Benchmarking FEMLAB 3.0a: Laminar Flows in 2D

by

Michael Hanke
Royal Institute of Technology
Department of Numerical Analysis and Computer Science

Parallel and Scientific Computing Institute

Royal Institute of Technology
and
Uppsala University

Report No. 2004:01

April 21, 2004

Benchmarking FEMLAB 3.0a: Laminar Flows in 2D

Michael Hanke
Royal Institute of Technology
Department of Numerical Analysis and Computer Science

April 21, 2004

Abstract

Recently, version 3.0a of the software package FEMLAB for solving multiphysics partial differential equations was released. In the present project we compared its performance with the previous release 2.3 and the package Fluent 6.1 in stationary laminar flow benchmark problems in two and three dimensions. It turns out that the new version is an essential improvement over the old version. It is much faster then Fluent in 2D

Contents

1 Introduction

Partial differential equations form the mathematical foundation for a host of important areas in engineering and physics. FEMLAB provides a powerful interactive environment for modeling and solving scientific and engineering problems which base on partial differential equations. Using FEMLAB one can model strongly nonlinear coupled multiphysics applications with ease. There is no inherent limitation on the simultaneous simulation of many physical phenomena. FEMLAB can handle (systems of) second and first order partial differential equations in one, two and three space dimensions. They are discretized by the finite element method. The (extensible) element library uses mostly polynomial elements on triangles (in 2D) and tetrahedra (in 3D), respectively. Some elements are available which are adapted to be applied special applications.

From the point of view of applicability, it is the multiphysics feature and the extensibility which distinguish FEMLAB. A host of models from different areas of applications are prepared in an easily accessible manner (the so-called applications modes) which can be combined by simple drag-and-drop techniques into complex multiphysics models.

A graphical user interface allows for an efficient graphical design of rather complex geometries in one, two and three dimensions.

The powerful capabilities of FEMLAB give immediately rise to the question whether the user has to pay in order to use such a convenient tool. There are a number of very advanced software packages on the market competing with FEMLAB. Often, they have some emphasis on certain applications areas. This allows for the use of numerical algorithms being more adapted to the application at hand. Compared to that, the algorithms in FEMLAB must be of a more general nature in order to cover the broad spectrum of applications FEMLAB is intended for.

The latest version of FEMLAB is 3.0a. It distinguishes itself from all previous versions in that all computational kernels have been reimplemented in C++. The previous versions are implemented in MATLAB. While opening all the features of MATLAB to be used in FEMLAB it has the drawback of slowing down the numerics and increasing the memory requirements. Even if FEMLAB 3.0a can be run standalone, an interface to MATLAB is available such that

the numerical kernel of FEMLAB can be used as a computing server. We used this possibility extensively in the following benchmark tests.

In the present report, we compare the recent version of FEMLAB against an older one and Fluent. The latter one has its strength in CFD modeling. Therefore, we will use standard benchmark problems for the computation of the laminar flow around a cylinder. We have three aims:

1. Compare different discretizations in FEMLAB 3.0a. Does it pay to use higher order elements? FEMLAB includes the feature of an automatic mesh adaption. Does it pay to use those automatically adapted grids?

2. Compare the performance of FEMLAB 3.0a and the previous MATLAB-based FEMLAB version. We expect shorter computation times and a more memory-economic behavior. Can we quantify it?

3. Compare FEMLAB 3.0a with Fluent. How is the performance of the new version compared to a well-established CFD software tool?

2 Benchmark Methodology

2.1 Benchmark Example

We adopted the benchmark problems proposed in [2, 1] because these problems are well documented and a number of reference values are available which can be used to verify our test results. The fluid properties are identical for all test cases. The problem consists of solving the incompressible stationary Navier-Stokes equation

$$-\nabla\cdot\eta(\nabla\mathbf{u}+(\nabla\mathbf{u})^T)+\rho(\mathbf{u}\cdot\nabla)\mathbf{u}+\nabla p=\mathbf{F},$$
$$\nabla\cdot\mathbf{u}=0.$$

Here, η is the dynamic viscosity, ρ the density, $\mathbf{u}$ the velocity field, p the pressure, and $\mathbf{F}$ a volume force field. The parameters are chosen to be

$$\rho=1\text{kg/m}^3,\quad \eta=10^{-3}\text{m}^2/\text{s},\quad \mathbf{F}=\mathbf{0}.$$

The problem configuration is the flow around a cylinder with circular cross-section in 2D and 3D.

The 2D Configuration

The channel has a length of $L=2.2\text{m}$ and a height of $H=0.41\text{m}$. The cylinder has a diameter of $D=0.1\text{m}$. Its center is located at $(x,y)=(0.2\text{m},0.2\text{m})$ which is slightly unsymmetric. Therefore, a lifting force will appear. At all boundaries we have no-slip boundary conditions

$$\mathbf{u}=\mathbf{0},$$

with the exception of the inlet boundary $x = 0$ where the inflow velocity is prescribed by

$$\mathbf{u}(0,y) = (4U_m y(H-y)/H^2, 0)^T, \quad U_m = 0.3\text{m/s},$$

and the outflow boundary $x = L$ where the boundary condition reads

$$p = 0, \quad \mathbf{K} = \mathbf{n} \cdot \eta(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) = 0.$$

The mean velocity in this case is

$$\bar{U} = \frac{2}{3} U_m$$

such that the Reynolds number becomes

$$Re = \frac{\bar{U} D}{\eta} = 20.$$

The following quantities should be computed: the pressure difference $\Delta p = p(x_a, y_a) - p(x_e, y_e)$ with the front and the end point of the cylinder $(x_a, y_a) = (0.15\text{m}, 0.2\text{m})$ and $(x_e, y_e) = (0.25\text{m}, 0.2\text{m})$, respectively, the drag coefficient c_D, and the lift coefficient c_L. The latter are given by

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

with the drag and lift forces given by

$$F_D = \int_S (-K_x + p * n_x) dS, \quad F_L = \int_S (-K_y + p * n_y) dS.$$

Here, S is the surface of the cylinder, K_x, K_y are the components of the viscous force, and $\mathbf{n} = (n_x, n_y)^T$ is the outer normal.

2.2 Test Criteria

With all of the cited software tools we computed the quantities mentioned above. The aim is to compare the accuracy of this quantities and the amount of computing resources necessary. Taking into account the very different architecture of the packages with respect to programming principles and user interfaces we decided to benchmark only the pure numerical kernels. This includes also the handling of geometry and grid generation. While this is an integral part of the FEMLAB family it is split over different modules in Fluent. Therfore, the tests are carried out as follows:

FEMLAB The test cases were generated in the graphical user environment and saved as an MATLAB m-file. Then the generated m-files were hand-edited in order to remove all unnecessary commands. This includes all graphical output. The protocol output was restricted to the standard value `report = 'on'`. This gives negligible overhead for FEMLAB 2.3. In case of FEMLAB 3.0a, a JAVA subprocess was started whose resources we

neglected. The only post-processing left is the computation of the benchmark quantities pressure drop, drag coefficient, and lift coefficient. Then, MATLAB was started without the JAVA virtual machine thus reducing its resource requirements to a minimum. For FEMLAB 3.0a, the server was started in parallel such that a connection to MATLAB was established.

Fluent The test cases were constructed in the graphical user interface and exported as a Fluent cas-file. The output was reduced to a bare minimum on the console. In particular, the iteration history was only printed every tenth step in order to reduce the overhead additionally. No graphics output was generated. The Fluent jou-file was reduced to the bare minimum such that the programme could be run in a batch queue without any user interaction.

Our aim was to measure both execution time and memory needs.

Memory Especially in the 3D calculations, we expect that the memory requirements will become a restricting factor. Therefore, it would be worth to measure the memory requirements. Unfortunately, these are not available. The programmes do not report such quantities. Neither it is possible to use indirect measurements via tools of the operating system because multi-threading is used. So the only information available is when the programmes started to swap. A problem size was considered computable if the process did not start to swap.

Time FEMLAB integrates geometry definition, mesh generation, initial value generation, numerical solution, and post-processing into one module. Therefore, we considered one test case as consisting of exactly these components. Similarly, the execution of one Fluent jou-file having the same sub-tasks was considered as one test case. Therefore, we did not have immediate access to the cpu time. Instead we decided to measure the wall-clock time. This was accomplished by the MATLAB commands `tic` and `toc` for FEMLAB, and the unix `time` command for Fluent. As far as the latter is concerned, the wall-clock times differed never by more than one percent from the reported cpu time. The longer the run the smaller the difference between these two timings. In order to obtain a realistic timing, the computer was logically decoupled from the network and all unnecessary processes were killed. `top` snapshots of cpu usage indicated that around 99 percent of the cpu were usually dedicated to the processes to be timed. Another problem is concerned with FEMLAB in 3D. The net generator contains a certain randomness. So it is not possible to run the same test case twice. The number of degrees of freedom differs from run to run.

Another strategy was to mimic the average user. Both programmes have a lot of parameters which can be used to tune their behavior. We did not try to tailor these parameters for maximal performance. Instead, the standard parameter settings were always used with the exceptions of those indicated explicitely. There is one exception to this rule: The termination criteria for the nonlinear iteration base on a scaled error estimation in FEMLAB and a scaled residual in Fluent, respectively. In order to obtain criteria on par, the tolerance in Fluent was decreased to the value used in FEMLAB (10^{-6}).

In FEMLAB, there are two possibilities to compute the drag and lift coefficients. One uses the formulae using the viscous forces directly. They are made available by differentiating the finite element solution. A second possibility consists of the use of so-called weak constraints. In this approach, the Dirichlet boundary conditions are enforced indirectly via the use of Lagrange multipliers. Since these Lagrange multipliers are approximated by finite elements independently of the elements used for the discretization of the velocity, they are available with a much larger accuracy. Since the Lagrange multipliers represent the viscous forces, they can be used for the computation of the lift and drag coefficients thus giving rise to a much higher accuracy. The price to pay is that the discrete nonlinear system is much harder to solve, in particular in three dimensions where a useful preconditioner is much harder to provide. Nevertheless, we used the approach via weak boundary conditions because the lift coefficient could not be computed reliably otherwise – even the sign was sometimes wrong.

2.3 Test Environment

Hardware	Pentium 4, 2.4 GHz, 1 MB RAM (Dell Precision 340)
Operating system	SuSE Linux 8.1, kernel 2.4.21-SuSE
LINPACK 1000	gnu f77 3.2: 216 MFlops; ifc 7.0: 223 MFlops
MATLAB 6.5	build 180913a
FEMLAB 2.3	build 145
FEMLAB 3.0a	build 207
Fluent	6.1.22

3 Results in Detail

The results of the benchmark computations are presented in tables. Each run is characterized by a certain parameter which is explained near the respective tables and used as a label in every row. The next rows contain the number of degrees of freedom (unknowns), the computed values of pressure difference (Δp), drag coefficient (c_D), and lift coefficient (c_L), and the computation time (time). The exact analytical values for the benchmark quantities are not known. In [2], intervals are proposed which should contain these true values. The last rows of every table contain these values. The grayed out tabular entries contain values which lie out of this range.

The performance of FEMLAB 3.0a

The following table contains results which are obtained by using the standard grid and refining is uniformly a number of steps (`#ref`).

# ref	unknowns	Δp	c_D	c_L	time
0	2922	0.11925	5.5784	0.01038	6
1	11163	0.11777	5.5793	0.01051	15
2	43602	0.11763	5.5795	0.01061	62
3	172308	0.11755	5.5795	0.01062	344
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

In the next test, we initialized the grid and applied a number of mesh adaption steps (#ngen).

# ngen	unknowns	Δp	c_D	c_L	time
0	2922	0.11925	5.5784	0.01038	6
1	7133	0.11866	5.5788	0.00932	13
2	17482	0.11721	5.5794	0.00990	32
3	39902	0.11750	5.5796	0.01062	83
4	90122	0.11767	5.5795	0.01062	221
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

In this test series, the performance of different order Lagrange elements is compared. For an increasing number of adaption steps (ngen) the following results are obtained.

ngen = 0

element	unknowns	Δp	c_D	c_L	time
P2-P1	2922	0.11925	5.5784	0.01038	6
P3-P2	6913	0.11778	5.5795	0.01056	14
P4-P3	12677	0.11761	5.5794	0.01059	30
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

ngen = 1

element	unknowns	Δp	c_D	c_L	time
P2-P1	7133	0.11866	5.5788	0.00932	13
P3-P2	19844	0.11792	5.5795	0.01059	49
P4-P3	37525	0.11749	5.5794	0.01080	114
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

ngen = 2

element	unknowns	Δp	c_D	c_L	time
P2-P1	17482	0.11721	5.5794	0.00990	32
P3-P2	60068	0.11751	5.5795	0.01062	165
P4-P3	123253	0.11752	5.5795	0.01062	457
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

ngen = 3

element	unknowns	Δp	c_D	c_L	time
P2-P1	39902	0.11750	5.5796	0.01062	83
P3-P2	163870	0.11753	5.5795	0.01062	539
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

For the P3-P2 element, a very small amount of swapping took place (but the CPU usage was almost always 100%). The case of the P4-P3 element could not be used because of lack of memory.

A Comparison of FEMLAB 3.0a, FEMLAB 2.3, And Fluent

The following tables contain the results using a standard refinement strategy:
FEMLAB 3.0a

# ref	unknowns	Δp	c_D	c_L	time
0	2922	0.11925	5.5784	0.01038	6
1	11163	0.11777	5.5793	0.01051	15
2	43602	0.11763	5.5795	0.01061	62
3	172308	0.11755	5.5795	0.01062	344
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

FEMLAB 2.3

# ref	unknowns	Δp	c_D	c_L	time
0	2753	0.11781	5.5775	0.00618	14
1	10465	0.11754	5.5807	0.01034	31
2	40766	0.11754	5.5800	0.01059	179
3	ca. 160000	Out	of	memory	
min		0.1172	5.57	0.0104	
max		0.1176	5.59	0.0110	

Fluent

# ref	unknowns	iters	Δp	c_D	c_L	time
1	14394	890	0.1090	5.5289	0.02573	72
2	31080	1280	0.1128	5.5633	0.01016	206
3	59994	2170	0.1145	5.5755	0.00725	660
min			0.1172	5.57	0.0104	
max			0.1176	5.59	0.0110	

4 Conclusions

1. Mesh convergence in FEMLAB 3.0a is clearly seen. The value of Δp and c_D are relatively easy to approximate. The value for c_L which is considered to be hardest computable was surprisingly well approximated.

2. In the present example, adaptive mesh refinement does not provide any advantages. The accuracy between standard refinement and grid adaption is comparable while the computation time increases.

3. It is not quite clear if it pays to use higher order elements. Comparing computation time and accuracy, P2-P1 elements seem to be slightly preferable.

4. Since the basic numerical algorithms in FEMLAB 3.0a and FEMLAB 2.3 are identical, it is not surprising that the accuracy of both versions is identical for the same number of degrees of freedom. The new version is much more memory efficient than the older one. The new version is by a factor of 2–3 faster than the old one.

5. Also in Fluent, mesh convergence can be observed. The number of degrees of freedom was too small for obtaining accurate results. This observation was also made in 3D: Compared to FEMLAB, Fluent needs many more degrees of freedom for a comparable accuracy. Even for the same number of unknowns, Fluent is by a factor of 8–10 slower than FEMLAB 3.0a.

6. We ran even the time-dependent benchmark problem 2D-3 of [2, 1]. The results obtained with FEMLAB 3.0a indicated an unphysical behavior. We did not even obtain results with Fluent. Because of this suspicious behavior, we excluded the results from the benchmark tests.

5 Summary

FEMLAB 3.0a is fast and reliable in computing 2D laminar flows. It is much faster than Fluent 6.1 in these computations.

The implementation of weak boundary conditions gives rise to an accurate computation of drag and lift coefficients even with a low number of degrees of freedom.

FEMLAB 3.0a is an essential improvement over the previous version 2.3 with respect to speed and memory requirements.

Acknowledgements Our sincere gratitudes go to Kalle Pettersson who prepared the cas-files for all Fluent runs.

References

[1] M. Schäfer, R. Rannacher, and S. Turek. Evaluation of a CFD benchmark for laminar flows. In Hans Georg et al. Bock, editor, *ENUMATH 97. Proceedings of the 2nd Europeen conference on numerical mathematics and advanced applications*, pages 549–563, Singapore, 1997. World Scientific.

[2] M Schäfer and S Turek. Benchmark computations of laminar flow around a cylinder. In E.H. Hirschel, editor, *Flow Simulation With High-Performance Computers II*, volume 52 of *Notes on Numerical Fluid Dynamics*, pages 547–566. Viehweg, 1996.