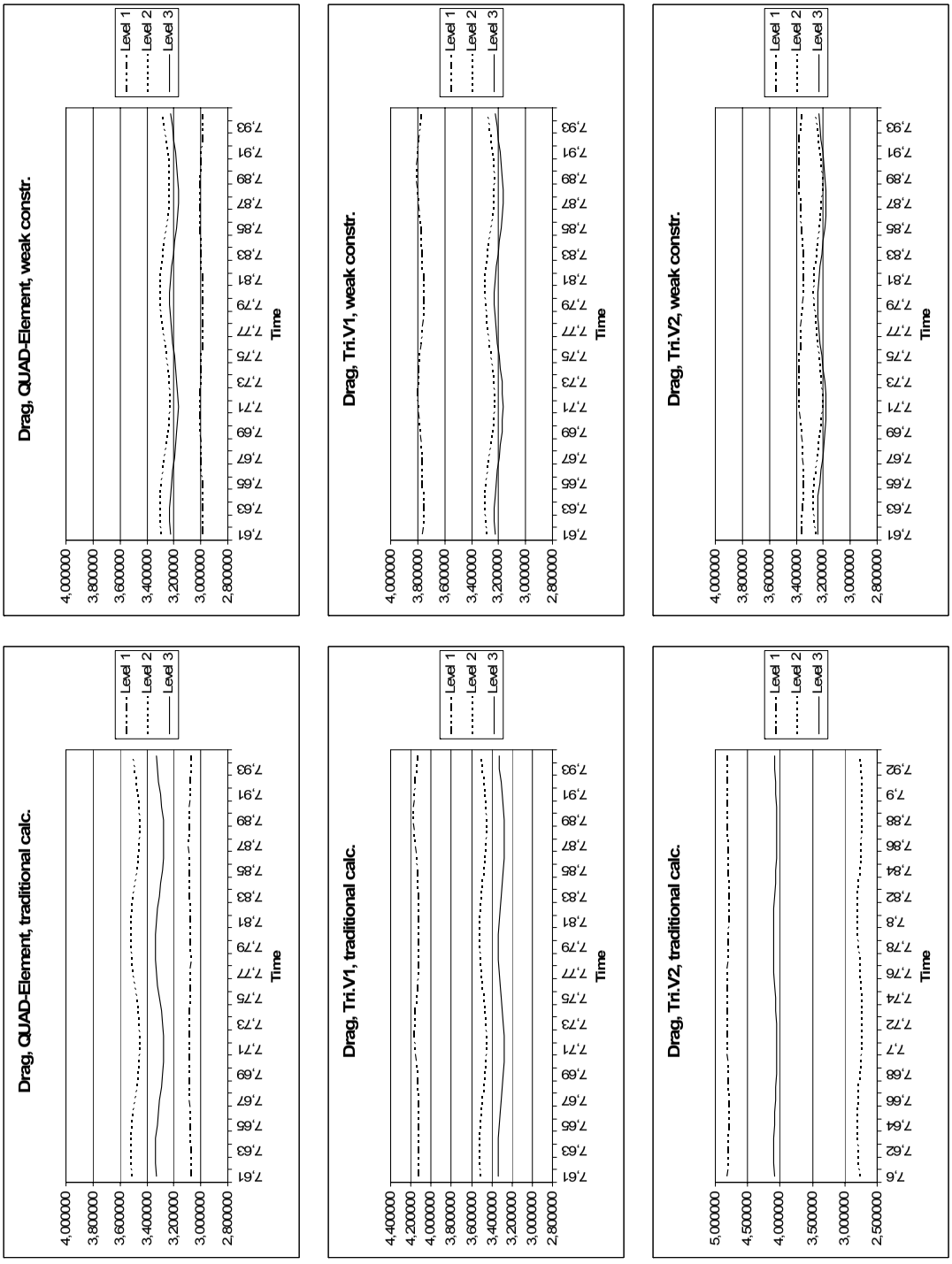

Figure 9: Dynamical behaviour of the drag in the last simulated period

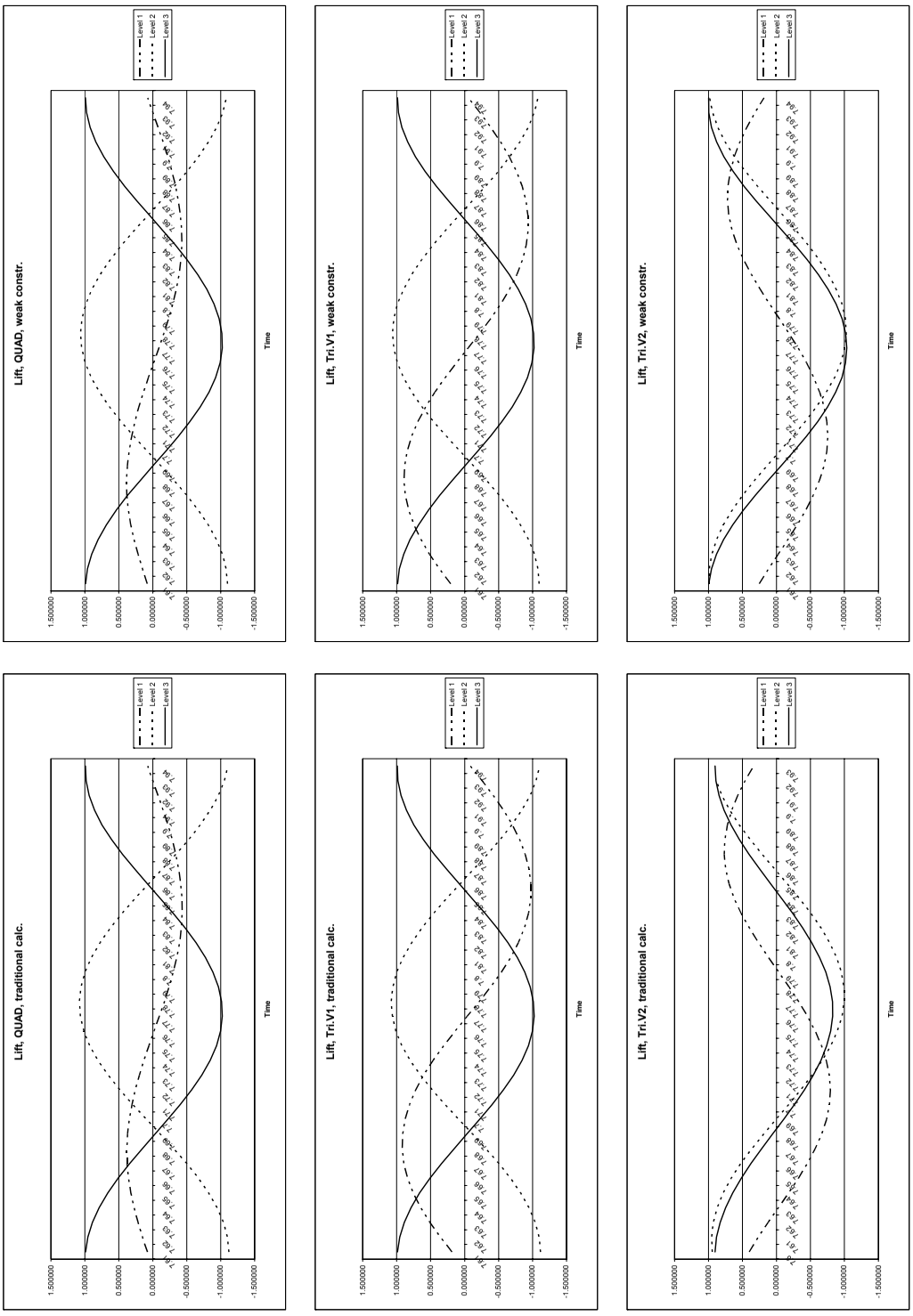

Figure 10: Dynamical behaviour of the lift in the last simulated period

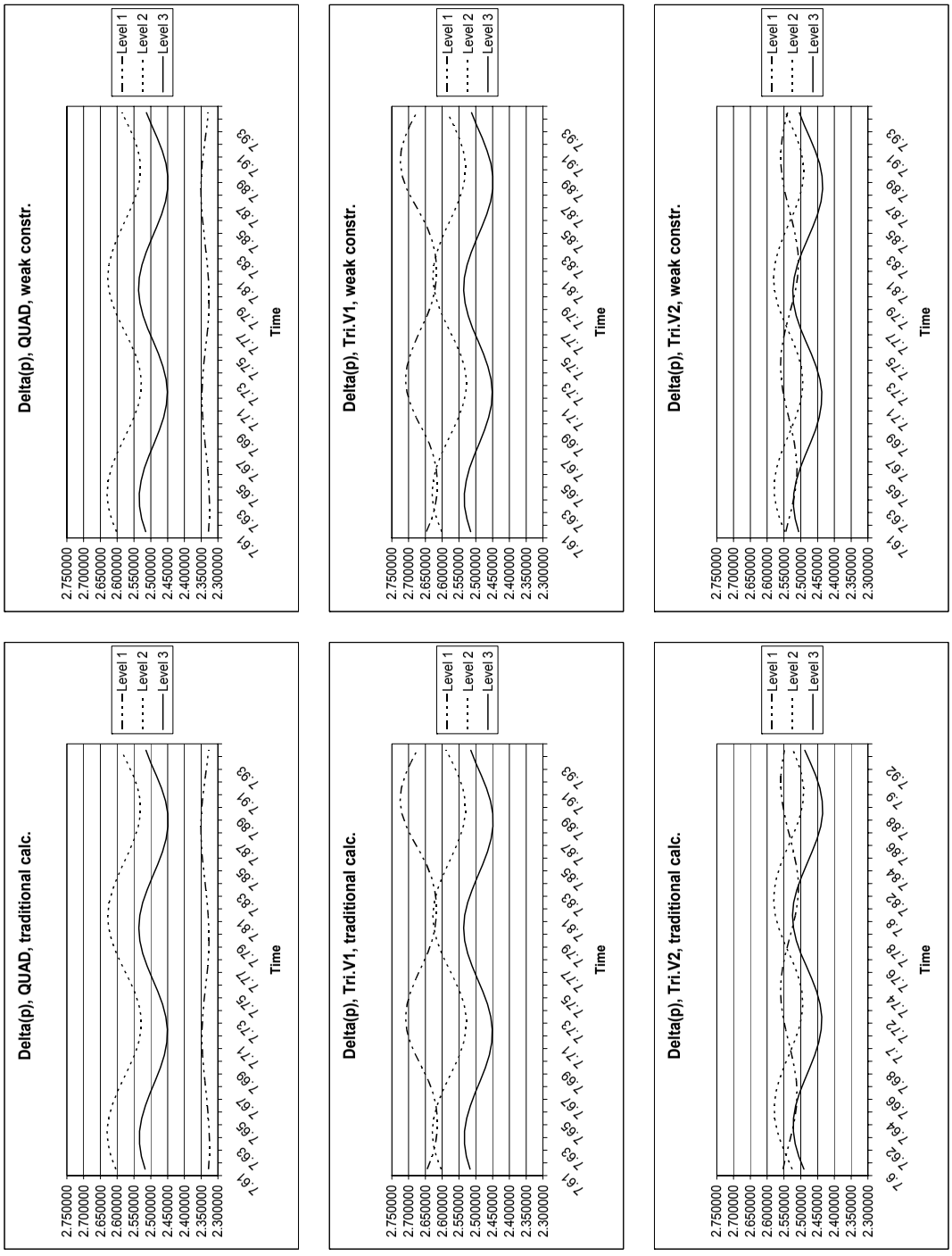

Figure 11: Dynamical behaviour of the pressure difference in the last simulated period

3.2 Fluent

For the instationary simulation with Fluent we choose the instationary solver with the same solver parameters as described in the section about the stationary simulation. Here the choices about the underrelaxation factors and the discretization are even more decisive than in the stationary case: Using the standard parameters results in a stationary flow, although the Reynolds-number is 100!

Table 17 and Table 18 now depict the results in the time-dependent case. For the time-discretization we choose a time-step of 0.01 seconds with a maximum of 10 calculations per time-step. Fluent uses a fixed time-stepping scheme. The errors in the drag are (apart of the *Coarse*-grid) < 2%, the errors in the lift < 15%, the error in St-number and pressure-difference < 5%. It is conspicuous that in both the triangular and the quadrilateral case the *Fine*-grid does not show the most accurate results. We assume this to be an effect of an improper chosen boundary layer in the grid, but we have not been able to figure out this for sure.

Grid	Type	Solver time (s)	c_{Dmax}	c_{Lmax}	Err_c_{Dmax}	Err_c_{Lmax}
Tri	Coarse	1800	2.46554790	0.25959188	0.23667248	0.74040812
Tri	Med	2400	3.20722980	0.95372046	0.00704960	0.04627954
Tri	Fine	4560	3.18958340	0.87247853	0.01251288	0.12752147
Quad	Coarse	1634	3.59253120	0.33654044	0.11223876	0.66345956
Quad	Med	3560	3.20051500	0.90896231	0.00912848	0.09103769
Quad	Fine	5910	3.18268480	0.87978273	0.01464867	0.12021727

Table 17: Solver-Time and Drag-/Lift-coefficients in the time-dependent case.

Grid	Type	f	St	Delta(p)	Err_St	Err_p(t)
Tri	Coarse	1.92307692	0.19230769	1.81090527	0.35897436	0.26979626
Tri	Med	2.94117690	0.29411769	2.40726079	0.01960770	0.02933033
Tri	Fine	2.94117734	0.29411773	2.42209929	0.01960755	0.02334706
Quad	Coarse	2.63157825	0.26315783	2.20621880	0.12280725	0.11039565
Quad	Med	2.94117734	0.29411773	2.40289673	0.01960755	0.03109003
Quad	Fine	2.85714163	0.28571416	2.46600899	0.04761946	0.00564154

Table 18: Frequency, Strouhal number and pressure difference in the time-dependent case.

Figure 12 finally shows the drag-coefficient, lift-coefficient and pressure-difference as a function of time. The depicted time interval is chosen such that the last oscillation of the lift is visible. While the drag- and lift-coefficients show a rather symmetrical behavior, the pressure difference behaves rather unsymmetrical for all grids in all cases, although the relative error in the pressure difference that is shown in Table 18 indicates a rather accurate behavior.

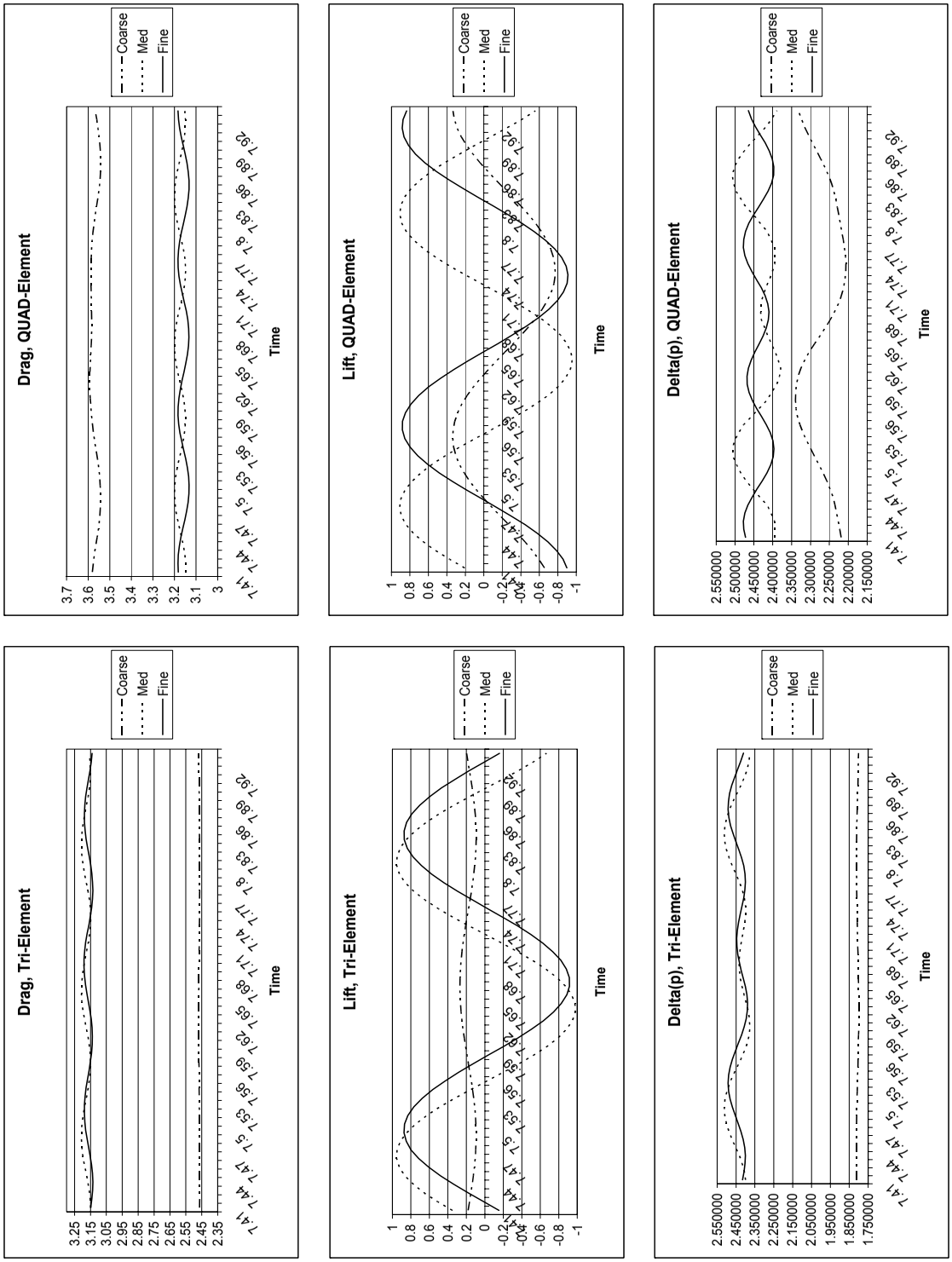

Figure 12: Dynamical behaviour of the drag- and lift-coefficient as well as the pressure difference in the last simulated period

3.3 CFX

For the time-dependent simulation in CFX we choose the *Shear-Stress-Model* as the turbulence model, 2^{nd}-order backward Euler as transient scheme and *high-order advection scheme*. CFX uses fixed time-stepping, we use a time-step of 0.02 seconds with less than 5 iterations per time step. As convergence-criterion we choose the maximum-norm of the residual being $< 10^{-3}$ like in the stationary case. Unfortunately CFX is not able to compute the result accurate enough to form a vertex-shedding behind the cylinder: The resulting flow is stationary! So we use a small trick to *initiate* the vortex-shedding:

For the first 0.2 seconds of simulation time we dynamically rotate the cylinder with 20°/second in clockwise direction; this is possible by defining appropriate expressions and assigning it to the *angular-velocity* parameter that is assigned with the boundary component defining the cylinder. In the next 0.2 seconds we turn the cylinder back by -20°/second. For the rest of the simulation time we do not move the cylinder anymore.

The result of the computation can be seen in Table 19, Table 20 and in Figure 13 as a function of time. Both the drag- and the lift-coefficient show a rather large error of about 20-25%. Again the pressure difference is met quite well with an error of < 2%.

Grid type	Solver time (s)	c_{Dmax}	c_{Lmax}	Err_c_{Dmax}	Err_c_{Lmax}
Hexahedral	14444	2.49620000	0.93354000	0.22718266	0.06646000

Table 19: Solver-Time and Drag-/Lift-coefficients in the time-dependent case

Grid type	f	St	Delta(p)	Err_St	Err_p(t)
Hexahedral	2.70270270	0.27027027	2.44234001	0.09909910	0.01518548

Table 20: Frequency, Strouhal number and pressure difference in the time-dependent case

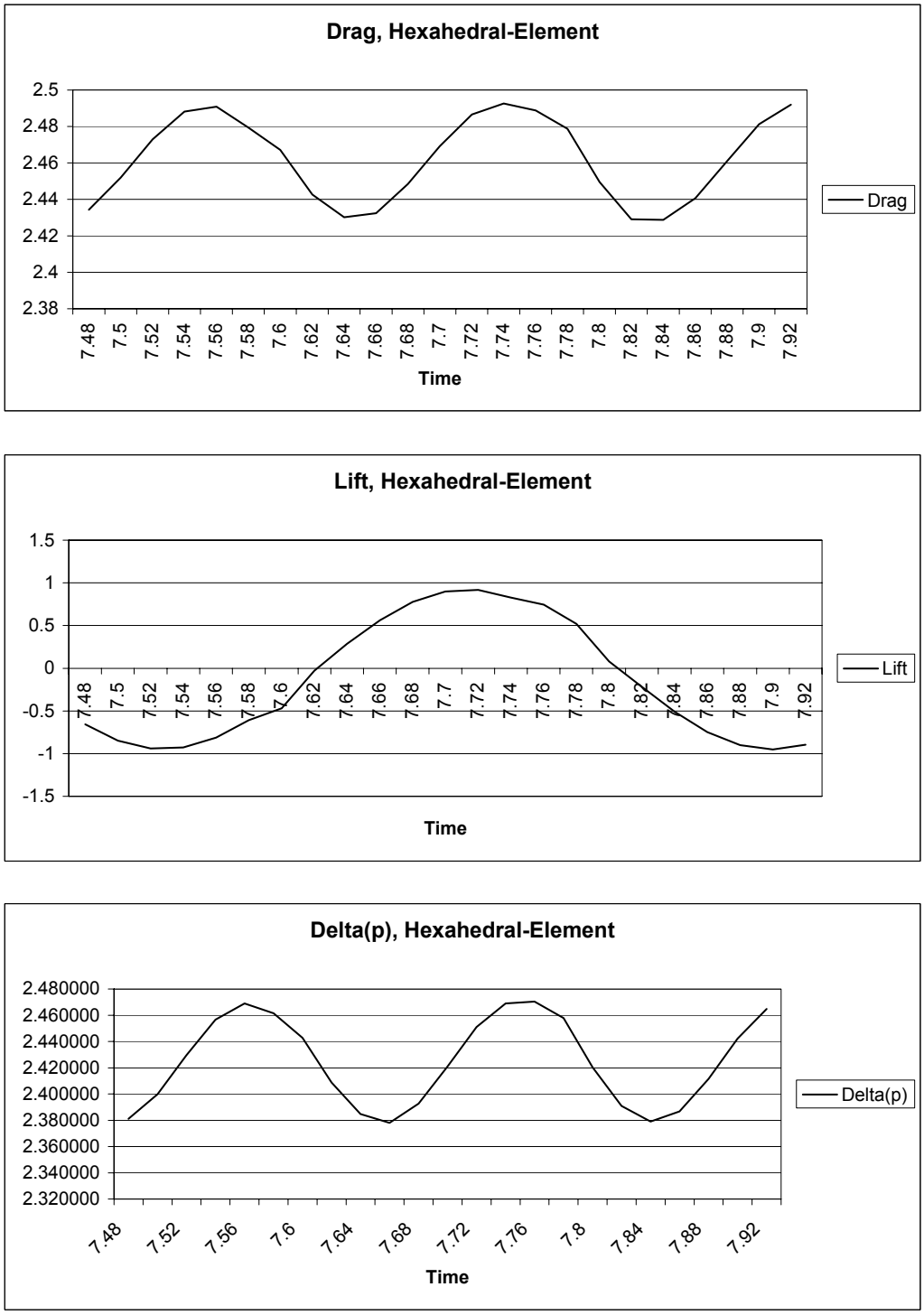

Figure 13: Dynamical behaviour of the drag- and lift-coefficient as well as the pressure difference in the last simulated period

4 Conclusion and Remarks

4.1 Femlab

The Femlab package normally acts on classical 2D, but can be used for 3D-simulation, too. It consists of a geometry creator, a mesh generator, a solver and postprocessing tool, all integrated together into one application environment. For simulation the software uses the finite element approach and allows the weak formulation of the problem to be entered in a very general way with a flexible user interface. Different element types and a large variety of boundary conditions and weak formulations for a lot of types of PDE's are supported. The software offers solvers for linear, non-linear and time-dependent simulations and lets the user configure the parameters of each solver. The computed results are the most accurate ones, especially when the formulation with Lagrangian multipliers is used. The terminology that is combined with the use of modern FEM-techniques is realized in a very user-friendly and intuitive way. Furthermore all results and numbers in Femlab are calculated dimensionless. This makes it easier for inexperienced users to use the software – more experienced users have to transform the calculated numbers into the correct unit system if necessary.

The greatest drawbacks of this software are in our opinion the sometimes slow JAVA-GUI as well as the use of non-specialized solvers. Although the software supports many different types of solvers with very many properties, for many computations the user has to switch back to the general Gauss-elimination-type solvers as most solvers cannot be directly applied to saddle-point-problems which appear for example when solving the Navier-Stokes-equations. Decoupling the equations seems not to be possible as this would destroy the very general approach of this software. Apart of this the software still seems to have problems with the orientation of line-integrals as well as the mixing of finite-element spaces in one domain. Furthermore while an instationary computation is running the user cannot observe any results numerically, only a visual picture of the flow field is updated dynamically. For analyzing any numerical data one has to stop the calculation, observe the data and restart afterwards.

4.2 Fluent

Fluent as well as Femlab directly supports classical 2D-simulations, but in contrast to Femlab, Fluent is based on the Finite-volume approach. It consists of a solver and postprocessing software component and a mesh-generator component, which both are separated.

The mesh generation tool *Gambit* is a rather powerful mesh generator for 2D- and 3D-meshes/domains. It supports triangle elements as well as quadrilateral elements and is also able to mix them in one domain, e.g. when creating boundary layers around objects. Unfortunately the software lacks on a very basic grid generation technique: Refinement of an existing grid. It is not possible to perform uniform refinement as well as selective refinement of elements in a special area. The user is completely restricted to the use of the automatic mesh-generation tool and has to completely re-mesh the domain in order to get a finer mesh.

The solver and postprocessing tool seems to be a bit spartanic and less user friendly at a first glance as the layout presents a text- and command-line with a menu bar. Fortunately all necessary functions are accessible via the menu in a graphical user interface, although some functionality is hidden behind a less intuitive menu items (e.g. the stopping tolerances of the solver is hidden in the menu "*Monitors*", or if point values should be monitored, one has to define a monitor function which sums up values in all points on a surface line-component,

where the surface component only consists of one point). Unfortunately Fluent does not support to enter expressions like it is possible in Femlab and CFX, which makes the software a bit more inflexible.

As both the solver and the postprocessing tool are combined in one software component, this allows the user to monitor all data numerically (by the output of monitored values in the text window) as well as graphically in multiple windows while the computation is running. Unfortunately Fluent seems to get confused with the windows sometimes: When defining an upper/lower bound for the Y-axes of the lift-window, the same bounds were taken for the drag-window.

Concerning the accuracy of the solver, Fluent is able to compute rather accurate results with our self-created grids – at least if we tune some solver parameters and prescribe the correct reference values! The predefined parameters for the solver/preconditioner are too conservative in our opinion, as the results of the time-dependent calculation resulted in a stationary flow until we changed to a higher-order discretization method with lower damping parameters. On the other hand using these slightly tuned parameters, a clearly stationary flow showed slight oscillations using the same parameters. So the user must be experienced how to prescribe these values in order to obtain accurate results. Also the computational grid has to be chosen wisely: As we have seen, the results with our finest grids are not always the most accurate ones!

Furthermore the user has to prescribe some reference values for the correct calculation of drag and lift values. As Fluent works with physical units, a wrong reference value results in wrong values in the drag- and lift-coefficients. It took us a long time to figure out which reference values have to be entered, as some reference values have misleading names. One example: In the reference value "Area (m^2)" we had to enter "0.1" which is the height of the face of the cylinder in the channel (0.1m). This can be explained as follows: If the 2D-geometry is extruded into a 3D-geometry by 1m depth, the circle transforms into a cylinder which faces the flow by a vertical area of (0.1m x 1m)=0.1m^2. This fact is confusing for inexperienced users and is clearly a source of error, although the approach of using physical units might be helpful to more experiences users.

4.3 CFX

CFX is the only software tested here that does not support 2D-modelling. Each sub-package of CFX presents itself as an individual software package that can be used separately from all other modules. Fortunately the interaction between the different software modules works quite well – the generated files of one package can be read into the next one without any problems, and most software components are already prepared to "launch" the "next" module. Unfortunately this strict separation prevents the user from observing the flow while the computation is running like it is possible in Fluent – only the behavior of the monitor values (residuum, self-designed monitor values) can be observed.

All modules offer a fairly well designed and fast GUI which makes working with the program convenient. In CFX-Pre and CFX-Post the user has more options to directly select and affect each component of the geometry that it is possible in Femlab. As the GUI is not designed in JAVA, it is much faster than Femlab – not to speak of Fluent, which has no graphical interface at all for the geometry except for the geometry generator itself. The handling of expressions is a little bit less well designed than in Femlab, but it does a good job.

The results that the solver computes seem to be a bit inaccurate. The fact that the vortex-shedding in the time-dependent simulation did not initiate by itself seems to be a result of a

less accurate discretization like in Fluent, but we have not been able to directly find a way to increase it. The GUI offers so many different parameters for the solver and all the mathematical models and schemes, that the user needs some mathematical understanding how to select the correct ones – also most of the parameters are hidden if they are not used or unimportant. Also the fact that the calculation of the drag- and lift-coefficients was not directly possible and nowhere described – although the user's manual speaks of drag- and lift-coefficients, but these terms are used in another sense – was a very unpleasant detail to us.

5 References

[1] Schäfer, M.; Turek, S.: *Benchmark computations of laminar flow around a cylinder*; in E.H. Hirschel (editor), *Flow Simulation with High-Performance Computers II*, Vol. 52, Notes on Numerical Fluid Mechanics, 547-566, Vieweg 1996.

[2] John, V.: *Higher order finite element methods and multigrid solvers in a benchmark problem for the 3D Navier-Stokes equations*. Int. J. Numer. Meth. Fluids, 40:775–798, 2002. DOI 10.1002/fld.377.

[3] John, V., Matthies, G.: *Higher order finite element discretizations in a benchmark problem for incompressible flows*. Int. J. Numer. Meth. Fluids, 37:885–903, 2001. DOI 10.1002/fld.195.

[4] John, V., Matthies, G.: *Reference values for drag and lift in a two-dimensional time-dependent flow around a cylinder*. Int. J. Numer. Meth. Fluids, 44:777–788 2004. DOI 10.1002/fld.679.

[5] Nabh, G.: *On high order methods for the stationary incompressible Navier-Stokes equations.* PhD thesis, Universität Heidelberg. Preprint 14/98, 1998.

[6] Turek, S.: *Efficient Solvers for Incompressible Flow Problems*, Lecture Notes in Computational Science and Engineering 6, 1999 Springer-Verlag Heidelberg

[7] ANSYS CFX, Ansys Inc., http://www.ansys.com/

[8] FEMLAB Multiphysics Modelling, COMSOL Inc., http://www.femlab.com/

[9] FLUENT Flow Modeling Software, Fluent Inc., http://www.fluent.com/