Contents

The Fictitious Boundary Method for the implicit treatment of Dirichlet boundary conditions with applications to incompressible flow simulations

Stefan Turek, Decheng Wan, and Liudmila S. Rivkind

Institute of Applied Mathematics LS III, University of Dortmund,
Vogelpothsweg 87, 44227 Dortmund, Germany

Summary. A 'fictitious boundary method' for computing incompressible flows with complicated small-scale and/or time-dependent geometric details is presented. The underlying technique is based on a special treatment of Dirichlet boundary conditions, particularly for FEM discretizations, together with so-called 'iterative filtering techniques' in the context of hierarchical multigrid approaches such that the flow can be efficienctly computed on a fixed computational mesh while the solid boundaries are allowed to move freely through the given mesh. The presented method provides an easy way of incorporating geometrically complicated objects and time-dependent boundary components into standard CFD codes to simulate (at least) the qualitative flow behaviour of complex configurations. Furthermore, higher accuracy can be reached via local mesh adaptation techniques which might be based on local (coarse) mesh adaptation or mesh deformation techniques to avoid expensive (global) grid reconstruction.

We explain the mathematical and algorithmic details and provide numerical examples based on the FeatFlow [13] software for incompressible flow to illustrate qualitatively and to examine quantitatively the presented fictitious boundary method, for various stationary and time-dependent configurations. In particular, we compare with standard approaches which use geometrically adapted meshes, and with a 'viscosity-density blockage' method which describes internal objects via appropriate settings of the density and viscosity parameters in the Navier-Stokes equations. Moreover, we also discuss implementation details and software techniques which can provide very high MFLOP/s rates for such techniques in combination with special hierarchical data, matrix and solver structures.

1 Introduction

Flow configurations with numerous complex geometrical details and/or moving interfaces and boundaries have important applications in a variety of physical and engineering areas such as flows around objects, fluid-structure

interaction, multiphase flows with chemical reactions, stratified flows, bubble dynamics, melting and solidification, crystal growth, etc. The numerical investigation of these physical problems has to take into account the effect of the complicated stationary and/or moving boundaries, and it can require a huge amount of time for the regeneration or deformation of the computational grid when the corresponding interfaces or boundaries are changing. Developing an accurate, effective and robust approach for tackling these problems is necessary: The overall aim is to deal successfully with the moving interfaces or boundaries such that the accuracy of the numerical approximation is increased while at the same time also the computational cost is decreased.

Generally speaking, there are (at least) two major approaches to simulate fluid flows with complex stationary and/or moving boundaries. One of them is a 'body-conformal approach' which always keeps the computational mesh in accordance to the geometrical details. Another one is a 'fixed grid approach' in which case the mesh is (arbitrarily) fixed and internal objects are allowed to move freely through the mesh. One big advantage of such 'fixed grid' approaches over the conventional 'body-conformal' approach is that the computational mesh remains unchanged such that the CPU cost can be significantly decreased - less computational effort due to saving the expensive mesh generation - and that such techniques can be easily incorporated into standard CFD codes which mostly allow fixed computational grids without local adaptivity only.

In this paper, we describe a 'fictitious boundary method' which works on an arbitrarily fixed grid. To be more specific, we concentrate on the problem of how to manage time-dependent domains with complicated geometrical structures inside of 'standard' CFD software. To be even more precise, we additionally assume that (geometrical) multigrid solvers are applied such that appropriate techniques for a sequence of meshes have to be designed. If 'classical' multigrid solvers are applied, one central problem is the sufficiently accurate approximation of complex geometries and their associated boundary curves or surfaces in a hierarchical way. While there is "no" problem to design (very) fine meshes which provide all necessary information, particularly with professional mesh generation tools, the construction of sequences of "coarser" meshes may be more crucial; especially if one keeps in mind that sequences of more or less nested meshes are necessary for multigrid.

One possible approach is to start with a coarse mesh which contains already most of geometrical fine-scale details. This technique may work efficiently in 2D cases, but for analogous 3D applications the resulting "coarsest" mesh will be in general very large (in the range of more than 10,000 - 50,000 mesh cells) such that the typical multigrid efficiency is lost due to a dominating coarse grid solver. In contrast, completely different - since non–nested - grids in comparison to the finest mesh may be used, but the corresponding intergrid transfer routines, which interpolate from one mesh to another, are difficult to handle. Consequently, the resulting multigrid solver spends most of its time with grid transfer routines on lower levels. Further, the convergence

rates may deteriorate since they massively depend on the choice of subgrids and corresponding transfer operators as it is well-known from algebraic multigrid solvers.

Taking these problems into account, we propose another approach which is embedded into a general framework for implementing boundary conditions in iterative solution techniques: the *iterative filtering technique* in combination with *fictitious boundary conditions*.

'Employ a (rough) boundary parametrization which sufficiently describes all large-scale structures with regard to the boundary conditions. Treat all fine-scale features as interior objects such that the corresponding components in all matrices and vectors are unknown degrees of freedom which are implicitely incorporated into all iterative solution steps. Hence, standard tools for grid refinement in interior regions are easily applicable and highly accurate approximations can be obtained. Further, utilize filtering techniques to project the corresponding vector components onto the subspace of "correct" boundary conditions, before and directly after each iterative substep. This additionally ensures the typical performance of standard multigrid solvers without requiring additional modifications in the software components.'

Another important aspect is the corresponding modification of standard (geometrical) multigrid components and the resulting multigrid convergence, in particular if we apply the *filtering process* depending on the mesh level. So, we might apply mesh-dependent filters in such a way that the number of 'internal' objects is decreased when the mesh is coarsened, or the shape may become coarser due to less grid points, or it might even happen that the coarsest configurations for flow problems around internal objects do not contain any object: On the coarse levels, we even could perform a pure channel simulation while the internal objects are present only on the finer meshes! Such a technique - assuming that the typical multigrid convergence behaviour is preserved - of modifying the geometrical details on coarser meshes could be a very powerful tool in the context of multilevel approaches.

It is quite understandable that such an approach might work well even for the nonlinear Navier-Stokes equations, since the essential background idea for our applied nonlinear solution schemes of quasi-Newton type is the strong separation between **nonlinear treatment** and **linear solution tools**. Our approach approximates nonlinear effects on the finest mesh only, while multigrid is exclusively applied to solve linear subproblems in a preconditioning step. Consequently, we have to take care that at least the finest mesh is sufficiently accurate to describe all important physical effects. However, the coarser meshes are – roughly spoken – only involved to "provide some spectral approximation for the acceleration of matrix problems" for the performed linear multigrid. In this context, multigrid is only applied as very efficient linear **preconditioning tool**, separated from the discretization aspects regarding the nonlinearity. This is a very essential difference to *direct nonlinear multigrid* approaches which require the approximation of analogous nonlinear problems

on the coarser meshes, too. Then, due to the separation of both tasks, we can apply standard multigrid tools without any modification and we can guarantee at least the fast treatment of the linear subproblems in each nonlinear iteration step.

This approach may not be the best for some other problems in which case nonlinear multigrid approaches can lead to better results. However, our experience in the case of the incompressible Navier–Stokes equations shows explicitely that this "separation approach" seems to be superior. One reason is that typically the nonlinear convergence behaviour is only determined by the strength of the nonlinearity on the continuous level, that means the Reynolds number, while mesh dependent effects are of (almost) no importance. Hence, quasi-Newton schemes might be superior in comparison to nonlinear multigrid approaches for such problem configurations.

Before we describe and analyse the corresponding numerical details in the next section, we provide an example for applying this technique in the case of a prototypical simulation of moving flaps in a channel. Figure 1 shows the norm of the velocity as well as corresponding 'Particle Tracing' snapshots for simulating the incompressible Navier-Stokes equations in a channel configuration while the movement of the 'flaps' is analytically prescribed. All simulations are computed on a fixed mesh such that the moving flaps are described via the fictitious boundary conditions only and their position is incorporated via the iterative filtering technique (from [12]).

2 Concepts for fictitious boundary conditions

2.1 The incompressible Navier-Stokes resp. Boussinesq equations

All following concepts for integrating boundary conditions into CFD software for incompressible flow problems are realized in the FeatFlow software [13] which is based on (nonconforming) FEM discretizations, adaptive implicit time-stepping, nonlinear Newton-like methods, (geometrical) multigrid solvers (for velocity, temperature and pressure separately as well as for all physical quantities simultanously) on quite arbitrary domains. There exist special solvers in the framework of the so-called 'pressure Schur complement' methodology (see [8]) which can be directly applied as stationary approaches ('CCnD') as well as in fully nonstationary configurations ('PPnD') following operator splitting ideas. The software is realized for $n = 2$ as well as $n = 3$ dimensions; however, in this paper we mainly restrict to the 2D variants which already contain all important numerical features (Remark: The corresponding 3D realizations in FeatFlow do exist, too [13]).

For the following considerations, let Ω be a bounded domain with a piecewise smooth boundary Γ. The equations to be solved are the incompressible Navier-Stokes, resp., Boussinesq equations

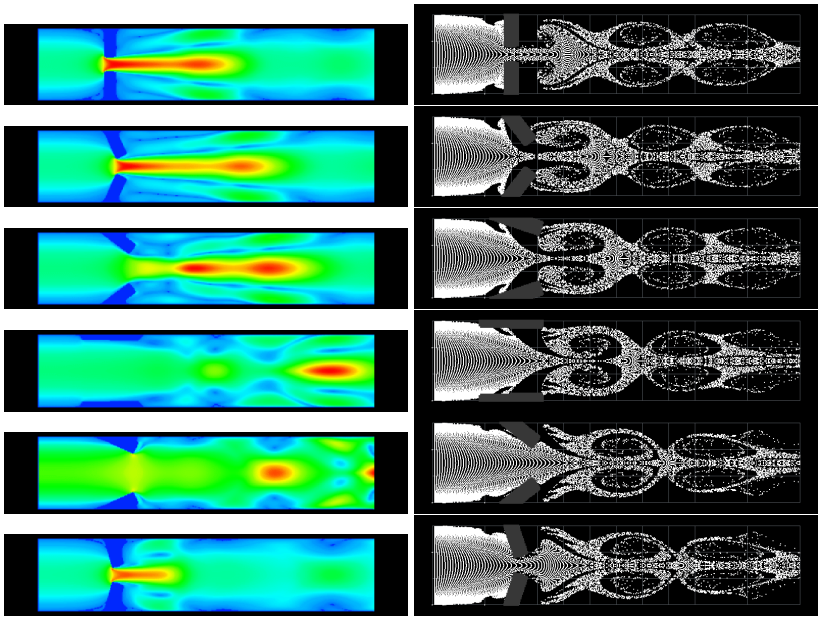


Fig. 1. Prototypical application of the fictitious boundary conditions for 'moving flaps in a channel' (from [12])

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{f} - \nabla p + \nu \nabla^2 \mathbf{u} + \rho \mathbf{j} T \quad , \quad \nabla \cdot \mathbf{u} = 0$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \nu_{\mathbf{T}} \nabla^2 T$$

where $\mathbf{u}$ is the velocity, p the pressure, ν the kinematic viscosity coefficient, ρ the density, $\mathbf{f}$ source term which may include the gravitational force, $\nu_{\mathbf{T}}$ the diffusion coefficient for T which is the prototype for a transported quantity such as temperature, density, viscosity, etc., $\mathbf{j}$ is the unit vector in the y-direction. The Boussinesq approximation for the buoyancy forces is assumed to be valid; i.e., only small temperature excursions from the mean temperature are admitted. If we neglect the quantity T, resp., if $\mathbf{j} = 0$ such that $\mathbf{u}$ and p are independent of T, we obtain the incompressible Navier-Stokes equations. The above equations are to be solved with $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_\partial(\mathbf{x}, t)$ on parts of the boundaries of the flow domain where $\mathbf{u}_\partial(\mathbf{x}, t)$ is the prescribed boundary velocity, including time-dependent moving boundaries.

2.2 Iterative filtering techniques

For a better illustration of these techniques which are not restricted to the Navier-Stokes equations, we consider the following abstract continuous problem given in operator notation which - for reasons of simplicity - may be linked to a typical second-order PDE (for instance, Poisson problem),

$$Au = f\,. \tag{1}$$

For a general variational approach, let V be a Hilbert space with inner product $(\cdot,\cdot)$ and corresponding norm $\|\cdot\|$, and let $a(\cdot\,,\cdot)$ be a bilinear form. We seek a solution to the following variational problem:

$$\text{Find } u \in V, \text{ such that:} \quad a(u,\varphi) = (f,\varphi) \quad \forall \varphi \in V\,. \tag{2}$$

This problem is approximated by a finite element method using a sequence of finite dimensional subspaces "$V_h \subset V$" (also nonconforming FEM are allowed) parameterized by a discretization parameter h, such that the discrete problems read as usually:

$$\text{Find } u_h \in V_h, \text{ such that:} \quad a_h(u_h,\varphi) = (f_h,\varphi) \quad \forall \varphi \in V_h\,. \tag{3}$$

In matrix-vector notation, we derive the discrete linear system

$$A_h U_h = F_h\,. \tag{4}$$

Further, we assume that the continuous problem is defined with corresponding boundary conditions $B(u) = g$ on a boundary part Γ. Let be B a boundary operator which involves a combination of function values and partial derivatives of u on Γ. In the following we imagine for simplicity that we perform Dirichlet boundary values $u = g$ on Γ. Then, besides the possiblity of applying a typical penalty approach, there are the following three possiblities after the discretization process to involve these boundary conditions into the solution process of the matrix-vector problem (4). In all cases, we assume that the matrix A_h (and also U_h and F_h) has not yet incorporated any boundary condition which in fact corresponds to the corresponding natural boundary condition due to the partial integration involved, for instance

$$\partial_n u = 0 \quad \text{ on } \Gamma = \partial\Omega\,. \tag{5}$$

Further, let $S_h(\Gamma)$ denote all degrees of freedom, resp., all components of the solution vector U_h, which are related to the boundary Γ. For example, in the case of bilinear finite elements these are the nodes on Γ (or better: on Γ_h as an approximation to Γ), while for the nonconforming rotated bilinear elements (see [6]) the edges on Γ, resp., on Γ_h are associated. Then, we can proceed as follows to apply Dirichlet boundary conditions:

1) Fully explicit treatment:
We eliminate in A_h all rows and columns belonging to $S_h(\Gamma)$. Additionally, we modify all components of the right hand side vector F_h not belonging to $S_h(\Gamma)$ according to this elimination process. Further, we may prescribe the correspondingly prescribed Dirichlet values for all these components of the vectors U_h and F_h which belong to $S_h(\Gamma)$. But this step is not necessary

since all components of $S_h(\Gamma)$ are treated as being well-known and hence do not longer belong to the set of unknowns. This process is often performed if direct solvers (Gaussian elimination) are applied. However, problems arise if the right hand sides change since corresponding modifications due to (then already) eliminated matrix elements of A_h have to be repeated. As a further consequence, two different matrices may be needed if the same operator A is required with two different boundary conditions, for instance Dirichlet values for the momentum equations and natural settings for associated Poisson problems in the incompressible Navier-Stokes equations.

2) Semi-implicit treatment:
We replace only those rows of A_h by rows of the identity matrix which correspond to $S_h(\Gamma)$. The other rows and columns remain unchanged. Further, during an initialization process only, we have to prescribe the given Dirichlet values for all these components of the vectors U_h and F_h which belong to $S_h(\Gamma)$. This setting guarantees in combination with iterative solvers that the resulting solution vectors always satisfy the prescribed boundary conditions. This approach seems to be the most common in the framework of iterative solution tools. Although the components of $S_h(\Gamma)$ cannot change their value, they belong explicitly to the set of unknowns and are treated by each component of the iterative solver (e.g., smoothing, prolongation, restriction, defect calculation, preconditioning). Again, two different matrices may be needed when two different boundary conditions, for instance Dirichlet and natural conditions, are prescribed for the same operator A.

3) Fully implicit treatment:
We do not modify A_h which consequently is the Neumann matrix due to natural boundary conditions (see [3]). Again, we have to prescribe the given Dirichlet values for all these components of the vectors U_h and F_h which belong to $S_h(\Gamma)$, but before and after each iteration step (*filtering*). All components of the solution vectors have to be treated as unknowns by all components of the iterative solvers. This approach is the most general for iterative solution techniques. Even if different boundary conditions are involved with the same operator A, we can always work with only one matrix A_h. Only the performed *filtering operator* has to be changed.

While the use of direct solution tools often requires the application of the *fully explicit treatment*, all three approaches are equivalent if iterative solvers are applied, and lead to exactly the same results. We prefer the *fully implicit treatment* since this is the most general approach for multigrid solvers and finite element approaches. One important component inside is the *projection filter* which administrates the boundary setting for all vector components of $S_h(\Gamma)$ during the iterative procedure. This tool will be shown to be the essential trick for easy implementations of complicated boundary value settings, in particular for small-scale details due to complicated geometrical boundaries

and time-dependent moving boundary parts, and also with respect to several boundary components for discretely divergence-free FEM [11].

Before we examine more carefully the numerical and algorithmic properties of this approach w.r.t. accuracy and robustness, hereby analysing particularly the corresponding multigrid convergence behaviour, we shortly present another (well-known) 'fictitious boundary condition' method which does not necessarily capture (internal) geometrical objects via the computational mesh, too. Moreover, it does not require (complicated) modifications of the source code due to changing bilinear forms for incorporating a typical penalty approach or for applying boundary conditions: it only prescribes problem-adapted input parameters. In the following, we use specific values for density and viscosity inside of the incompressible Navier-Stokes equations which act as the corresponding filter to satisfy - approximately - the corresponding Dirichlet boundary conditions, at least for homogeneous no-slip conditions.

2.3 Density-viscosity blocking techniques

Since the value of the density or viscosity inside of the Navier-Stokes equations determines the state of fluid, the fluid changes into the state of solid when the values of density or viscosity become infinity. Motivated by this idea, we can use 'huge' values for the density or viscosity parameters to mark a solid body which is positioned in the flow field. So, we can view it as another fictitious boundary method. The difference is that in the previous 'mesh blockage' method, the mesh in the area occupied by a solid body is blocked, while in the 'density-viscosity' blockage method, the density or viscosity parameter inside of the stationary and/or moving solid body is 'blocked'.

It is obvious that this method is the easiest way to incorporate complicated fine-scale structures into the code, since no geometrical adaptations have to be performed and no changes in the code due to implementing boundary conditions are necessary: Only the parameter values for density and viscosity have to be prescribed in an appropriate way! However, it is also well-known that multigrid-like solvers have problems with strongly discontinuous coefficients. Moreover, the influence of the 'strength' of the discontinuity, that means the size of the values for the 'solid' viscosity, resp., density, onto the resulting solution, in particular w.r.t. physically interesting values directly on the surface of the interior objects, is not clear. And, probably the most important criticism, choosing finite parameters for density and viscosity as an approximation for the 'infinite' values of a real solid leads to penetration from the surrounding flow into this object since the corresponding - implicitely given - natural boundary conditions on the interface between the solid and the surrounding liquid do not enforce that $\mathbf{u} \cdot \mathbf{n} = 0$ is valid. However, this internal diffusion is mainly related to the actual size of viscosity and density, and the time scale for observing such perturbations may be much larger than the ac-

tual time interval for the calculation. These considerations will be subject of our subsequent numerical analysis.

3 Numerical tests: Qualitative analysis

In this section we will show several applications of the fictitious boundary methods in order to illustrate their flexibility and their qualitative behaviour for quite complex configurations; a precise comparative study and quantitative examination will be part of the next section. Here, we concentrate on the main characteristics w.r.t. flexibility and applicability in the context of incompressible flow simulations.

3.1 Numerical Examples for the 'Iterative Filtering' technique

Flow generated by an oscillating heated plate in an enclosure box.

In the first example, we consider a unit square with an interior heated plate which is fixed at the center of the box (see Fig. 2). The length of the plate is 0.6, thickness is 0.04. The enclosure boundary conditions consist of no-slip walls, and constant cold temperature ($T = 0$) at horizontal and vertical walls while the surface of the plate is held at a constant hot temperature ($T = 15$). The fluid is initially at rest.

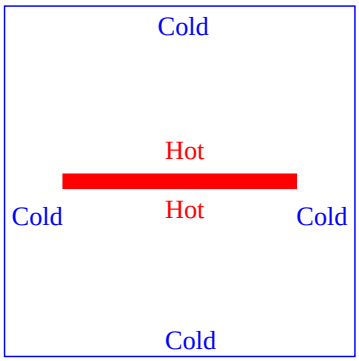
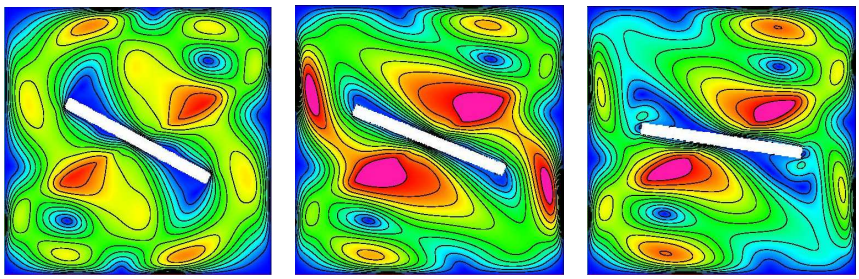


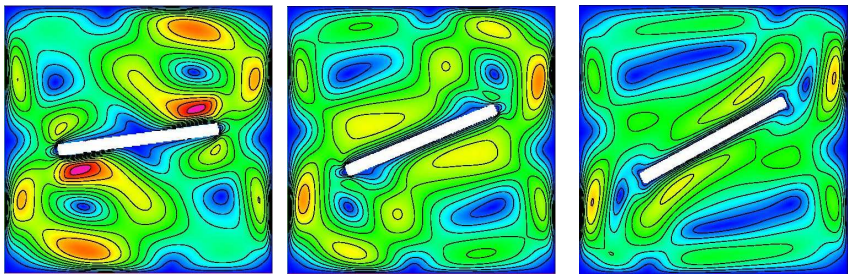
Fig. 2. Description of the 'heated plate in a box' problem

The time-dependent position of the plate is prescribed by a sinusoidally oscillating rotation with angle velocity $\omega = 0.05\pi sin(0.1\pi t)$ (t is the time). We use the fictitious boundary method which does not take care about the employed mesh. Instead, we just need to generate a simple mesh for the whole box, and then use a corresponding filter procedure to represent those elements which are occupied by the plate. Fig. 3, Fig. 4 and Fig. 5 give snapshots for the

contour plots of the norm of velocity, temperature and pressure distributions, respectively. From these pictures, we can see that the fluid motion in the box is initiated by the oscillating rotation of the plate, and the vortex shedding and circulation is generated periodically in the box.



(a) $t = 9.44$ (b) $t = 12.23$ (c) $t = 14.02$

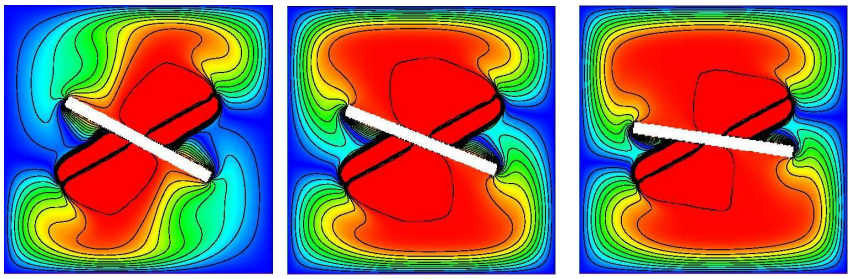


(d) $t = 15.85$ (e) $t = 17.91$ (f) $t = 19.99$

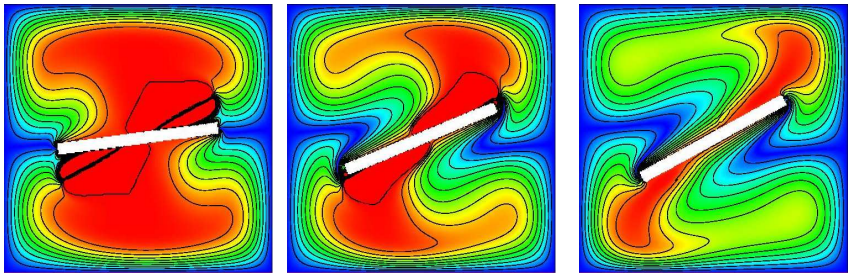
Fig. 3. Oscillating rotation of a heated plate in a box: norm of velocity

Flow generated by an oscillating cylinder in a channel.

The following flow configuration has been chosen to demonstrate the fictitious boundary method for a configuration in which case the flow is disturbed by a transversely oscillating circular cylinder such that vortex shedding is observed. Here, we simulate a cylinder undergoing a sinusoidally transverse oscillation with explicitely specified amplitudes and frequencies. The computational domain size is $(L \times H)$=(2.2×0.41), L being the length of the channel, and H is the width of the channel. The location of the cylinder center (x_0, y_0) is $(0.8, 0.2)$ relative to the left bottom corner of the domain. The cylinder diameter D is equal to 0.2. No-slip condition is prescribed on the walls. The fluid in the channel is initially at rest. The cylinder oscillates sinusoidally such that the location of its center (x_c, y_c) is given by $x_c(t) = x_0 + Asin(2\pi ft)$; $y_c(t) = y_0$, where t is the time, and $A = 0.6$ and $f = 0.5$ are the amplitude and



(a) $t = 9.44$ (b) $t = 12.23$ (c) $t = 14.02$



(d) $t = 15.85$ (e) $t = 17.91$ (f) $t = 19.99$

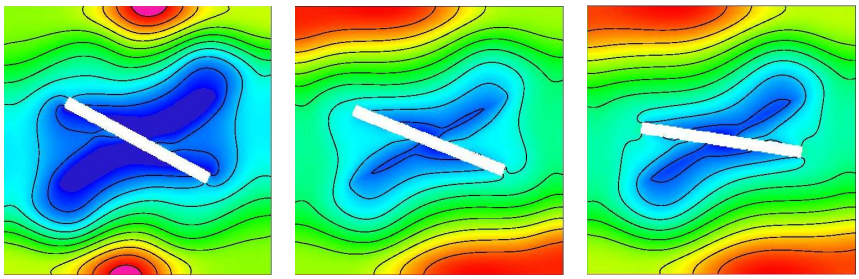
Fig. 4. Oscillating rotation of a heated plate in a box: temperature

frequency of the oscillation, respectively. A nonuniform mesh is used in the simulation such that enhanced resolution is provided in the cylinder vicinity and in the wake. In the horizontal direction, improved resolution is provided up to three diameters on either side of the cylinder location, which is adequate to cover the near wake for all the oscillation amplitudes (see Fig. 6).

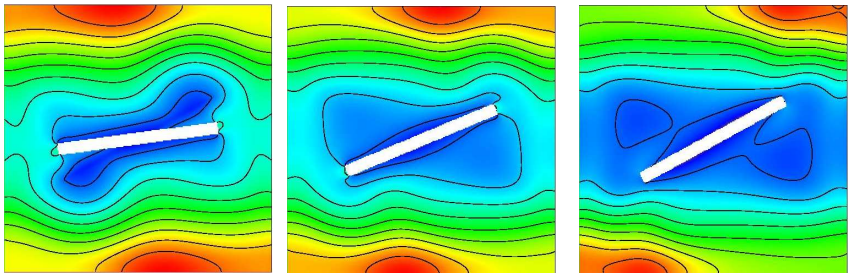
Fig. 7 gives a sequence of contour plots for velocity distributions which show the shedding of vortices when the cylinder is oscillating in the channel. Fig. 8 presents corresponding contour plots for the pressure distribution. From these pictures, we can see that the flow in the channel is significantly disturbed by the oscillating cylinder, and vortex shedding is generated periodically in the wake of the cylinder. The range of the wake becomes longest when the cylinder is at the end of the moving direction ($t = 3.59, 4.44, 5.50$), while the flow is seriously perturbed and more complex ($t = 4.01, 5.08$) when the cylinder is in the middle position of its oscillation.

3.2 Numerical Examples for 'Density-Viscosity' blocking

In this subsection, numerical examples for the flow around single or multiple circular cylinders are presented by using the described density-viscosity blockage method. We can see that the density-viscosity blockage method seems to be a flexible and efficient tool to compute the flow with complex immersed



(a) $t = 9.44$ (b) $t = 12.23$ (c) $t = 14.02$



(d) $t = 15.85$ (e) $t = 17.91$ (f) $t = 19.99$

Fig. 5. Oscillating rotation of a heated plate in a box: pressure

stationary boundaries. However, the numerical tests in the following Section will show that this approach can be quite 'dangerous' since penetration into the 'solid' cannot be avoided by this simple approach. Nevertheless, understanding the different time scales due to the simulation process and due to the perturbations which are introduced by the size of the parameters, a qualitative flow prediction with this methodology seems to be possible.

In the following simulations, the computational domain is $L = 6.0$ and $H = 1.2$, with L and H being the length and the height of the channel. The cylinder diameter D is equal to 0.2. No-slip is prescribed on the walls while natural 'do-nothing' boundary conditions are employed at the right boundary. A parabolic velocity $U = U_\infty(H-y)y$ is prescribed at the inflow (the left part of the channel). The Reynolds number is defined as $Re = U_\infty D/\nu$. Again, a fixed nonuniform mesh is used in the time-dependent simulation (see Fig. 6).

Fig. 9 and Fig. 10 show the numerical results for flow around a cylinder with $Re = 100$. In this case, we set the value of density within the cylinder equal to $10^5\rho$, ρ is the normal density of the fluid. Fig. 9 (a) and (b) are the velocity and vorticity in the region nearby the cylinder area. Fig. 10 (a) and (b) are the corresponding pictures in the whole computational domain. We can see that vortex shedding is initiated behind the cylinder which is imitated by the high viscosity and density values. In Fig. 10, we set the value of the

viscosity within the two circular cylinders into $10^9\nu$, ν is the normal viscosity of the fluid. Fig. 10 (c) and (d) show the velocity and vorticity for the case of Reynolds number $Re = 100$, respectively. We can see that the interaction of vortices behind the wake of two cylinders is obvious and strong. Fig. 10 shows the corresponding flow around three circular cylinder, again with Reynolds number $Re = 100$. In this case, we set both the values of density and viscosity into 'infinity' ($10^5\rho$ and $10^9\nu$, respectively) within the three cylinder areas. Fig. 10 (e) and (f) present the velocity and vorticity, respectively.

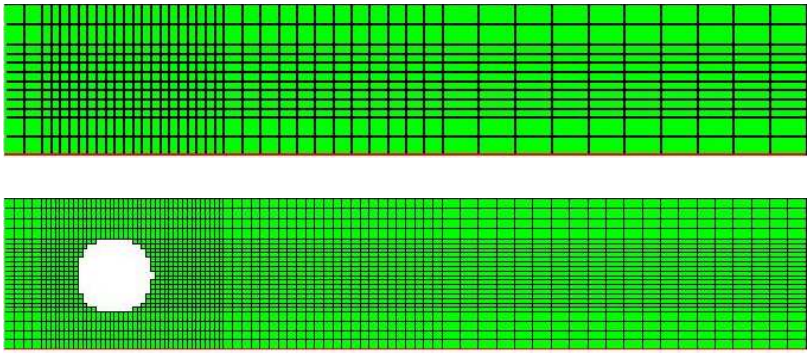


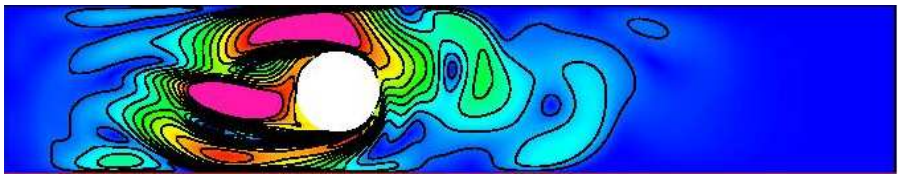
Fig. 6. Complete computational mesh and snapshot for actual 'active elements' on the (globally) refined mesh (Remark: The filtering procedure (de-)activates edges due to the nonconforming FEM approach for the velocity, but not pressure cells!)

4 Numerical tests: Quantitative analysis

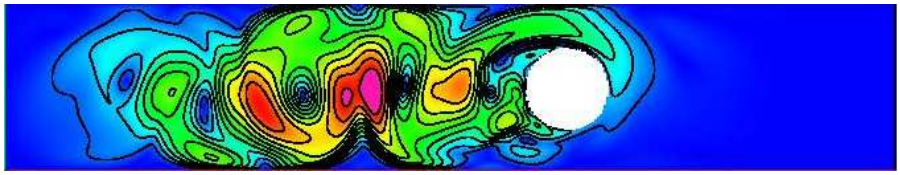
4.1 Numerical analysis of multigrid convergence behaviour

We start with demonstrating the related (linear) multigrid behaviour when we apply the fictitious boundary method together with (level-dependent) filtering processes: here and in the following, the shown coarse meshes are successively refined by connecting opposite midpoints, and LEVMAX corresponds to the highest number of such global refinements. In the first example (see Table 1 for the coarse mesh and a typical snapshot for the pressure), we perform - on the same hierarchy of meshes - calculations for (linear) Stokes flow. We vary the filtering process by applying no filter at all - that means we calculate a simple parabolic channel flow - in comparison to prescribing the corresponding filter for flow around a cylinder on all mesh levels. In addition, we show results for modified configurations which apply this filtering process on highest mesh levels only, starting from the finest level LEVMAX down to level LEVMAX - `LEVFILT`: Choosing `LEVFILT`=1 means that only the two finest meshes correspond to 'flow around cylinder' while choosing `LEVFILT`=2, resp., `LEVFILT`=3

(a) $t = 3.59$



(b) $t = 4.01$

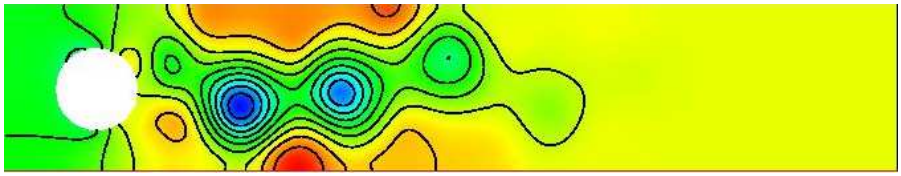


(c) $t = 4.44$

(d) $t = 5.08$

(e) $t = 5.50$

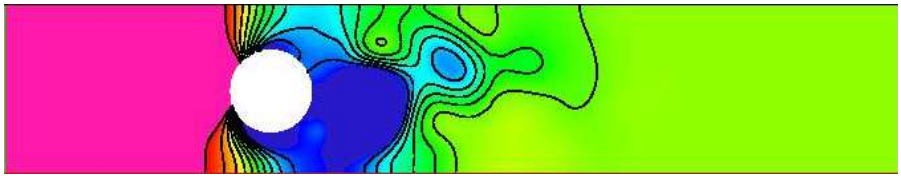
Fig. 7. Sequential contour plots for velocity showing the shedding of vortices



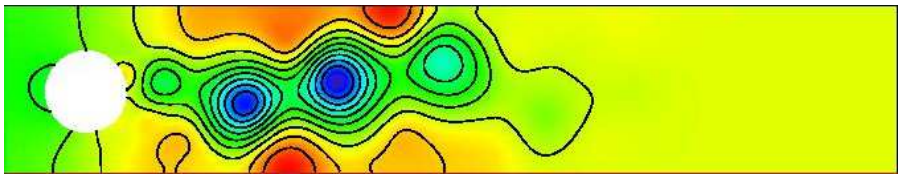
(a) $t = 3.59$

(b) $t = 4.01$

(c) $t = 4.44$



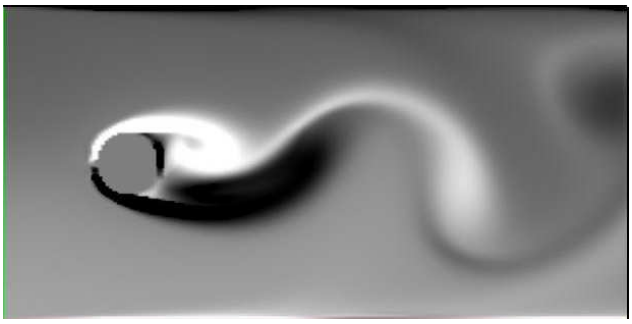
(d) $t = 5.08$



(e) $t = 5.50$

Fig. 8. Sequential contour plots for the pressure distribution

(a) velocity



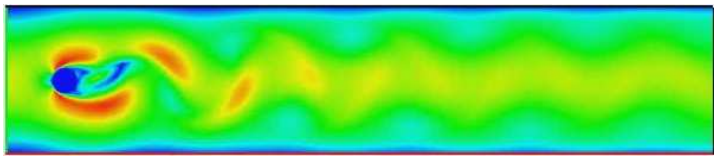
(b) vorticity

Fig. 9. (Zoomed) Flow around a circular cylinder ($Re = 100$)

denote the cases that the three highest, resp., the four highest level in the mesh hierarchy contain the internal object.

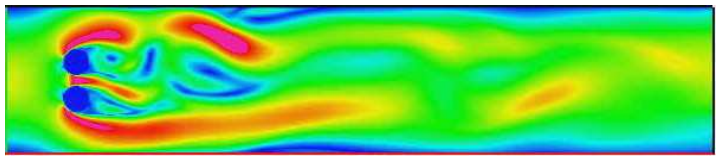
It seems to be sufficient to perform such filter techniques on the three or four finest levels only while on the coarser meshes a simple channel configuration (without any interior object!) can be applied. In these cases, the resulting multigrid rates do not (significantly) differ from the case of applying this filter on all levels: To be on the safe side, the number of smoothing steps should be increased if the filtering process is applied on the finest mesh levels only. This technique of modifying the geometrical details on coarser meshes - taking less objects or coarsening the objects in a level dependent way - has shown to be a powerful tool in the context of multilevel approaches, in particular in 3D (see the examples in `www.featflow.de/album`).

In the next simulations, we employ the same hierarchy of meshes - which do not capture the interior objects - and analyse the nonlinear and linear solution behaviour for a low Reynolds number configurations ($Re \approx 20$) such that the direct steady solver 'CC2D' from FeatFlow [13] can be applied. While the nonlinearity is treated via a standard fixed point-defect correction approach



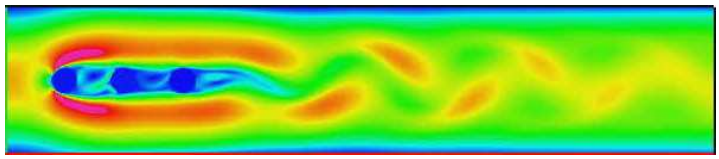
(a) velocity

(b) vorticity

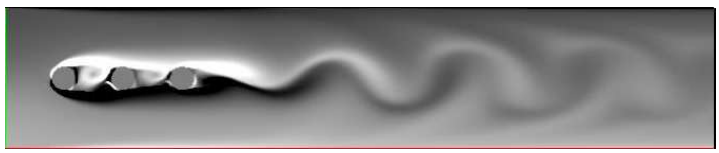


(c) velocity

(d) vorticity

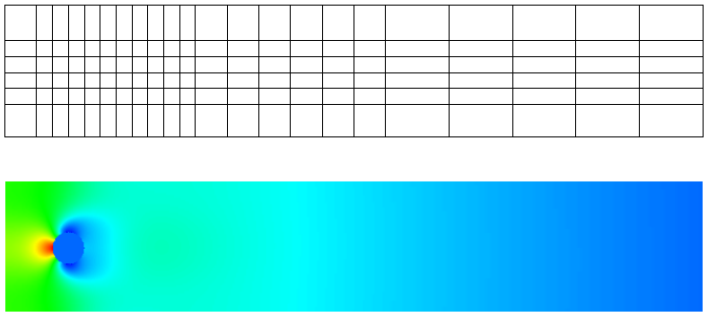


(e) velocity



(f) vorticity

Fig. 10. Flow around one, two and three circular cylinders ($Re = 100$)



LEVMAX	Channel Flow	Circle Flow	LEVFILT=3	LEVFILT=2	LEVFILT=1
3	0.13	0.12	0.12	0.12	0.12
4	0.11	0.11	0.11	0.11	0.11
5	0.11	0.13	0.13	0.20	0.64
6	0.10	0.10	0.11	0.45	0.73

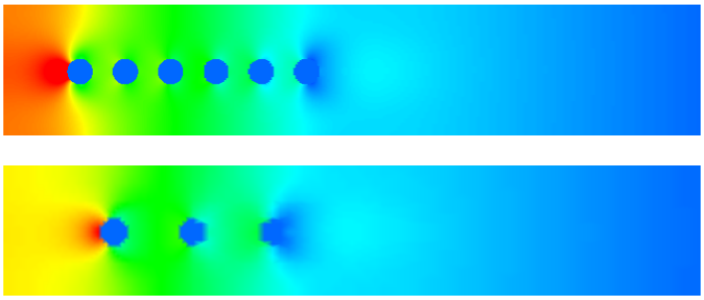
Table 1. Multigrid convergence (CC2D with 4 smoothing steps) for Stokes flow

(see [8] for all details), the linear auxiliary Oseen problems are treated with the same coupled multigrid solver as before. However, in these configurations we vary the number of interior objects between 3 and 6. The configuration '6' in Table 2 corresponds to 6 interior circles on all mesh levels, while 'LEVEL' denotes the case that the number of inner circles is related to the (highest) mesh level LEVMAX. The first number in each column denotes the number of necessary nonlinear iteration steps while the second entry corresponds to the total number of linear multigrid sweeps involved.

These numerical tests demonstrate the predicted result that the nonlinear solution behaviour - for this low Reynolds number - is more or less independent of the number of interior objects and the mesh level. Moreover, the linear multigrid solver for the auxiliary Oseen problems is robust against variations of the filtering process and produces the typical iteration numbers as known for standard approaches which directly integrate the interior objects into the computational mesh.

4.2 Analysis of approximation properties: Stationary case

The aim of the following tests is a systematic examination of the approximation properties of the fictitious boundary method, particularly in combination with the iterative filtering approach. We perform the same stationary simulations (flow around cylinder, $Re = 20$) as before, but on several meshes which incorporate the circle into the (coarse) mesh (CIRC1 - CIRC3) and on various meshes which are full channel meshes (CHAN1 - CHAN4). The following Fig. 11 shows different coarse meshes with various degrees of a priori semi-adapted



LEVMAX	LEVEL	6
3	6/11	6/11
4	6/11	6/11
5	6/11	6/11
6	6/11	6/11

Table 2. Snapshots for the resulting pressure for the '6 circle' configurations on the grid level LEVMAX=6 (top) in comparison to '3 circles' on grid level LEVMAX=3; the table shows the corresponding nonlinear and linear solution behaviour.

refinements near the location of the cylinder which have been successively refined via connecting opposite midpoints.

The following Table 3 shows the results for the velocity (U1,U2) and the pressure (P) in an interior grid point which is part of all types of meshes. Additionally, we measure the 'pressure difference' ΔP which involves the pressure in the point before and behind the circle. In particular, the pressure difference ΔP is a very delicate quantity since it includes the result directly on the interface which is captured by the mesh (CIRC1 - CIRC3) or by the filtering process (CHAN1 - CHAN4).

First of all, we calculated a reference solution (obtained on the mesh CIRC1 with more than 1 million grid points; reference values are also well-known since this is exactly the stationary 2D configuration of the special 'flow around cylinder' benchmark [7]). Based on the reference values, we figure out which level of refinement, that means how many mesh cells, is necessary to obtain an approximation with less than 1%, resp., 5% error. This number can be used as an criterion for rating the approximation properties of the different meshes and boundary implementation techniques since it directly addresses the required mesh width and hence the accuracy of the examined configurations.

The results for such low Reynolds number simulations show that an appropriate global grid refinement (for interior control values, in particular in the right part of the channel which is the case for the evaluation of 'P' and 'U1')

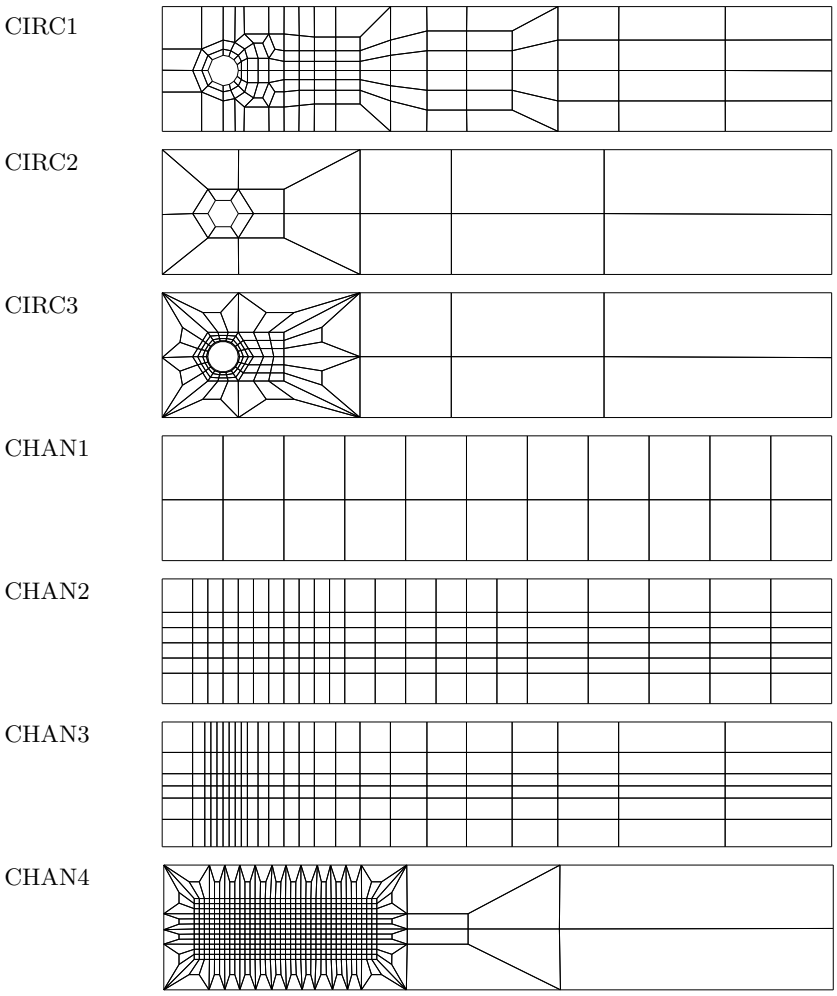


Fig. 11. Different coarse meshes

as well as adequate local mesh adaptation is necessary (in particular for the 'boundary' quantity ΔP). Anyway, the fictitious boundary method with the proposed filtering techniques proves to be competitive with the standard approaches - in terms of necessary grid points, multigrid sweeps and total CPU timing - for such typical CFD applications. In particular, the configurations CIRC1, resp., CIRC2, and the meshes CHAN2, resp., CHAN3 in the context of the fictitious boundary conditions give comparable results, particularly for the weaker, but usually satisfying criterion '5% error'.

5% Accuracy

	CIRC1	CIRC2	CIRC3	CHAN1	CHAN2	CHAN3	CHAN5
U1	2080	5632	32768	5632	8448	9216	9088
U2	2080	352	2048	352	2112	2304	9088
P	2080	1408	8192	352	2112	2304	36352
ΔP	2080	1408	2048	5632	8448	2304	9088

1% Accuracy

	CIRC1	CIRC2	CIRC3	CHAN1	CHAN2	CHAN3	CHAN5
U1	8320	90112	131072	360448	33280	36864	145408
U2	2080	1408	8192	1408	33280	9216	9088
P	8320	22528	131072	1408	2112	9216	581632
ΔP	33280	22528	32768	360448	135168	9216	145408

Table 3. Required number of elements to satisfy the criterion '5% error', resp., '1% error' for the different meshes and control quantities

4.3 Analysis of approximation properties: Nonstationary case

In the next test cases, we examine the resulting accuracy for the same control quantities, now for a medium range Reynolds number which leads to periodical time-dependent vortex shedding behind the cylinder. Again, this flow configuration follows the well-known 'flow around cylinder' benchmark [7] which results in $Re = 100$. Since we are mainly interested in the spatial accuracy of the fictitious boundary method, in particular w.r.t. capturing the important effects near the interior objects, we try to eliminate the temporal discretization error by choosing very small time steps. Then, we proceed the nonstationary simulations until a fully periodical flow behaviour of all quantities has been observed. Finally, we compare the results for one period (see Fig. 12). To be more precise, we figure out how many levels of global grid refinement, that means the required total number of mesh cells, are necessary to produce a flow behaviour - for one period - such that the agreement with a previously calculated reference solution is 'OK', 'GOOD', resp., 'PERFECT' (see the corresponding pictures in Fig. 12).

The following Table 4 shows the resulting number of mesh cells to satisfy the indicated approximation quality for the different meshes, techniques and quantities: CIRC1 - CIRC3 represent the standard approaches which capture the interior object directly via the mesh and appropriate boundary parametrizations. CHAN1 - CHAN4 lead to analogous results, but for a full channel geometry without capturing the interior objects via grid points; the circle is described via the fictitious boundary conditions together with the proposed filtering techniques.

The typical results from such nonstationary simulations for medium Re numbers show that a combination of local as well as global grid adaptation is necessary if all relevant flow scales have to be resolved. Moreover, the results also show that for such 'simple' (geometrical) configurations, the fictitious

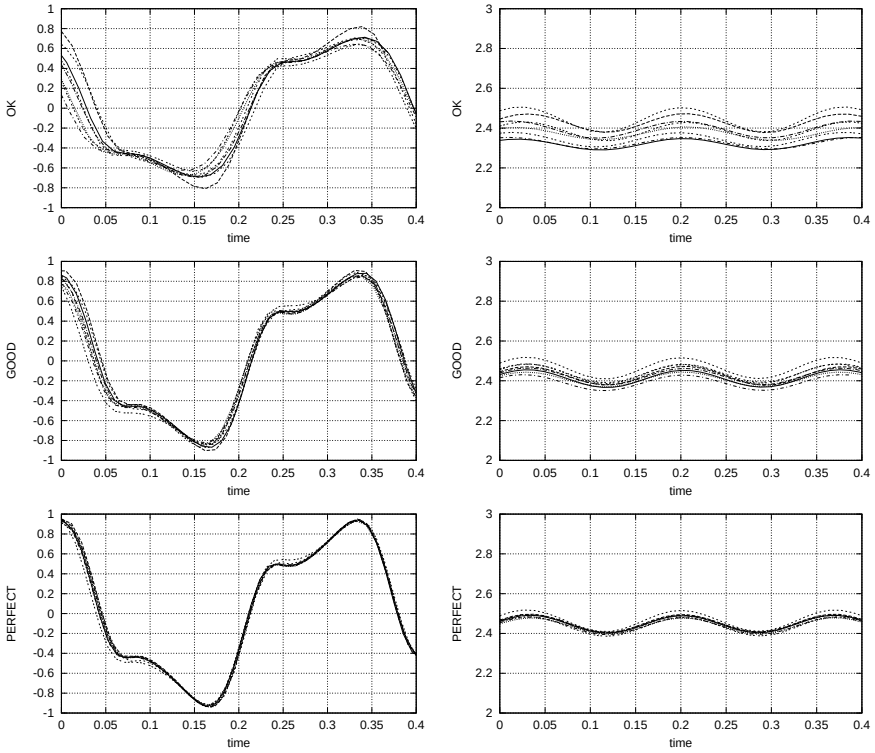


Fig. 12. Fitting of the results w.r.t. the control quantities 'U2' (left) and 'ΔP' (right) for the different categories

boundary method leads to comparative results as the standard approaches with 'body-fitted' meshes, at least if we compare the number of grid refinements, resp., the required total number of mesh cells/grid points in our computational mesh. Since both implementations are almost identical - the only difference is that we have to apply the filtering process after each matrix-vector application which however costs $O(N)$, N the number of unknowns, arithmetic operations only - the resulting CPU times are more or less identical since the number of nonlinear iterations and linear multigrid sweeps is almost the same (see Table 2). Moreover, it is obvious that the right way to get higher (local) accuracy is via local mesh adaptivity: Since the proposed 'fixed mesh' approach leads to a piecewise constant approximation of the boundaries only, the local mesh size has to be decreased in an appropriate way. Another possibility is a local deformation of the grid points to align them with the boundary segments. Using this approach, the local mesh topology ('connectivity') is preserved while a piecewise linear approximation of 2nd order accuracy can be obtained.

'OK'

	CIRC1	CIRC2	CIRC3	CHAN1	CHAN2	CHAN3	CHAN4
U2	8320	22528	32768	22528	8448	9216	9088
ΔP	8320	22528	32768	22528	8448	9216	9088

'GOOD'

	CIRC1	CIRC2	CIRC3	CHAN1	CHAN2	CHAN3	CHAN4
U2	33280	90112	131072	90112	33792	36864	36352
ΔP	33280	22528	131072	90112	33792	36864	36352

'PERFECT'

	CIRC1	CIRC2	CIRC3	CHAN1	CHAN2	CHAN3	CHAN4
U2	133120	360448	524887	360448	135168	147456	145408
ΔP	133120	90112	131072	360448	135168	147456	145408

Table 4. Required number of elements to satisfy the different quality criterions for the proposed meshes/filtering techniques

4.4 Analysis of computational efficiency

The following examples in Fig. 13 show (local) generalized tensorproduct meshes which allow such deformations and adaptive movements of the grid points, however without destroying the connectivity between the grid points. Therefore, locally adapted tensorproduct-like meshes with more or less arbitrary non-equidistant grid stretching can be generated.

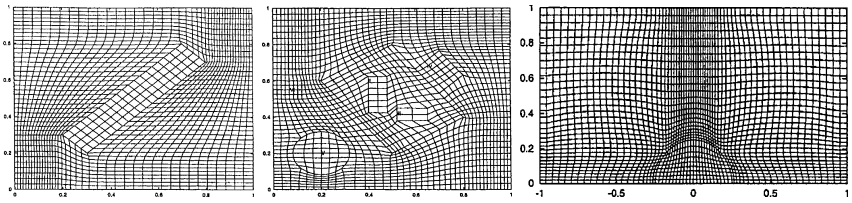


Fig. 13. 'Deformed' and locally adapted generalized tensorproduct meshes for capturing interior interfaces (from [4])

Since our complete computational mesh is built up - on a macro level - by a numerous collection of such generalized tensorproduct meshes, the described mesh adaptation can be performed locally - on certain macros only - such that only a small part of (local) stiffness matrices has to be re-assembled. Moreover, we preserve the full flexibility of standard FEM approaches since the macro structure is allowed to be fully unstructured while the 'substructures' (= macros) consist of many structured parts only which however can be adapted via mesh deformation, mesh alignment or local mesh refinement

with hanging node techniques (see [9]). These meshing techniques have been currently implemented into the FEM package FEAST (see [1]) which is a basic tool box for FEM applications on high performance computers.

The underlying main philosophy in FEAST is to recursively split the global problem into a sequence of local problems, preferably on such *macros* which allow the robust and efficient solution of the auxiliary problems not only w.r.t. high numerical efficiency, but also in terms of very high MFLOP/s rates. The number of arithmetic operations tends to be larger than for the standard approach, but these local operations are cheap [9]: *Data moving, not data processing is costly !* However, these ideas must be combined with corresponding hierarchical data and matrix structures, which can exploit the described tensorproduct-like meshes on each *macro* to achieve the high performance rates for the necessary Numerical Linear Algebra components in the local (multigrid) solvers. And, the introduced fictitious boundary/iterative filtering techniques are important ingredients since this combination allows the use of such highly structured, generalized tensorproduct-like meshes, in particular for time-dependent constellations!

To demonstrate such performance considerations, we give an examples with our FeatFlow code which is applied to a prototypical "2D flow around a car" configuration, and we measure the resulting MFLOP/s rates for the MV multiplication inside of the multigrid solver for the momentum equation (see [8] for mathematical and algorithmic details). We use the typical (for FEM approaches) 'two level' (TL) numbering (old vertices preserve their numbers if the mesh is refined), a version of the bandwidth-minimizing Cuthill-McKee (CM) algorithm and an arbitrary 'stochastic' numbering. Hereby, we apply *sparse MV* concepts which are the standard techniques in FEM codes (and others), also well known as 'compact storage' technique: Depending on the programming language, the matrix entries plus index arrays/lists/pointers are stored as long arrays or heaps, containing the 'nonzero elements' only (for instance, see [18] and the literature cited therein). While this *sparse* approach can be applied for general unstructured meshes and arbitrary numberings of the unknowns, no explicit advantage of (possible) highly structured parts can be exploited. Consequently, a massive loss of performance with respect to the possible peak rates may be expected since - at least for large problems with more than 100,000 unknowns - no 'caching in' and 'pipelining' can be exploited such that the higher cost of memory access will dominate the resulting MFLOP/s rates.

The following Table 5 shows that highly structured MV techniques ('sparse banded BLAS' SBB [9]) can be performed much faster since we can exploit vectorization facilities and data locality. Additionally, we can further differ between the case of *variable matrix* entries ('xxx-V') and *constant bands* ('xxx-C') as typical for Poisson-like PDE's.

Summing up, restringing the grid geometry to a collection of generalized tensorproduct meshes, complete (local) multigrid solvers with very sophisticated smoothers can realized which can actually work with several hundreds

Computer	#Unknowns	CM	TL	STO	ILU-CM	ILU-TL	ILU-STO
	8,320	147	136	116	90	76	72
DEC 21264	33,280	125	105	100	86	73	63
(667 MHz)	133,120	81	71	58	81	52	55
'ES40'	532,480	60	51	21	40	35	22
	2,129,920	58	47	13	38	30	14
	8,519,680	58	45	10	36	30	11

Computer	#Unknowns	STO (CM)	SBB-V	SBB-C	MGTRI-V	MGTRI-C
DEC 21264	65^2	178 (205)	538	795	370	452
(667 MHz)	257^2	110 (224)	358	1010	314	487
'ES40'	1025^2	11 (78)	158	813	185	401

Table 5. MFLOP/s rates for sparse MV multiplication and ILU smoothing in FeatFlow for different numberings and grid levels; the second table shows corresponding performance results on a generalized tensorproduct grid for the Poisson problem, discretized by conforming bilinear FEM. We compare the standard sparse approach (STO/CM) with the 'sparse banded BLAS' techniques (SBB-V/C) and corresponding multigrid solvers MGTRI with very robust 'linewise Gauß-Seidel smoothers'.

of MFLOP/s on recent hardware platforms. Moreover, we can incorporate these 'local structures' into very complex FEM simulation tools on arbitrary domains such that modern numerical components can be applied, as for instance *adaptive meshing* and *a posteriori error control*, or generalized *multigrid/domain decomposition* solvers of SCARC type (see [9]). Since our introduced fictitious boundary methods together with the iterative filtering techniques allow the use of fixed, time-independent meshes - up to local adaptation via mesh deformation and alignement - the demonstrated high numerical efficiency of this approach can be linked with the very high computational efficiency of corresponding data structures, such that the combination of such numerical and algorithmic tools promises a significant performance gain for future simulation codes.

4.5 'Iterative filtering' vs. 'Density-Viscosity blocking'

Finally, we shortly analyse the alternatively proposed fictitious boundary method, namely the addressing of internal objects via prescribing 'huge' values for viscosity, resp., density. It is obvious that this approach seems to be most straightforward since no modification of the source code is required: We only have to prescribe 'appropriate' parameters in the input files. However, there arise very severe difficulties:

1) Typically, the involved multigrid solvers have big problems if such extreme discontinuities arise, in the momentum equation for the velocity (viscosity and density) as well as in the corresponding Pressure-Poisson problem (see [8]) which includes the density. In fact, if the jumps, the means the size of the

discontinuities, is larger than $10^4 - 10^6$, we observe severe convergence problems in our numerical simulations which are based on standard geometrical multigrid components.

2) Another severe problem arises since - due to the finite parameters for the viscosity and density - the true boundary condition $\mathbf{u} \cdot \mathbf{n} \neq 0$ between solid and fluid cannot be guaranteed. In practical applications, this defect results in penetration into the interior objects and corresponding (very small) flow velocities even inside of the 'solids'. However, as the following example in Fig. 14 shows, it may happen that the time scale due to such unphysical penetration effects may be much longer than the time scale for the actual flow simulation. Moreover, this time scale is related to the actual size of the viscosity, resp., density changes such that jumps of size $10^4 - 10^6$ and more may lead to useful simulations. However, one has to keep in mind that larger discontinuities, which lead to a more physical behaviour, lead to more severe convergence problems for the involved multigrid solvers, too.

Moreover, one has to state that the evolution time for such penetration effects is finite as soon as the interior solids are described by huge, but finite values for density and viscosity. While for nonstationary configurations the simulations might lead to useful results (see Fig. 14) due to the differences of the time scales, we observe severe problems for stationary configurations: Since standard Poiseuille flow is a corresponding solution, independent of the values for ρ and ν (Remark: $\mathbf{u} \cdot \nabla \mathbf{u} = 0$, $\nu(x, y) \Delta \mathbf{u} = c\nu(x, y)$ such that $C\nu(x, y)x$ is the corresponding pressure, independent of ρ), the direct stationary approach results in this solution, that means pure channel flow.

3) While the results seem to be quite useful for the velocity and pressure values in interior (grid) points, it is not clear what happens in those vertices/midpoints of the mesh which are directly on the interface between liquid and 'solid' (= interior object). The examples in Fig. 15 show a comparison - on level 6 for CIRC1 and CHAN2 which lead to comparable grid resolution and numbers of elements - for the values in an interior point ('U2', 'P') as well as for the pressure difference ('ΔP') directly on the 'cylinder'. While the results for 'U2' and 'P' lead to very similar results, the pressure difference for the 'density-viscosity blocking' approach is completely wrong and depends massively on the values for density and viscosity. It might be possible to get better values by a better postprocessing, however this study will be subject of future research.

Altogether, it appears that the use of this 'density-viscosity blocking' method may lead to a good agreement of the results with the exact solution, however (up to now) only for interior grid points and the 'right' time scale. Nevertheless, this approach is quite dangerous since the underlying model will lead to wrong results - on the corresponding time scales - which however is very hard to predict by a priori analysis. So, although this approach seems to be very easy to implement and to integrate into existing codes, the problems w.r.t. controlling the effects and solving the resulting systems of equations seem to be very severe.

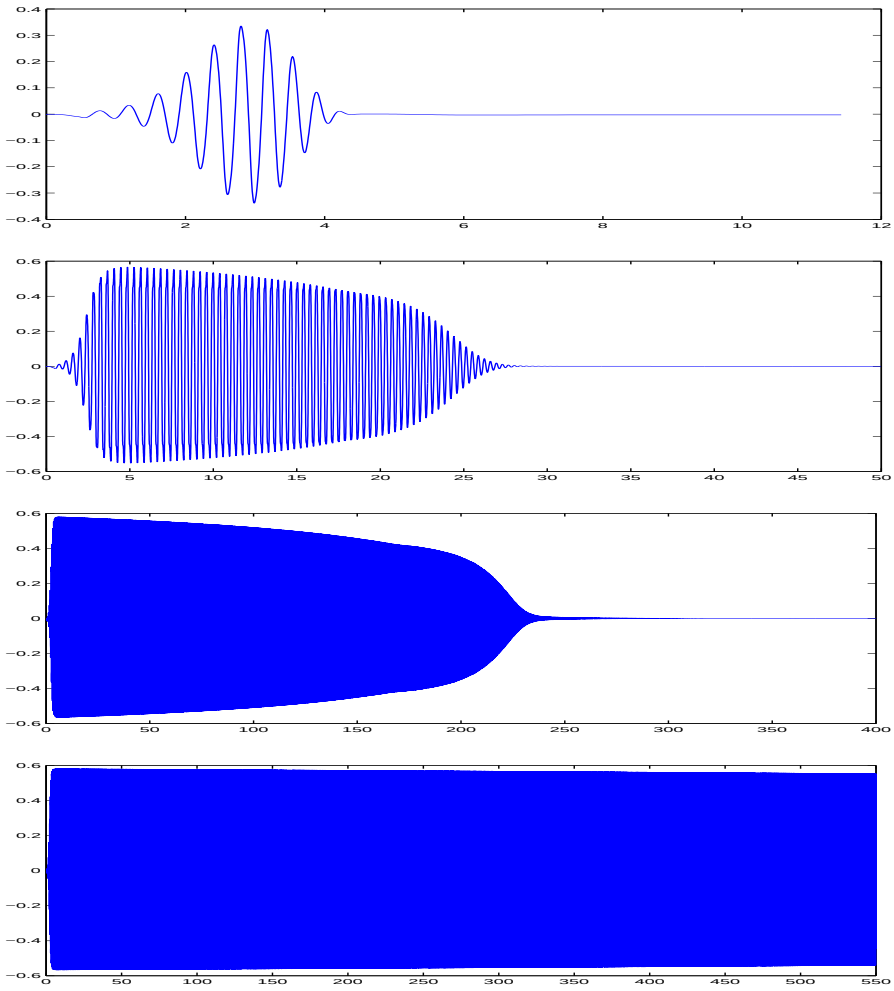


Fig. 14. Nonstationary behaviour of the velocity value 'U2' (mesh CHAN2 and grid level 4) for the density-viscosity jumps of order 10^3 (top), 10^4, 10^5 up to 10^6 (bottom)

5 Conclusions

We do not claim that these 'fictitious boundary' methods together with 'iterative filtering' techniques are superior to the typical approach of resolving accurately small-scale structures by boundary parametrizations and corresponding grid adaption. But there are situations which might require the combination of both techniques. Our numerical experience can be concluded in the following "thumb rules":

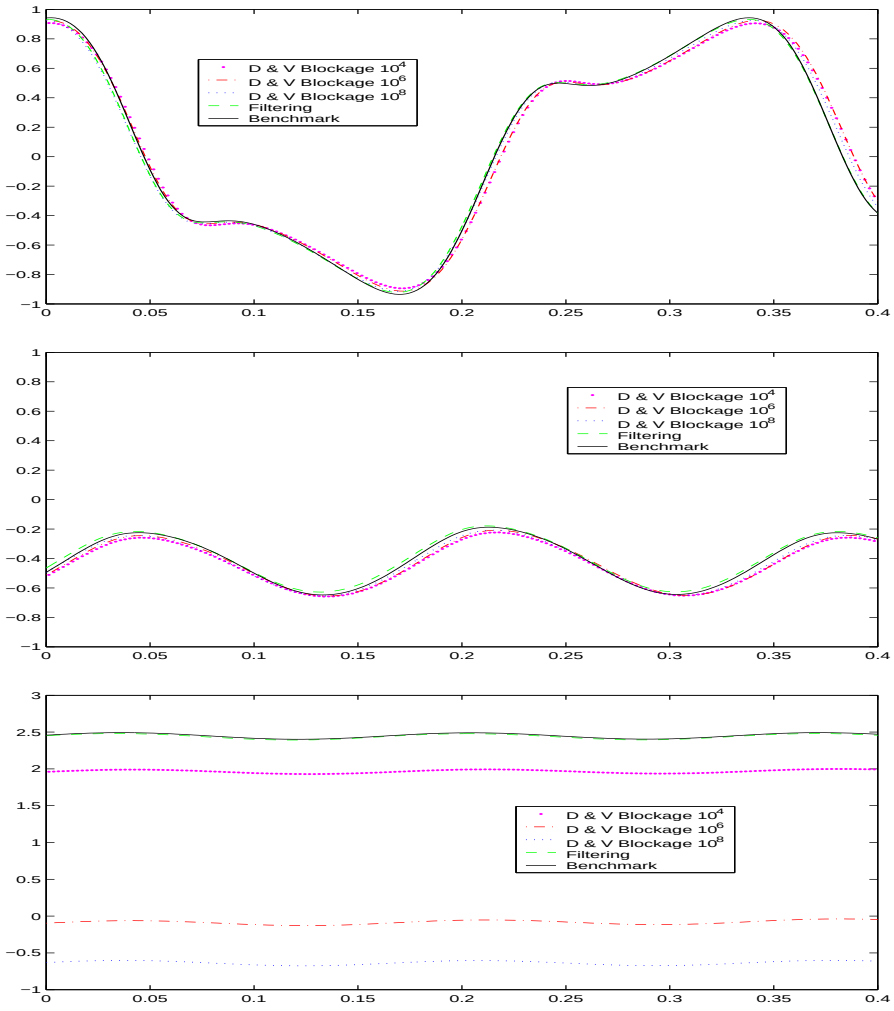


Fig. 15. Comparative study of one period for the velocity value 'U2' (top), for the pressure value 'P' (middle) and for the pressure difference 'ΔP' (bottom)

If the geometry is not too complex, approximate the boundary by corresponding parametrizations (as piecewisely parametrized functions) and via adapted meshes (for instance, objects which are not too numerous and which have a piecewise smooth surface as circles, squares, etc.).

Try to approximate the boundary parts by a "rough" boundary parametrization which contains already most of the important structure. Additionally, apply the iterative filtering *techniques to resolve the fine structures depending on the granularity of the mesh. This approach may be useful if large-scale*

objects with complicated surfaces are approximated, for instance mountains, buildings or car shapes with special features.

If the geometrical details are very small and numerous, perform the fictitious boundary *techniques as an alternative. This approach is advantageous if many (> 100) boundary parts are required (see [12] for the example of a 3D heating device with about 1000 small holes at the outflow and 250 apertures in the interior) or if small-scale objects are involved (for instance, cars with antennae).*

In comparison to the other fictitious boundary condition method, based on the 'density-viscosity blockage', the advantages seem to be more clear. While such discontinuities in parameter choices always lead to trouble with approximation properties ($\mathbf{u} \cdot \mathbf{n} \neq 0$ on the interface) and robustness problems of the involved multigrid solvers - depending on the real size of these parameters - the proposed 'iterative filtering' procedure exhibits a much more robust behaviour, is easily implementable and is very flexible w.r.t. various kinds of boundary conditions. In particular for time-dependent boundary parts, for instance moving objects in the interior, this approach shows a very advantageous behaviour w.r.t. robustness, efficiency, flexibility and also accuracy if additionally ALE techniques are included.

Summing up, the combination of **local mesh adaption** and **iterative filtering techniques** together with **fictitious boundary conditions** may lead to the following advantageous numerical behaviour if these techniques are realized together with appropriate implementation strategies:

1. If these techniques are applied with respect to different levels of refinement, the coarse mesh is allowed to consist of few elements only. If the accuracy of approximating the complex boundary parts is simultanously increased with refining the mesh, the full multigrid efficiency can be obtained. Moreover, it is quite straightforward to perform the corresponding modifications in standard (CFD) codes. Especially for complex 3D geometries (look at `www.featflow.de/album`) and for 'moving obstacles', these techniques provide an elegant and 'quick' way to obtain qualitatively accurate results without paying too much for additional implementation or extended run-time behaviour.
2. The combination of semi–adapted meshes (near boundary parts) with these *fictitious boundary* techniques leads to quite accurate results, in particular due to improved approximation results for locally orthogonal meshes. However, it is clear that the accuracy near (curved) boundary parts is locally of first order only due to a piecewise constant approximation of the interface: Much better results - in a quantitative meaning - will be obtained if additionally local mesh deformation or mesh adaptation concepts are applied to guarantee (at least) a linear approximation of the boundaries. However, one has to take care that the remeshing can be performed in a very local way since time-dependent global remeshing can lead to disastrous CPU timings.

3. The resulting number of unknowns is not 'optimal' since components corresponding to some parts of the boundaries with prescribed values are explicitly involved before they are set again to the prescribed boundary values. However, the employed data structures can be locally regular such that much higher performance in terms of MFLOP/s rates on fast computer platforms can be obtained. (See [9] for a discussion of such approaches which are the basis of our 'hardware-oriented techniques for PDEs'.) The combination of appropriate hierarchical data, solver and matrix structures which can combine the efficiency and accuracy of modern adaptivity concepts and fast multigrid-domain decomposition principles with the huge GFLOP/s rates of modern processors will be the optimal playground for such fictitious boundary and iterative filtering techniques in the future.

Up to now, we have applied these techniques in the case of **explicitely** prescribed boundary parts, resp., movement of solids only. The even more interesting case are free interfaces, multiphase flow, solidification and fluid-structure interaction when the deformation and shape of internal interfaces and boundaries is described implicitely. However, it is obvious that the presented **fictitious boundary methods** together with the **iterative filtering techniques** can be analogously applied. The main difference is that the set of 'active unknowns' for the filtering process has to be calculated in each time step, in most applications via corresponding transport problems involving the fluid velocity **u**. However, as soon as the position of the boundary parts is given which also means the index numbers in the corresponding coefficient vector, the filtering process can be applied in the same way. Recently, we examine these extensions more carefully and we are implementing such techniques in the context of fluid-structure interaction and particularly solidification and free boundaries.

References

1. Becker, Chr., Kilian, S., Turek, S.: Consequences of modern hardware design for numerical simulations and their realization in FEAST. Proceedings "Euro-Par'99 Parallel Processing" (P. Amesto, P. Berger, M. Dayde, I. Duff, V. Fraysse, L. Giraud, D. Ruiz eds.). Toulouse, France, August/September (1999)
2. Fadlum, E.A., Verzicco, R., Orlandi, P. and Mohd-Yusof, J.: Combined immersed-boundary finite difference methods for three dimensional complex flow simulations. J. Comput. Phys. **161** (2000) 35
3. J. Heywood, R. Rannacher, S. Turek: Artificial boundaries and flux and pressure conditions for the incompressible Navier–Stokes equations. Int. J. Numer. Meth. Fluids **22** (1996)
4. Hyman, J.M., Li, S., Knupp, P., Shaskov, M.: An algorithm for aligning a quadrilateral grid with internal boundaries. J. Comput. Phys. **163** (2000)

5. Koopmann, G. H.: The vortex wakes of vibrating cylinders at low Reynolds numbers. J. Fluid Mech. **28** (1967) 501
6. Rannacher, R. and Turek, S.: A Simple nonconforming quadrilateral Stokes element. Numer. Methods for Partial Differential Equations. **8** (1992) 97
7. Schäfer, M., Turek, S.: *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, Vieweg (1996)
8. Turek, S.: Effecient Solvers for Incompressible Flow Problems. Springer Verlag, Berlin-Heidelberg-New York (1999)
9. Turek, S., Becker, Chr., Kilian, S.: Hardware-oriented Numerics and concepts for PDE software. to appear in: Special Journal Issue for *PDE Software*. International Conference on Computational Science ICCS2002, Amsterdam (2002)
10. Turek, S.: Tools for simulating nonstationary incompressible flow via discretely divergence–free finite element models. Int. J. Numer. Meth. Fluids. **18** (1994) 71
11. Turek,S.: Multigrid techniques for a divergence-free finite element discretization. East-West J. Numer. Math. **2(3)** (1994) 3229
12. Turek, S., Kilian, S: The Virtual Album of Fluid Motion. Springer (Multimedia DVD) ISBN 3-540-14900-7 (2002) (`http://www.featflow.de/album`)
13. Turek, S. et al.: **FEATFLOW** . Finite element software for the incompressible Navier–Stokes equations: User Manual, Release 1.2 (1999)
14. Udaykumar, H.S., Mittal, R., Rampunggoon, P. and Khanna, A.: A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. J. Comput. Phys. **174** (2001) 345
15. Udaykumar, H.S., Kan, H.C., Shyy, W. and Tran-Son-Tay, R.: Multiphase dynamics in arbitrary geometries on fixed Cartesian grids. J. Comput. Phys. **137** (1997) 366
16. Wesseling, P.: Introduction to Multigrid methods. Wiley, New York (1992).
17. Ye, T, Mittal, R., Udaykumar, H.S. and Shyy, W.: An accurate Cartesian grid for viscous incompressible flows with complex immersed boundaries. J. Comput. Phys **156** (1999) 209
18. SPARSKIT, `http://www.cs.umn.edu/Research/arpa/SPARSKIT/sparskit.html`