CFD for Incompressible Flow: Numerical Efficiency vs. Gigaflops

S. Turek, Institute for Applied Mathematics & Numerics, University of Dortmund

One of the major tasks of nowaday Applied Mathematics and Numerics is the development of efficient methods for the numerical solution of Partial Differential Equations (PDE's). However, their simulation on modern computers with continuously increasing computing power is currently performed by so many different groups from Physics, Chemistry, engineerical disciplines, Computer Science and more, such that the actual role of Mathematics is not quite clear. The aim of this article is to demonstrate that particularly modern Numerics has a large potential to achieve very significant improvements for the quality of computer simulations, especially for CPU-intensive 'real life' problems. In contrast, we also illustrate by examples that it is by far not sufficient to wait for higher GFLOP/s rates only!

For the more or less special class of incompressible flow, we discuss the fundamental problems of *Computational Fluid Dynamics* (CFD) and we propose 'new' mathematical and algorithmic concepts which are currently under development. Realizing them as software, the practitioner will obtain highly improved tools - with respect to efficiency and accuracy - which may really allow in future the qualitatively and quantitatively precise prediction of realistic flow behaviour.

1 Motivation

The system of the incompressible Navier–Stokes equations describes – in a domain $\Omega \subset \mathbf{R}^3$ and for a time intervall $(t_0, T]$ – the velocity $\mathbf{U}(x_1, x_2, x_3, t)$, with components (U_1, U_2, U_3), and the pressure $P(x_1, x_2, x_3, t)$ of an incompressible fluid for given physical properties (kinematic viscosity ν, density ρ) and prescribed initial and boundary values:

$$\rho \frac{\partial U_i}{\partial t} + \rho \frac{\partial}{\partial x_j}\left(U_j U_i\right) = \rho\nu \frac{\partial}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{\partial P}{\partial x_i} \qquad , \qquad \frac{\partial U_i}{\partial x_i} = 0$$

These equations seem to have a quite simple structure, nevertheless they provide 'grand challenge' problems for mathematicians and physicists as well as engineers, and they are (among others) object of very intensive research activities. However, beside people coming from research areas also software developers and providers of commercial codes work hard in this field, and all together meet at one common point, the *Computational Fluid Dynamics* (CFD) software.

Particularly for mathematicians, incompressible flow problems provide a great potential for research activities since they include a wide variety of difficulties which typically arise in the numerical treatment of PDE's. Thus, they are the perfect playground to test the efficiency of today's Numerics. Going into detail, the problems which scientists and practitioners are confronted with concern the following mathematical aspects:

- time dependent partial differential equations in complex domains
- strongly nonlinear systems of equations
- saddle–point problems due to the incompressibility constraint
- local changes of the problem character in space and time
- temporarily stiff systems of differential equations

These characteristics impose great challenges on almost all fields of numerical and computational approaches. Among others, the following exemplary issues have to be taken into consideration if efficient CFD tools shall be designed:

1. *very large nonlinear systems of equations* (**millions of unknowns**)
2. *locally varying time steps* (**implicit schemes**)
3. *locally anisotropic spatial meshes* (**boundary layers, complex geometries**)
4. *efficient solvers (adapted to* **workstations** *and/or* **parallel supercomputers**)

Active research in numerical and computational methods is going on since more than 30 years and the number of publications and software packages is enormous (see [1] for an impressive overview): As the following example shows, CFD software leads in many cases to qualitatively precise flow prediction. However, it is also fact that **no** recently existing CFD software contains rigorous mechanisms which are able to control the quantitative precision of the numerical solution with respect to the (unknown) exact solution. A requirement which is absolutely necessary if for instance drag or lift values have to be specified!

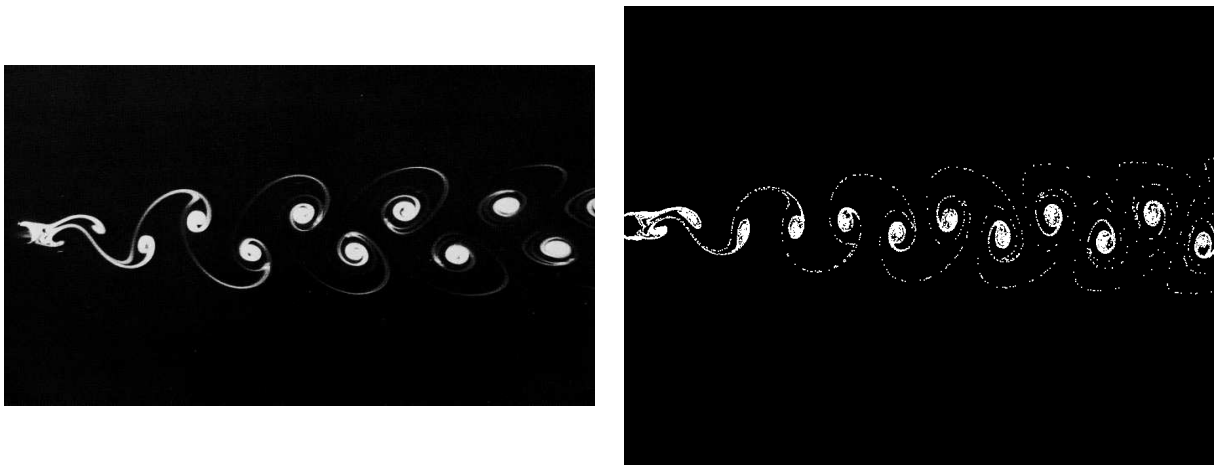

Figure 1: Comparison of experiment (from: Van Dyke's 'Album of Fluid Motion' [12]) and numerical simulation (from: 'Virtual Album of Fluid Motion' [11]): *Flow around cylinder for medium Reynolds number*. Both pictures show a qualitatively 'good' agreement (unfortunately, the Reynolds numbers and the length scales are not identical!), but there is no chance to get estimates - based on the graphics only - for the quality of quantities as drag and lift coefficients directly on the cylinder.

One might expect (and it is often true!) that at least laminar incompressible problems - in 2D, but also in 3D? - could be solved if we only exploit the very large computing power today. However, this is not sufficient! Current trends in CFD are to combine these basic equations with even more complex components to simulate 'harder' applications. Only to call some of them: models from Physics and Chemistry are added for simulating turbulence, multiphase flow, nonlinear fluids, combustion/detonation, free boundaries, weakly compressible effects, etc. Additionally, the mathematical community begins to incorporate error control mechanisms of a posteriori type which require the handling of fully adaptive concepts in space and time.

These physical and mathematical extensions have one common aspect: They all require very efficient and robust solution schemes for generalized Navier–Stokes–like systems as part of the complete system. Someone not being a CFD insider might expect that the numerical solution of such basic (laminar) incompressible Navier–Stokes equations should be completely under control, despite all the well-known difficulties described above. Is it true? Therefore, before the embedding into more complex frameworks is intensified, one should examine the question:

> *'Are the existing solution algorithms for (laminar) incompressible flow problems already sufficient or is further, maybe even tremendous improvement necessary?'*

Analyzing current software tools for the numerical simulation of the incompressible Navier-Stokes equations, one may get the following impression:

- Most software packages promise to solve *all* problems!
- Pre- and postprocessing and supported computer systems are *state-of-the-art*!

However, a more careful look shows that the employed mathematical and algorithmic methods often are far away from being *state-of-the-art*. In fact, the *numerical kernels* of many (commercial) simulation tools have been developed in the 70's or 80's, more or less neglecting many numerical improvements of the last decade! And since in many cases the main programmers have left the company, it is hard to change these innermost computational kernels...
Based on the following *DFG–Benchmark* which has been performed by many groups during the last years we could examine these statements more carefully. Since the participating groups and codes represent a large part of most recent research activities in this field, the following results provide still today a quite good overview concerning the state-of-the-art in Computational Fluid Dynamics for the laminar incompressible Navier-Stokes equations.

2 The 'DFG–Benchmark' configurations

Under the DFG Priority Research Programme 'Flow Simulation on High Performance Computers', (see [4]) several test cases for incompressible flow problems have been defined which have been performed by many groups up to now, with more or less considerable success. Several new techniques such as *unstructured grids*, *multigrid*, *operator splitting*, *domain decomposition* and *mesh adaptation* which are typically used in order to 'improve the performance' could be compared and evaluated by these benchmark problems. In the following, we restrict us to the cases of *Flow around a 3D Cylinder*, in particular to *stationary 3D flow, Reynolds number* $Re = 20$

and *nonstationary 3D flow with time-dependent inflow, (maximum) Reynolds number* $Re = 100$; the complete description of all configurations and results can be found in [4], [5] and [6].

The quantities to be calculated are - among others - the drag coefficient c_d, the lift coefficient c_l and the pressure difference ΔP on the cylinder, as steady values as well as time-dependent curves. In particular, controlling the drag and lift coefficients has proven to be very hard. This fact is very interesting for industrial applications since both represent realistic quantities from 'real life' applications which however seem to be almost impossible to predict accurately.

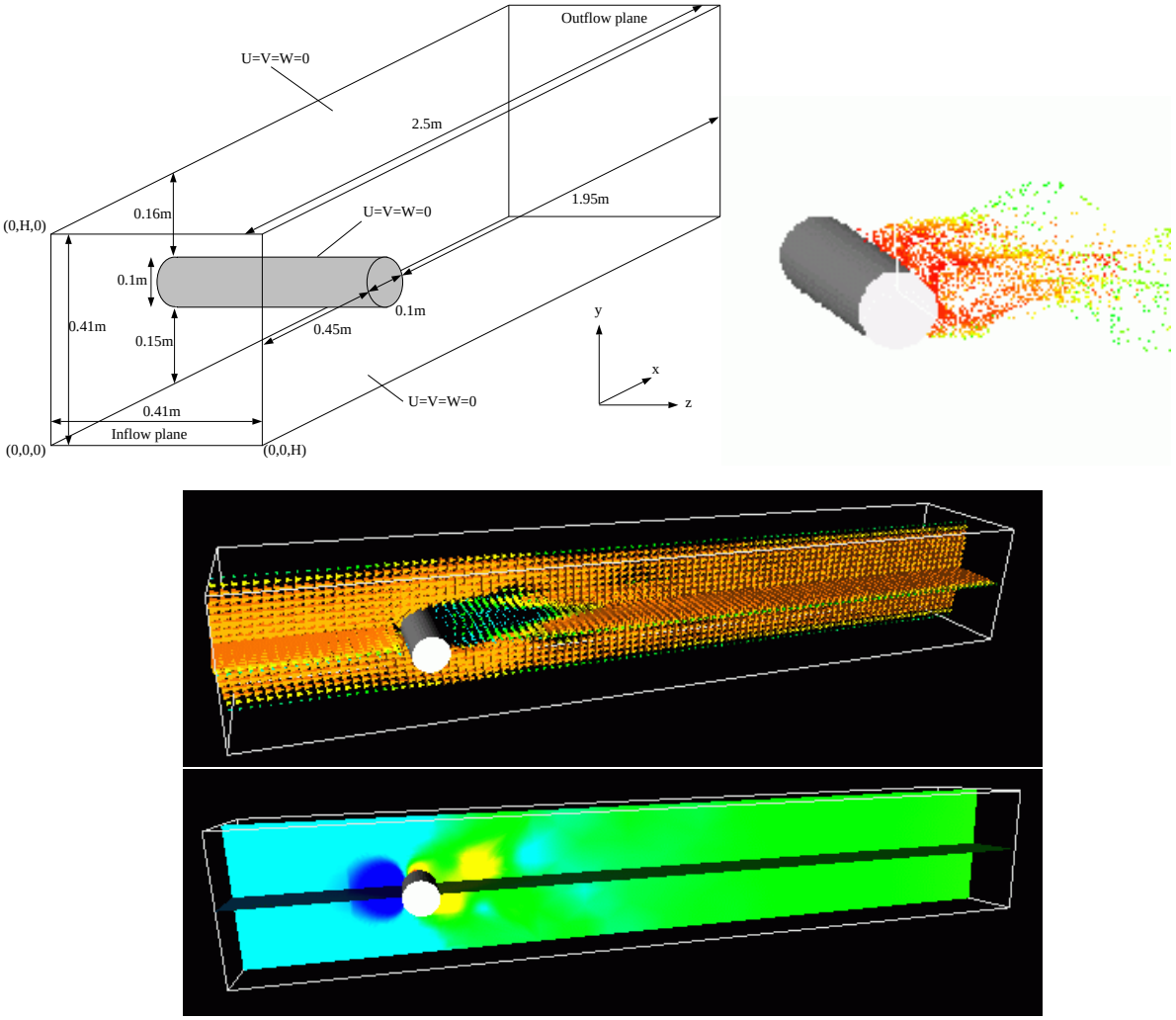

Figure 2: Geometry with boundary conditions and 'snapshots' of the nonsteady solution: We use a particle advector tool [11] which is based on the calculated flow field and show velocity and pressure on a cutplane.

Some of the 'official' results (that means such results which have been performed during the official benchmarking period) are listed in Tables 1 and 2 (we are group 10) including the used computers ('#NEQ' stands for 'spatial unknowns', '#TIME' for 'number of time steps', 'RAM' for 'needed RAM memory in MByte', 'CPU(s)' for 'elapsed CPU time in seconds', 'S' for 'supercomputer', 'W' for workstation/PC', 'REF' for 'proposed reference values').

Gr.	#NEQ	c_d	c_l	ΔP	RAM	CPU(s)	Computer
1	2426292	6.1295	0.0093	0.1693	233	**2097**	Fuj. VPP500 (S)
2	555000	6.1440	0.0074	0.1604	122	**8731**	IBM 6K/370 (W)
3	608496	6.1600	0.0095	0.1690	74	**4150**	Cray T3D/16 (S)
6	6303750	6.2330	-0.0040	—	43	**221706**	HP735 (W)
7	12582912	6.1932	0.0093	0.1709	3571	**2630**	GC/PP128 (S)
8	362613	6.1430	0.0084	0.1694	126	**51280**	IBM 6K/590 (W)
9	2355712	6.1800	-0.0010	0.1691		**62000**	IBM 6K/590 (W)
10	6116608	6.1043	0.0079	0.1672	700	**8440**	IBM 6K/590 (W)
REF		$\sim$ 6.15	$\sim$ 0.008	$\sim$ 0.17			

Table 1: Results for the stationary 3D test (details in [4] and [6])

Gr.	#NEQ	#TIME	c_d	c_l	ΔP	RAM	CPU(s)	Computer
1	630564	800	3.2826	0.0027	-0.1117	79	**156460**	Fuj. VPP500 (S)
3	608496	1600	3.2590	0.0026	-0.1072	74	**76142**	Cray T3D/16 (S)
6	6303750	18000	4.1600	0.0200	—	43	**142646**	HP735 (W)
7	1572864	1600	3.3011	0.0026	-0.1102	518	**149923**	GC/PP128 (S)
8	199802	1000	3.2120	0.0122	-0.1112	105	**846000**	IBM 6K/590 (W)
10	6116608	668	3.2802	0.0034	-0.0959	840	**164837**	IBM 6K/590 (W)
REF			$\sim$ 3.29	$\sim$ 0.003	$\sim$ -0.1			

Table 2: Results for the nonstationary 3D test (details in [4] and [6])

Several conclusions have been drawn on the basis of these benchmark results which were accepted by all participating groups (see [4] and [6] for the complete discussion):

- While a **qualitatively** correct prediction of flow behaviour is possible by most existing codes, accurate simulations with respect to **quantitatively** precise results are very hard to perform. While in 2D the required accuracy can often be achieved by exploiting massive computing power, those methods get into trouble even on supercomputers if they are based on low-order discretization schemes or solvers with non-optimal run-time behaviour. Surprisingly, for some of the laminar test cases in 3D in the complete benchmark suite [4], the corresponding reference values could not be obtained, even using supercomputers: The maximum possible mesh size had to be chosen due to CPU limits and less due to storage requirements!

- Schemes which do not use **implicit** time-stepping and modern **multigrid-based** solvers are much too inefficient: Codes on existing workstations together with modern numerical techniques can be faster than 'old-fashioned' software on supercomputers.

- **Nonsteady** flow solvers still need significant improvements. Looking carefully at the actually elapsed CPU times, even on supercomputers, it is obvious why the test cases have been restricted to the moderate cases of Reynolds numbers up to $Re = 100$ only.

These conclusions seem to be valid not only for this special benchmark, but also for more complex flow problems. As a conclusion, they give a quite good impression of the capacity of recent CFD tools, including also commercial packages: *Computational Fluid Dynamics gets into trouble if quantitatively precise flow prediction is required!*

3 Numerical and algorithmic concepts

One of the aims of modern Numerics is the realization of the vision 'safe solution on demand' which is under progress during the last years:

The incompressible Navier–Stokes equations are solved numerically on a computer in such a way that a specific flow quantity which is prescribed by the user is guaranteed up to a certain error tolerance in comparison to the (unknown) exact solution. The user has only to define the application framework and the error tolerance (and hence a time limit). Then, all discretization and solution processes are handled fully automatically by the code itself.

To realize such a 'mathematical vision' as applicable software, several aspects have to be considered, resp., all the following components have to be successfully combined:

- Mathematical schemes which are rigorously analyzable ($\rightarrow$ *discretization*)
- Efficient algorithms ($\rightarrow$ *solver*)
- Verified and hardware-optimized software ($\rightarrow$ *implementation*)

If we could derive such optimized components, one should obtain (at least) the same solution quality with much **less unknowns**, using much **more robust and faster discrete solvers**, producing much **higher MFLOP/s rates**. These orders of magnitudes of efficiency enhancement are the potential of optimal numerical approaches!

At the moment, one of the most promising approaches (see [2] and [6]) is based on Galerkin formulations and Finite Element (FEM) discretizations: They seem to provide a complete framework which allows rigorous a posteriori error control and corresponding adaptive grid manipulations. While classical approaches are better called *error indicators* since a sharp quantitative estimation of the error can be given only in some specific configurations, this new approach seems to overcome these failures and can be formulated in a very general framework such that also the Navier–Stokes equations and other CFD relevant models can be treated. For a motivating presentation of the recent state-of-the-art we propose the papers by Rannacher and his group (see for instance [3]). A frequent point of criticism is that the so-called *dual solutions* including all *stability* and *interpolation constants* due to the specific user-defined control functionals as, e.g., the lift and drag coefficients or the pressure distribution on contours have to be calculated, too. Consequently, one single computation which is based on such control mechanisms may spend more CPU time than classical approaches; but at the moment we do not see any alternative approach which provides rigorous strategies for local refinement or coarsening of the mesh to end up with an (almost) optimal computational mesh to guarantee a certain accuracy!

An important aspect for such self-adaptive error control mechanisms is the efficiency enhancement of the corresponding algorithmic approaches. Since no more a priori information is available on the mesh topology including locally refined spatial grids and varying time steps, solvers with certain *Black Box* character are required to treat iteratively the resulting huge (nonlinear) systems for arbitrary configurations. The aim must be to develop optimal multigrid components for the complete incompressible Navier–Stokes equations; such methods seem to be - at least at the moment - the only available schemes with uniformly bounded efficiency, independent of the problem configuration.

In contrast to classical iterative solvers, they do not work on a single computational mesh only, but they exploit auxiliary solutions on a sequence of hierarchical meshes. Therefore it is obvious, that the development of corresponding *intergrid operators* is strongly coupled with the underlying FEM space. Based on the general class of *Multilevel Pressure Schur Complement* methods (MPSC) which are a generalization of many existing Navier–Stokes solvers, very efficient solvers in combination with special Finite Element spaces can be constructed. A very detailed description of such methods and the corresponding FEM components is part of our book [6] which contains lots of information about recent advances with regard to the numerical solution of discretized incompressible Navier–Stokes problems.

As final aspect we still have to address the *quality of the implementation*, particularly on modern computer platforms. It is true that the speed of computers still increases in a continuous way such that PC's are currently able to perform almost 1 GFLOP/s peak performance (these are 1 billion arithmetic operations per second!). Additionally, extrapolations of the recent technology promise further efficiency enhancement such that workstations/PC's in the year 2010 will perform with 1 TFLOP/s (10^{12} arithmetic operations per second) which is about the same as the fastest parallel supercomputers today, but achieving the same peak performance as single processor! However, one often neglects that these 'peak values' are achievable only if the internal cache memory and piplining units are optimally exploited! If in contrast the implementation has not been carried through in a hardware-optimized manner, the computational efficiency dramatically decreases such that many codes do not obtain a significant percentage of the currently possible 100 MFLOP/s up to 1 GFLOP/s. Much more realistic are efficieny rates of about 1 - 10 MFLOP/s on each processor for the fastest applications (as matrix-vector multiplications, for instance) while complex CFD applications often have problems to achieve 1 MFLOP/s at all. In such a case, the application on parallel computers does not lead to the necessary performance improvements since even the use of 100 processors will lead to a realistic total computational efficiency which is lying still under the possible (peak) performance of one single processor.

This aspect and its consequences for software design and the applied mathematical algorithms is quite new and its study in publications is still very rare. Mathematical basic research of this type, which is strongly coupled with knowledge on computer architecture and software engineering, is a very young discipline and is part, for instance, of so-called 'hardware-oriented Numerics'. For an overview on current problems and recent strategies we refer to the papers [7], [8] and [9]. They contain a description of the underlying concepts for our FEAST software project and the derived FEAST INDICES and SPARSE BANDED BLAS for evaluating and exploiting the high-performance facilities of modern processors for FEM multigrid approaches.

4 Outlook

We are still at the beginning of understanding sufficiently well the proposed mathematical approaches such that rigorous self-adaptive error control can be optimally coupled with corresponding highly efficient solvers. Then, the next laborious step will be to develop professional software for the grand-challenge industrial problems. However, the benchmarks and numerical studies show that we must spend such efforts in improving the 'basic tools', before stepping to more complex simulations! Otherwise, I see no chance to tackle successfully much more complex simulation problems, providing not only qualitatively, but also quantitatively accurate results.

Some preliminary examples for such highly optimized 'basic tools' based on numerical methods and software for laminar incompressible fluids can be found at the FEATFLOW Homepage which also contains the sources for the high-performance FEAST project.

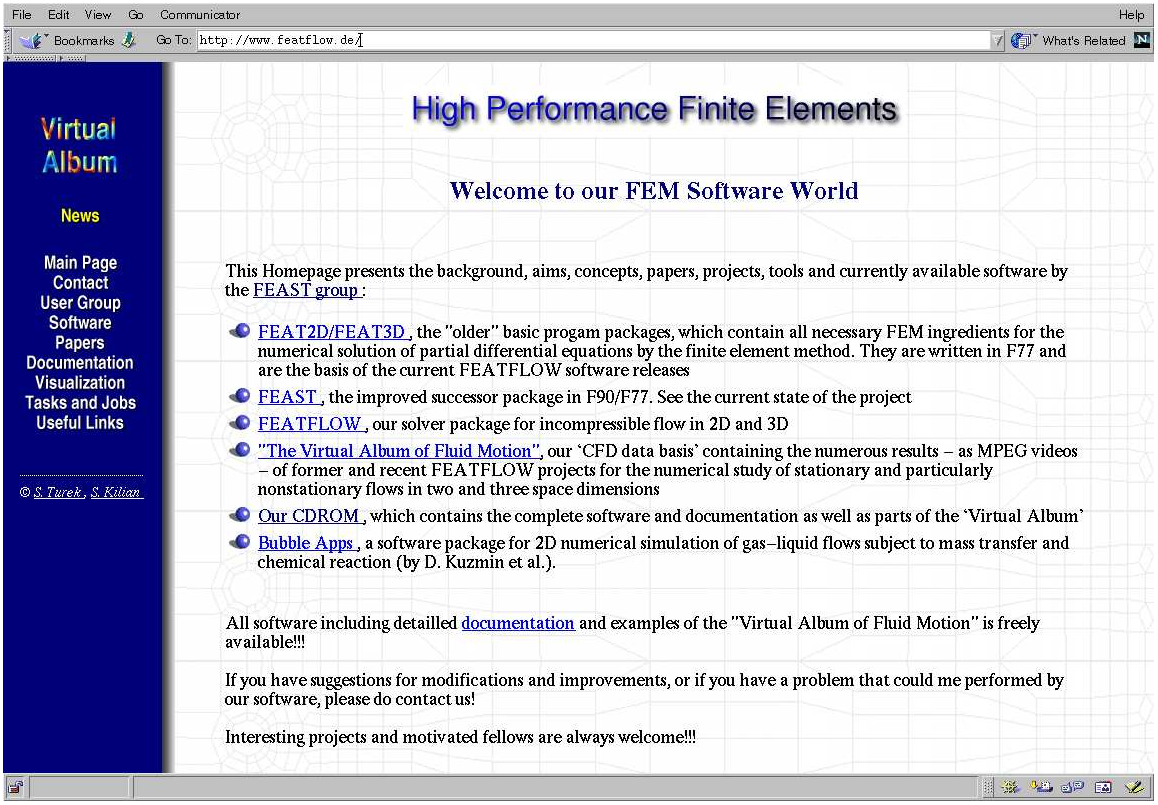

Figure 3: FEATFLOW Homepage (`http://www.featflow.de`)

For those who are interested in possible applications of FEATFLOW, we recommend to look at the 'Virtual Album of Fluid Motion' [11]. This is a unique collection of numerous movies (MPEG) which can be viewed on every workstation/PC. Together with the freely available FEATFLOW sources, they are the perfect starting point to perform your own CFD simulations.

References

[1] Gresho, P.M., Sani, R.L.: *Incompressible Flow and the Finite Element Method*, Wiley, Chichester, 1998

[2] Johnson, C., Rannacher, R., Boman, M.: *Numerics and hydrodynamic stability: Towards error contol in CFD*, SIAM J. Numer. Anal., 32 (1995)

[3] Rannacher, R., Becker, R.: *Weighted a posteriori error control in FE methods*, Proc. Enumath–95, Paris, 18–22 Sept., 1995

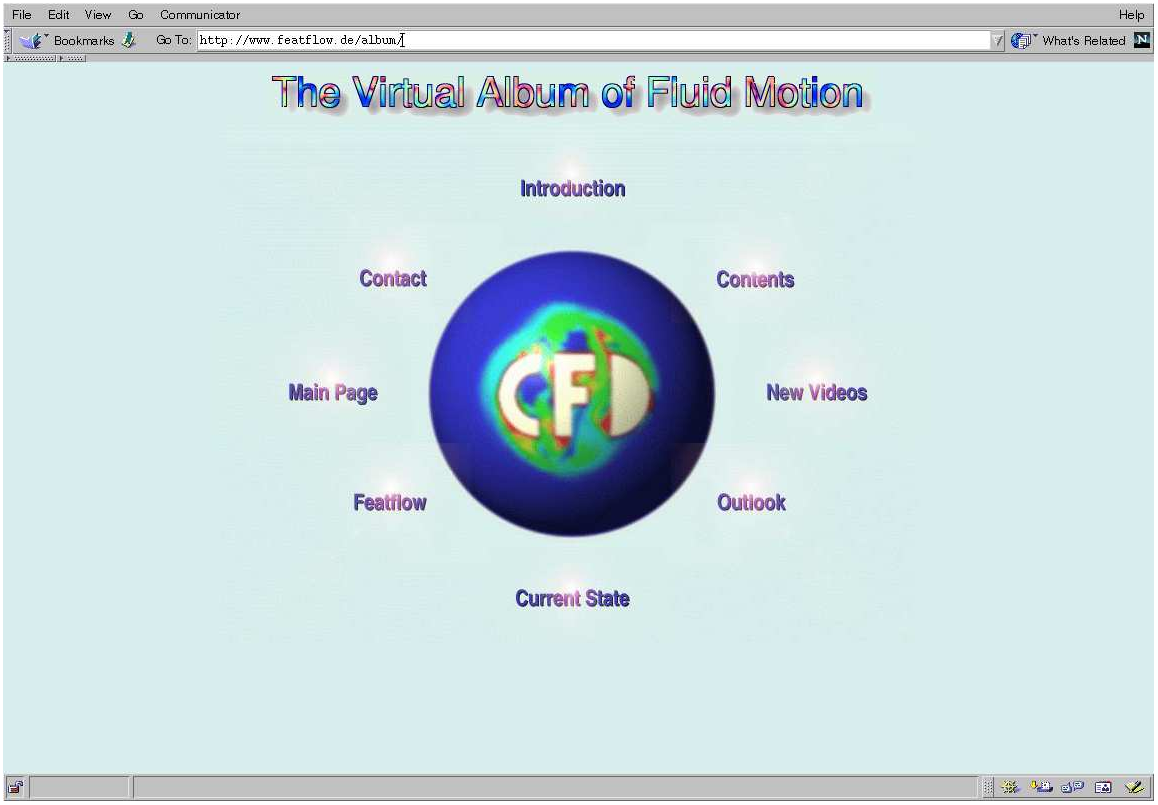

Figure 4: Homepage of the 'Virtual Album of Fluid Motion' (`http://www.featflow.de/album`)

[4] Schäfer, M., Turek, S. (with support by F. Durst, E. Krause, R. Rannacher): *Benchmark computations of laminar flow around cylinder*, in E.H. Hirschel (editor), *Flow Simulation with High-Performance Computers II*, Volume 52 of *Notes on Numerical Fluid Mechanics*, 547–566, Vieweg, 1996

[5] Schäfer, M., Rannacher, R., Turek, S.: *Evaluation of a CFD Benchmark for Laminar Flows*, Proc. ENUMATH-97, Heidelberg, October 1997, World Science Publ., 1998

[6] Turek, S.: *Efficient solvers for incompressible flow problems: An algorithmic and computational approach*, LNCSE 6, Springer-Verlag, 1999

[7] Turek, S.: *Trends in processor technology and their impact on Numerics for PDE's*, Preprint 99–31, U Heidelberg, SFB 359, 1999, submitted to *Numerische Mathematik*

[8] Turek, S. et al.: *Performance rating via the* FEAST INDICES, Computing, 63 (1999)

[9] Turek, S. et al.: *Proposal for* SPARSE BANDED BLAS *techniques*, Preprint 99–11, U Heidelberg, SFB 359, 1999, submitted to *Int. J. of High Performance Computing*

[10] Turek, S.: **FEATFLOW** . *Finite element software for the incompressible Navier–Stokes equations: User Manual, Release 1.1*, 1998

[11] Turek, S.: *The Virtual Album of Fluid Motion*, `http://www.featflow.de/album`

[12] Van Dyke, M.: *An Album of Fluid Motion*, The Parabolic Press, Stanford, California, 1982