A SPLITTING TECHNIQUE OF HIGHER ORDER FOR THE NAVIER-STOKES EQUATIONS

Jörg Frochte, Wilhelm Heinrichs

Universität Duisburg-Essen. Ingenieurmathematik
e-mail: joerg.frochte@uni-due.de, w.heinrichs@uni-essen.de
Universitätsstr. 3 D-45117 Essen, Germany

Abstract

This article presents a splitting technique for solving the time dependent incompressible Navier-Stokes equations. Using nested finite element spaces which can be interpreted as postprocessing step the splitting method is of more than second order accuracy in time. The integration of adaptive methods in space and time in the splitting are discussed. In this algorithm a gradient recovery technique is used to compute boundary conditions for the pressure and to achieve a higher convergence order for the gradient at different points of the algorithm. Results on the 'Flow around a cylinder'- and the 'Driven Cavity'-problem are presented.

1 Introduction

The time dependent incompressible Navier-Stokes equations are given by:

$$\begin{aligned}
\frac{\partial v}{\partial t} + (v \cdot \nabla)v - \nu\nabla^2 v + \nabla p &= f \quad \text{ in } \Omega,\ t \in [0,\hat{t}] && (1)\\
\nabla \cdot v &= 0 \text{ in } \Omega,\ t \in [0,\hat{t}] && (2)\\
v &= h \text{ on } \partial\Omega,\ t \in [0,\hat{t}]\ , && (3)\\
v &= v_0 \text{ for } t = 0,\ \text{in } \Omega\ . && (4)
\end{aligned}$$

The solution of these equations on the time interval $[0,\hat{t}]$ are the velocity v of a Newtonian fluid with the kinematic viscosity ν and the pressure p in a domain Ω. We assume that Ω is a bounded domain in $\mathbb{R}^2$ and that its boundary $\partial\Omega$ is polygonal. The boundary conditions are given by a function h on $\partial\Omega$.

We start by introducing a splitting technique for the Navier-Stokes equations with finite elements which is related to the one published by Haschke and Heinrichs [9] for spectral methods. We call this algorithm the *base splitting algorithm* and introduce it in section 2. Like most other splitting techniques it seems to be restricted to second order in time. The major reason for this is the fact that especially for higher Reynolds numbers it has not been possible so far to construct a stable pressure extrapolation of an order higher than one. With an approximation of first order the analysis done by Heinrichs in [10] for the Stokes equations gives a restriction to second order in time for splittings of this type.

To negotiate this problem we choose a hierarchy of finite element spaces in section 5 and conveniently nest two base splitting steps. The result is a technique of higher order in time that generally reduces the CPU costs compared to the base splitting with the same number of unknowns. So this technique can alternatively be seen as a postprocessing or a preconditioning splitting step.

For splitting techniques the boundary conditions for the pressure are always a challenge with a long history see e.g. [23], [15]. For the boundary conditions of the second-order pressure equation we refer to the discussion in the papers of Karniadakis et al. [13] , Maday et al. [16] and Timmermanns et al. [17] and [20]. For the computation of the pressure boundary conditions in section 2 we used a variation of the formulation given in [13] by Karniadakis.

To use this technique we have to evaluate the laplacian operator with linear finite elements which leads to a couple of problems that can be avoided using a gradient recovery technique. Beyond this such a gradient recovery technique can be used at different points of the algorithm to increase the accuracy in space, because for linear finite elements the convergence rate of the gradient is only of first order. Some gradient recovery techniques like the Z^2 gradient recovery [24] are less accurate at the edges of Ω where we want to compute boundary conditions. So in section 3 we develop a new gradient recovery technique for this splitting with better recovery results at $\partial\Omega$.

To test our splitting scheme on appropriate examples we use two different strategies. One quite common way to get appropriate examples consists of choosing a velocity/pressure pair $(v;p)$ and setting the right-hand side and the boundary conditions so that $(v;p)$ fulfills the Navier-Stokes equations. With this strategy it is easy to compare the finite element solution with the exact one. We tested the splitting on some examples of this type and present the results on two of them in this paper. However, a solution $(v;p)$ chosen in this way has in general no physical meaning. So beyond this we tested the algorithm on some standard CFD problems in section 6, the 'Flow around a cylinder'- and the 'Driven Cavity'-problem.

2 The stabilised base splitting

For the approximation of $\frac{\partial}{\partial t}$ we use a BDF scheme of third order. The leading coefficient of the BDF scheme is denoted with β_0 and the time step size with $\triangle t$. Similar to the splitting for spectral methods [9] one time step of the splitting follows this scheme:

Time step in the base splitting

1. Compute a guess ($\bar{p}^{m+1}$) for the pressure
2. Based on the pressure compute an intermediate velocity $\tilde{v}^{m+1}$
3. Solve the Poisson equation

$$\begin{aligned} -\nabla^2 p_{update} &= -\frac{\beta_0}{\triangle t}\nabla \cdot \tilde{v}^{m+1} && (5)\\ n \cdot \nabla p_{update} &= n \cdot 0 \text{ on } \partial\Omega && (6) \end{aligned}$$

   for the pressure and velocity update

4. Apply the update by

$$\begin{aligned} p^{m+1} &= \bar{p}^{m+1} + p_{update} && (7)\\ v^{m+1} &= \tilde{v}^{m+1} + \frac{\triangle t}{\beta_0}\nabla p_{update} && (8) \end{aligned}$$

The finite element spaces for the velocity and the pressure are chosen to fulfill the inf-sup-condition, so we use triangle Taylor-Hood elements with linear and in the context of the postprocessing also with quadratic base functions.

In [9] $\bar{p}^{m+1}$ is chosen $\bar{p}^{m+1} := \bar{p}^m$. This choice has two disadvantages.
The first one concerns the boundary conditions. With this choice the pressure function is sticked to unnatural homogeneous Neumann boundary conditions. Unfortunately it is problematic to integrate fitted boundary conditions in the Poisson problem (5), (6) because a too high accuracy is necessary to compute stable adapted boundary conditions. The reason for the need for high accuracy is that, particularly for a small time step size, an erasement of all trusted decimal places is possible if p does not change too much, $p^{m+1} - p^m = p_{update} \approx 0$ on $\partial\Omega$.

The second disadvantage concerns e.g. functions of the type $v(x, y, t) = z(t) \cdot w(x, y)$. Because of a kind of memory effect for such functions this method is not unconditionally stable for finite elements. In such cases the structure of v does not change in time. So the same e.g. mesh based errors that appear when solving the Poisson problem (5), (6) are added stepwise to the pressure in step 4 of the algorithm.

For small time step sizes the factor $\frac{\beta_0}{\triangle t}$ on the right side of (5) amplifies this effect which is compensated in (8) but not in (7).

For the used linear finite elements this leads to an unstable algorithm. To avoid this we have to look for a way to compute p without a memory effect. To do this we use (1) together with the fact that v is divergence free. If we apply $\nabla\cdot$ to (1) the linear terms $\frac{\partial v}{\partial t}$ and $\nu\nabla^2 v$ on the left side are eleminated. To evaluate the non-linear term we use the velocity and the right side f of the last time step. It turned out that a mixed-formulation of v^m and f^{m+1} is unstable in some cases as well as using an extrapolation of a higher order for the velocity. After all we end up with a Poisson equation for the pressure guess. This technique prevents the mentioned memory effect and the associated unstablity. The price is another Poisson equation to solve. This can be done very effectiv because the Galerkin matrix is constant during the whole simulation. So if the dimension allows the use of a sparse solver like UMFPACK [3], which was the case for all problems presented in this paper, the decomposition can be computed once and used during the whole simulation. If the dimension of the problem is beyone the scope of a sparse solver a sparse approximate inverse like [5] could be computed once and being used during the whole simulation.

So in difference to [9] we compute $\bar{p}^{m+1}$ by the following Poisson equation:

$$\begin{aligned} -\nabla^2\bar{p}^{m+1} &= -\nabla\cdot f^m + \nabla\cdot((v^m\cdot\nabla)v^m) &(9)\\ \underbrace{\Leftrightarrow}_{\nabla\cdot v=0} -\nabla^2\bar{p}^{m+1} &= -(f^m_{1x}+f^m_{2y}) + v^m_{1x}v^m_{1x} + 2v^m_{2x}v^m_{1y} + v^m_{2y}v^m_{2y}\ . &(10)\end{aligned}$$

All partial derivations on the right side were constructed with the later presented gradient recovery technique, which for sufficiently smooth v increases the accuracy of the right side data. To transfer the velocity and pressure data between the finite element spaces we use restringation or prolongation operators commonly known from multi-grid techniques.

The same technique is used to compute the Neumann boundary conditions for the PDE. These conditions as well are taken directly from the Navier-Stokes equations (1):

$$n\cdot\nabla\bar{p}^{m+1} = n\cdot(f - (\underbrace{\frac{\partial v^m}{\partial t} + (v^m\cdot\nabla)v^m}_{(*)} - \nu\nabla^2 v^m)) \quad \text{on } \partial\Omega\ . \tag{11}$$

The term $(*)$ is zero for homogeneous zero Dirichlet boundary conditions in v. In the case other boundary conditions are given $\frac{\partial v}{\partial t}$ is approximated with a BDF scheme of third order or taken from the given boundary conditions because we are only interested in $\frac{\partial}{\partial t}v^m$ on $\partial\Omega$. In the equation (11) we rewrite the Laplace term using the fact that $\nabla\cdot v = 0$

$$\nabla^2 v_1 = v_{1_{yy}} - v_{2_{yx}}\ ,\ \nabla^2 v_2 = v_{2_{xx}} - v_{1_{xy}}\ .$$

For an approximation of the Laplace operator this formulation is more accurate than $v_{i_{xx}}+v_{i_{yy}}$ $(i=1,2)$. Later in section 3 we will comment further on this.

With the coefficients of the BDF scheme $\beta_j (j = 1..3)$ we set

$$\tilde{f} = f - \nabla\bar{p}^{n+1} - \frac{1}{\triangle t}\sum_{j=1}^{3}\beta_j v^{m+1-j}$$

and so the intermediate velocity can be computed explicitly

$$\left(-\nu\nabla^2 + \frac{\beta_0}{\triangle t} I\right) \tilde{v}_i^{m+1} \quad = \quad \tilde{f}^{n+1} - (v_e \cdot \nabla) v_e \tag{12}$$

or implicitly

$$\left(-\nu\nabla^2 + \frac{\beta_0}{\triangle t} I\right) \tilde{v}_i^{m+1} + (v_e \cdot \nabla)\, \tilde{v}_i^{m+1} = \tilde{f}^{n+1} \tag{13}$$

using a kind of Picard iteration.

The initial value for this iteration is $v_e = v^m$ and after every iteration we set $v_e = \tilde{v}_i^{m+1}$. This continues until the stop criterion $\|\tilde{v}_i^{m+1} - \tilde{v}_{i-1}^{m+1}\| < \varepsilon$ is fulfilled. The boundary conditions for (12) , (13) are taken from (3).

3 The Taylor based gradient recovery technique

This section focuses on the gradient recovery technique mentioned in the abstract of this paper. It is a technique specially developed for this problem but the use it not limited to the presented splitting technique. Like the Z^2 gradient recovery [24] it can also be used as an error indicator or generel method for a postprocessing gradient recovery.

Let $\mathcal{T}_h$ be a triangulation of Ω and $T \in \mathcal{T}_h$. Thus the linear finite element space is $V_h = \{u_h \in C(\bar{\Omega})\ ; u_{h|T} \in \mathcal{P}_1 \text{ for } T \in \mathcal{T}_h\}$. To motivate this gradient recovery technique we assume that $u \in C^2(\Omega)$ and $I_h u = u_h \in V_h$ with I_h as interpolation operator on V_h. To recover the gradient of u at a node a of $\mathcal{T}_h$ we use a second order Taylor approximation with the values of u_h at a and $n \geq 5$ nodes (x_j, y_j) in the neighbourhood of a:

$$u_h(x_j, y_j) - u_h(x_a, y_a) = \mathbf{u_x}(\mathbf{x_a}, \mathbf{y_a})(x_j - x_a) + \mathbf{u_y}(\mathbf{x_a}, \mathbf{y_a})(y_j - y_a) \tag{14}$$

$$+\frac{1}{2}(\mathbf{u_{xx}}(\mathbf{x_a}, \mathbf{y_a})(x_j - x_a)^2 + \mathbf{u_{xy}}(\mathbf{x_a}, \mathbf{y_a})(x_j - x_a)(y_j - y_a) + \mathbf{u_{yy}}(\mathbf{x_a}, \mathbf{y_a})(y_j - y_a)^2) \tag{15}$$

The bold marked terms are the unknowns that are to be computed by solving a $5 \times n$-least squares problem. Generally all neighbours of a and also their neighbours are chosen. Figure 2 shows an example for such a neighbourhood of a. The new Taylor-based recovery technique (TBR) uses the data from all displayed nodes while a technique like the Z^2 recovery [24] only uses the information from the nodes with filled circles. The greater database together with a proper weighting improves the results, especially on adaptive refined meshes and at the edges of Ω.

The weighting is done as follows: We differ the nodes if they are located above $U_1 := \{x_j, y_j | y_j > y_a\}$, below $U_2 := \{x_j, y_j | y_j < y_a\}$, right $U_3 := \{x_j, y_j | x_j > x_a\}$ or left $U_4 := \{x_j, y_j | x_j < x_a\}$ from a. It is clear that for $\hat{U}$ the set of all nodes

$$\hat{U} = \bigcup_{i=1}^{4} U_i\ ,$$

with $U_1 \bigcap U_2 = \emptyset$ and $U_3 \bigcap U_4 = \emptyset$. Furthermore we define :

$$h_{1j} = x_j - x_a \qquad h_{2j} = y_j - y_a \qquad d_j = \frac{1}{\sqrt{h_{1j}^2 + h_{2j}^2}}\ ,\ (j = 1...m)$$

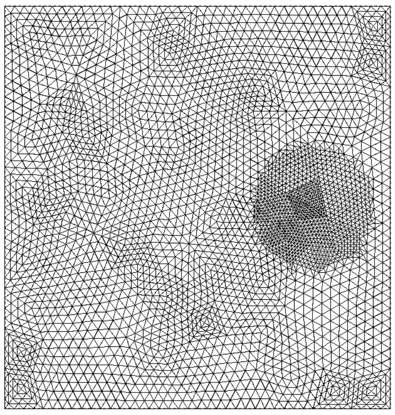
Figure 1: Mesh with 3233 unknowns

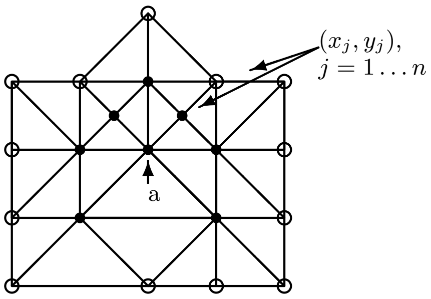

Figure 2: Database of G_T/TBR and the Z^2 gradient recovery technique

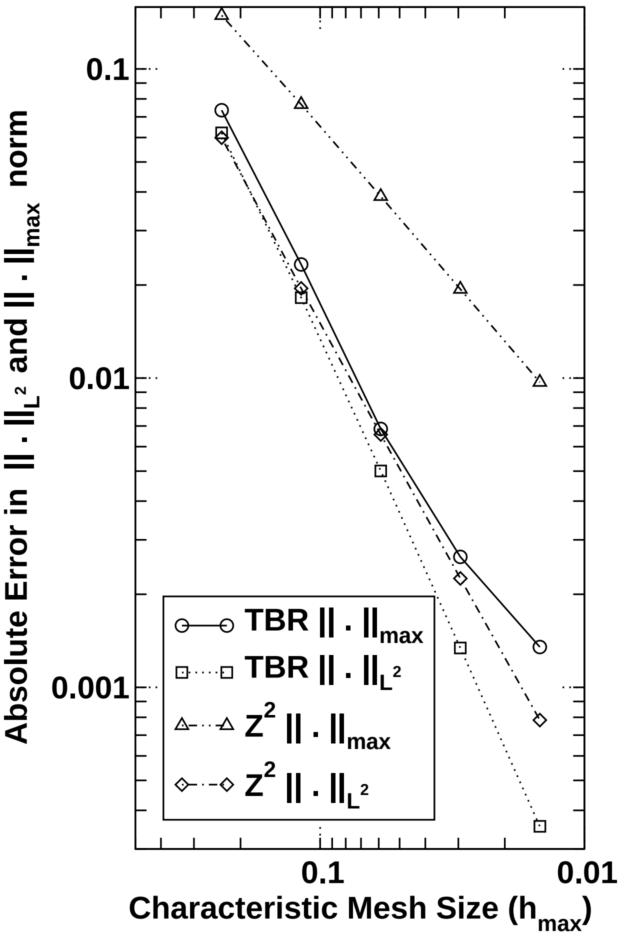

Figure 3: Error TBR and Z^2

$$g_{1j} = \frac{h_{1j}}{d_j} \qquad g_{2j} = \frac{h_{2j}}{d_j} \qquad \alpha_i = \begin{cases} 1 \text{ , if } (x_j, y_j) \in U_i \\ 0 \text{ , if } (x_j, y_j) \notin U_i \end{cases}$$

$$G_1 = \sum_{(x_j,y_j)\in U_1} |g_{1j}| \quad G_2 = \sum_{(x_j,y_j)\in U_2} |g_{1j}| \quad G_3 = \sum_{(x_j,y_j)\in U_3} |g_{2j}| \quad G_4 = \sum_{(x_j,y_j)\in U_4} |g_{2j}| \; .$$

To give a higher weight in the least squares problem on nodes near a we scale each equation with $\frac{1}{d_j}$. So finally we get the following weighting factor for an equation j in the least squares problem

$$w_j = \sum_{i=1}^{4} \alpha_i \frac{|h_{1j}| + |h_{2j}|}{d_j G_i} \; , \; (j = 1...m) \; .$$

Figure 3 shows the results of the two techniques recovering the partial derivation u_{hx} on a mesh like the one in figure 1. The data for the gradient recovery derives from a function

$$u_h \approx \sin(\pi(x-1)/2)\sin(\pi(y-1)/2)$$

which is the solution of a Poisson equation

$$\begin{aligned} -\nabla^2 u &= f, && (16) \\ u &= 0 \text{ on } \partial\Omega \; , && (17) \end{aligned}$$

with $\Omega = [0,1] \times [0,1]$. Figure 3 illustrates the fact that the TBR technique shows higher reduction rates in the L^2 norm. This technique can be used for any set of points and is not limited by its construction to linear base functions.
Very important for the computation of the needed boundary conditions for the pressure is the error in the nodal maximum norm because the maximum error often occurs at the edges of Ω.

If this technique is used for all nodes of a triangulation we will use this according to the approximated nabla operator by $G_T u_h$. The reconstruction of second order derivations at the edges still causes more problems than the recovery of the first order derivations.

It is well known that splitting techniques generally are designed for problems with a small viscosity, see e.g. [21] page 21f. For all practical problems in fluid dynamics this is not a drawback, because the kinematic viscosity is very small, e.g. $6.8 \cdot 10^{-4}$ for glycerin and $1 \cdot 10^{-6}$ for water with 20 centigrade.

Additionally to this generall property of splitting techniques the computation of the boundary values focus the case of small kinematic viscosity. A theoretical problem with a hugh kinematic viscosity may cause problems concering the boundary values. In this case it may be better to choose homogeneous Neumann boundary conditions for the pressure.

But generally - compared with the other terms - the very small kinematic viscosity ν heavily reduces the influence of second order derivations at the edges.

Nevertheless, the focus on a small kinematic viscosity is not a drawback.

Because of the general design of splitting techniques for problems with a small viscosity - we consider cases with $10^{-6} \leq \nu \leq 1$ - this is not an additional constraint. This special recovery of the boundary conditions may only amplify this effect for a huge kinematic viscosity.

Concerning the presented recovery technique itself we can sum up that as an error indicator the Z^2 will generally be a better choice because the CPU costs are lower. For a postprocessing gradient recovery the presented technique is more attractive because it achieves a higher accuracy.

4 Numerical Results for the base splitting

We implement the splitting scheme using Taylor-Hood elements with linear base functions.

To measurerror of reduction rate in time the error based on the discretisation in space has to be sufficiently small or it will tend to dominate the measured error. After the Splitting the equations are related to parabolic problems, for which this is a common fact, see e.g. [22]. This effect is amplified for the implicit treatment (13), because in this case we deal with convection dominated equations that tend to be less accurate compared to diffusion dominated equations. For details and error estimations for streamlinediffusion stabilised finite elements (SDFEM) see e.g. [18] p.231-233 or [14] p.325-329.

To avoid this it is possible to use two strategies. On the one hand we could perform the test for all Δt on a constant, fine grid. In this case one would observe that the measuring error for $\Delta t \to 0$ converges against the error in space. On the other hand we could start with a quite coarse grid and refine it if the reduction rate is too much affected by the error in space.

We used the second approach because this also avoids conflicts with the CFL condtion which can be important for the explicit picard iteration described at the end of section 2.

Nevertheless, it can be practicaly impossible to perform another global refiment because of the quadratic grow of the number of unknows. In particular if the order in time is higher than the order in space, as we will see in section 5, one bisection of Δt would require more than one global regulare refinement in space.
If a high accuracy in space is needed one should think of a local priori refinement, like it was performed for the 'Flow around a cylinder'- and the 'Driven Cavity'-problem in section 6. An adapitve refinement in space during the simulation is generally not suggested because it requires the addition of divergence free node to past values of v. If the divergence-freedom guarantee can not be guaranteed this may perturb the splitting.

Beside the 'Flow around a cylinder'- and the 'Driven Cavity'-problem we only use global regular refinements.

The first test problem (**I**) over the unit square $\Omega = [0,1] \times [0,1]$ is the same one as in [9] with the right side and the boundary conditions so chosen that the solution is

$$\begin{aligned} v_1(x,y,t) &= \cos(5t)(\sin(\pi x/2)\cos(\pi y/2)) \ , \\ v_2(x,y,t) &= -\cos(5t)(\sin(\pi y/2)\cos(\pi x/2)) \ , \\ p(x,y,t) &= \frac{\cos^2(5t)}{4}(\cos(\pi x)+\cos(\pi y)) + 10\cos(5t)(x+y-1) \ . \end{aligned}$$

The second test problem (**II**) is computed over $\Omega = \{(x,y) \in \mathbb{R}^2 | 1 \le r \le 2\}$ with $r = \sqrt{x^2+y^2}$. Again we choose the right side and the boundary conditions such that the solution is

$$\begin{aligned} v_1(x,y,t) &= -y(0.25-(r-1.5)^2)\sin(2\pi t) \ , \\ v_2(x,y,t) &= x(0.25-(r-1.5)^2)\sin(2\pi t) \ , \\ p(x,y,t) &= y\sin(x)\sin(2\pi t). \end{aligned}$$

The results e.g. from figure 4 and 5, the results presented later in table 1 and other data on different problems, see [6], verify that the base splitting is of second order in time for the

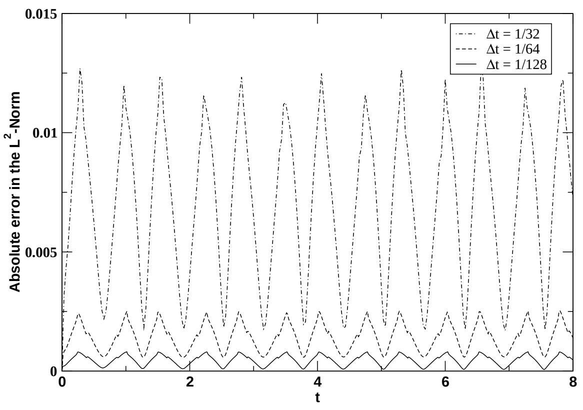

Figure 4: L_2-Error in v_1 on the test problem I with $\nu = 1/1000$

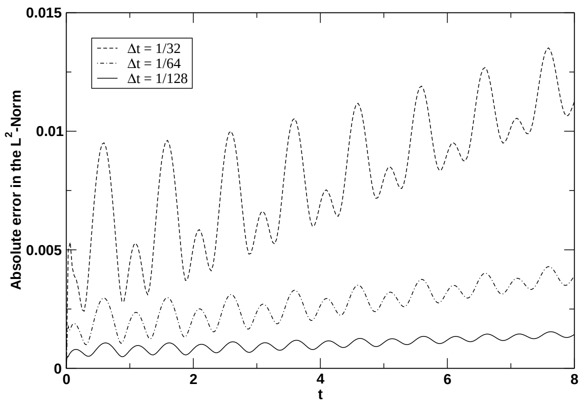

Figure 5: L_2-Error in v_1 on the test problem II with $\nu = 1/5000$

velocity and the pressure. As one can see in the figures 8 and 9 the explicit strategy works for Reynolds numbers Re less than 2000. The implicit strategy has been tested up to a Reynolds number of 10000 with streamlinediffusion stabilised finite elements, see e.g. figures 6 and 7.

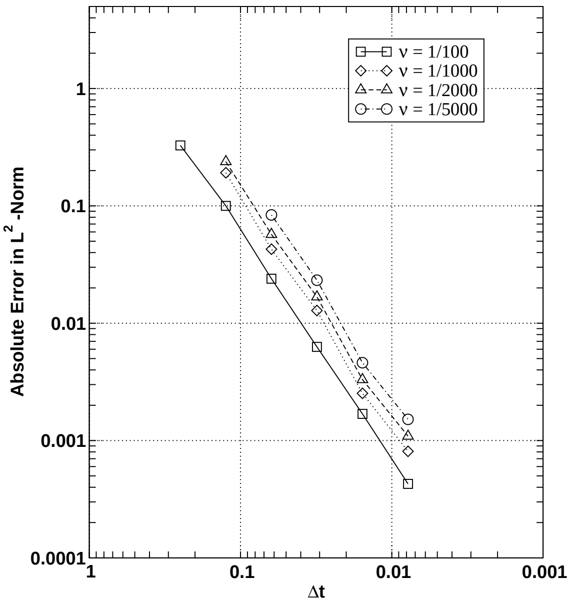

Figure 6: Absolute L^2-Error of v_1 on the test problem I (implicit)

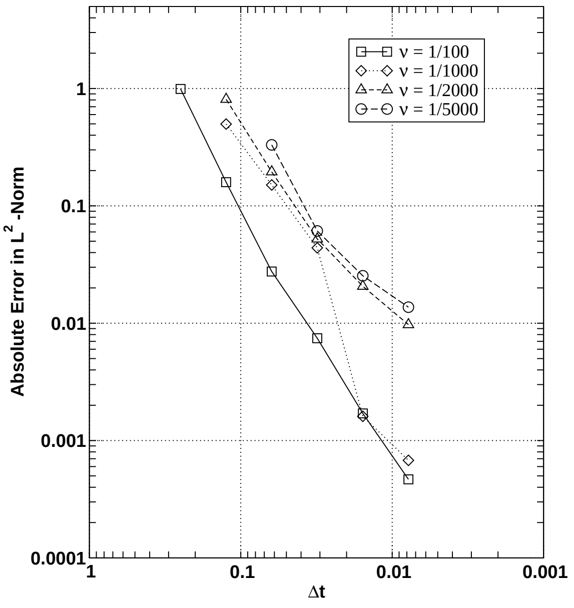

Figure 7: Absolute L^2-Error of p on the test problem I (implicit)

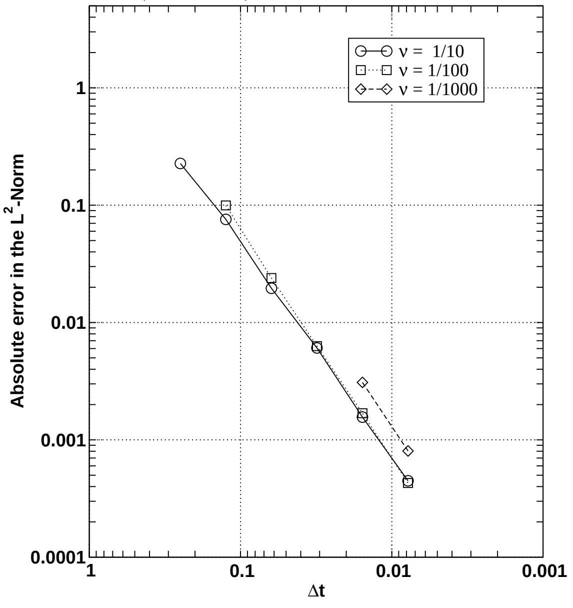

Figure 8: Absolute L^2-Error of v_1 on the test problem I (explicit)

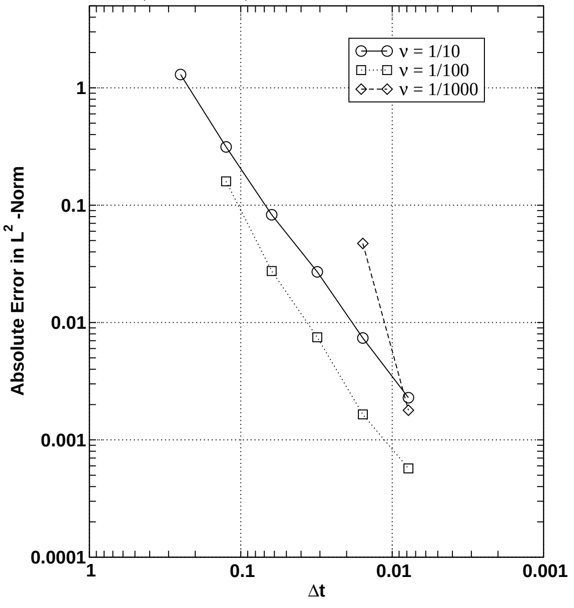

Figure 9: Absolute L^2-Error of p on the test problem I (explicit)

5 The multi-grid postprocessing

As the splitting with spectral methods [9] the base splitting is of second order in time. Now with a kind of postprocessing step there is a stable way to compute an approximation $\bar{p}^{m+1}$ at p^{m+1} of an order higher than one that can be used to compute $\tilde{v}$.

5.1 The full implicit approach

0. Compute an initial pressure p_h^0 for $t = 0$ with (10)

Time step with built-in postprocessing

(version with linear base functions and a full implicit treatment of the nonlinear term)

1. Solve the PDE (13) for the intermediate velocity $\tilde{v}_{h/2}^{n+1}$ using p_h^n
2. Solve the Poisson equation (5), (6) in V_h for the pressure and velocity update
3. Apply the update to the velocity $v_{h/2}^{n+1} = \tilde{v}_{h/2}^{n+1} + \frac{\triangle t}{\beta_0} \nabla p_{update}$
4. Solve the PDE (10) and use $\hat{v}_{h/2}^{n+1}$ on the right side to get $\bar{p}_{h/2}^{n+1}$
5. Solve the PDE (13) for the intermediate velocity $\tilde{v}_{h/4}^{n+1}$ with the initial value $v_e = P\hat{v}_{h/2}^{n+1}$ and $\bar{p}_{h/2}^{n+1}$ from step 4
6. Solve the Poisson equation (5), (6) in $V_{h/2}$ for the pressure and velocity update:

$$-\nabla^2 p_{h/2_{update}} = -\frac{\beta_0}{\triangle t} \nabla \cdot \tilde{v}_{h/4}^{n+1} \; ; \; p_{h/2_{update}} = 0 \text{ on } \partial\Omega$$

7. Apply the update to the velocity and the pressure

$$\begin{aligned} p_{h/2}^{n+1} &= \bar{p}_{h/2}^{n+1} + p_{h/2_{update}} \\ v_{h/4}^{n+1} &= \tilde{v}_{h/4}^{n+1} + \frac{\triangle t}{\beta_0} \nabla p_{h/2_{update}} \end{aligned}$$

8. Compute the restrictions for the next splitting step:

$$v_{h/2}^{n+1} = I_{h/2}\, v_{h/4}^{n+1} \; , \; p_h^{n+1} = I_h\, p_{h/2}^{n+1}$$

There we use a set of nested finite element spaces. Let $V_{h/2}$ be a finite element space that was built by a global regular refinement of the mesh of V_h. V_H is such a finite element space that V_h together with V_H satisfy the inf-sup-condition, e.g. quadratic base function of the same mesh or again a global refinement of V_h. Denote now $X_h = V_{h/2} \times V_{h/2}$ and $X_H = V_H \times V_H$. Set $V_{h,0}$ and $V_{h/2,0}$ as the subspace with the elements that satisfy $\int_\Omega u \, dx = 0$.

First we compute $(v_{h/2}^{n+1}, p_h^{n+1})$ in $W_H = X_h \times V_{h,0}$ and use the results to perform a splitting step in $W_H = X_H \times V_{h/2,0}$. With this technique the number of Picard iterations in W_H can generally be reduced and the intermediate velocity can be computed with a pressure approximation of a higher order than in the base splitting. The following algorithm is an example with linear base functions, so we set $H = h/4$, and with a full implicit treatment of the nonlinear term, in the sense of implicit in both substeps. Again the prolongation between the finite element spaces is done with the common prolongation and restriction from multi-grid solvers. Only in step 8 the interpolation operator is used.

In our approach the finite element spaces $V_{h,0} \subset V_h \subset V_{h/2} \subset V_{h/4}$ are nested in every part of the algorithm and the inf-sup-condition is fulfilled. Another advantage of this procedure is that many tasks concerning adaptivity, especially adaptivity in time, can be answered in the coarser finite element spaces. This helps economising CPU costs.

5.2 The semi-implicit approach

A cheaper but less robust variation of this algorithm uses quadratic Taylor-Hood-Elements with an explicit treatment of the nonlinear term in the postprocessing substep. This leads to a positive definite and symmetric Galerkin matrix for which very effective techniques like PCG and Multigrid-techniques can be used. For a constant time step size this matrix will also be constant during the whole simulation which offers again the opportunity to use techniques like the UMFPACK decomposition. The stabilty will be further discussed in section 6.1.

Time step with built-in postprocessing

(version with linear/quadratic base functions
and implicit/explicit treatment of the nonlinear term)

5. Solve the PDE (12) for the intermediate velocity $\tilde{v}_H^{n+1}$ with the initial value $v_e = P_H \hat{v}_{h/2}^{n+1}$ and $\bar{p}_{h/2}^{n+1}$ from step 4

6. Solve the Poisson equation (5), (6) in $V_{h/2}$ for the pressure and velocity update:

$$-\nabla^2 p_{h/2_{update}} = -\frac{\beta_0}{\triangle t} \nabla \cdot \tilde{v}_H^{n+1} \; ; \; p_{h/2_{update}} = 0 \text{ on } \partial\Omega$$

8. Compute the restrictions for the next splitting step:

$$v_{h/2}^{n+1} = I_{h/2}\, v_H^{n+1} \; , \; p_h^{n+1} = I_h\, p_{h/2}^{n+1}$$

Other variations based on this idea are a predictor-corrector technique with an explicit base splitting and an implicit postprocessing or a full implicit aproach.

5.3 Motivation for the multi-grid postprocessing

To motivate this technique we will start discussing how time dependent errors affect a gradient recovery technique. To do this let $v_h \in V_h$ be the finite element approximation on the function $v \in V$ with

$$\|I_h v - v_h\| \leq O(h^s) + O(\triangle t^k) \ .$$

Now, if the gradient of v_h is constructed high frequent time dependent errors are propagated depending on the meshsize h and the used recovery technique. Using the Lipschitz continuity of a gradient recovery technique G in the finite dimensional vector space it can be shown that

$$\|GI_h v - Gv_h)\| \leq C_1 O(h^l) + C_2 O(h^{s-m}) + C_3 O(\triangle t^k) \cdot O(h^{-m}) \tag{18}$$

with the order l of the built gradient and the order m with which the gradient recovery technique G propagates the error. In all numerical tests the second term is of no importance so that C_2 seems to be zero or at least very small. E.g. in the case of linear finite elements s is 2. If we simply use the gradient instead of a recovery technique we receive $l = m = 1$.

For most gradient recovery techniques $l > 1$ but also $m > 1$. For the presented gradient recovery technique $l \approx 2$ and $m \approx 2$. So of course, especially if gradient recovery techniques of high order are used, h should be chosen regarding the time step size $\triangle t$. Using combined adaptivity in time and space it is of great importance first to reduce the time dependent error and then to check for an adaptive mesh refinement. Because this is the standard process using adaptive techniques it is not a strong limitation.

Let the Navier-Stokes equations be given with homogeneous Dirichlet boundary conditions. Beyond this let us assume that the Neumann boundary conditions for the pressure are known, so that they can be exactly fulfilled, e.g.

$$n \cdot \nabla p^{n+1} = n \cdot \nabla \bar{p}^{n+1} = 0 \text{ on } \partial\Omega \ . \tag{19}$$

For simplicity we set $\mathcal{E} = C_1 O(h^l) + C_2 O(h^{2-m}) + C_3 O(\triangle t^2) \cdot O(h^{-m})$. Let now p^{n+1} be the exact pressure at t^{n+1}. So p^{n+1} fulfills the following equation:

$$-\nabla^2 p^{n+1} = \underbrace{-(f_{1x}^{n+1} + f_{2y}^{n+1}) + v_{1x}^{n+1} v_{1x}^{n+1} + 2 v_{2x}^{n+1} v_{1y}^{n+1} + v_{2y}^{n+1} v_{2y}^{n+1}}_{g_1} \ , \tag{20}$$

$$n \cdot \nabla p^{n+1} = 0 \text{ on } \partial\Omega \ , \tag{21}$$

while $\bar{p}_h^{n+1}$ is computed to fulfill the equation

$$-\nabla^2 \bar{p}_h^{n+1} = \underbrace{-(f_{1x}^{n+1} + f_{2y}^{n+1}) + v_{H1x}^{n+1} v_{H1x}^{n+1} + 2 v_{H2x}^{n+1} v_{H1y}^{n+1} + v_{H2y}^{n+1} v_{H2y}^{n+1}}_{g_2} \ , \tag{22}$$

$$n \cdot \bar{p}^{n+1} = 0 \text{ on } \partial\Omega \ . \tag{23}$$

To achieve a conclusion about the approximation quality of $\bar{p}_h^{n+1}$ we compare the right sides of (20) and (22). The derivations on the right side in (22) can be built with a kind of gradient recovery technique to achieve a higher order with the effects discussed above. If we assume that the domain Ω is such that this Poisson equation is H^2-regular we gain

$$\|p^{n+1} - p_H^{n+1}\|_0 \leq \|p^{n+1} - p_H^{n+1}\|_2 \leq c\|g_1 - g_2\|_0 \tag{24}$$

So we need an estimate for $\|g_1 - g_2\|_0$:

$$\begin{aligned}
\|g_1 - g_2\|_0 &= \|(v_{1x}^{n+1}v_{1x}^{n+1} - v_{H1x}^{n+1}v_{H1x}^{n+1}) + (v_{2y}^{n+1}v_{2y}^{n+1} - v_{H2y}^{n+1}v_{H2y}^{n+1}) \\
&+ 2(v_{2x}^{n+1}v_{1y}^{n+1} - v_{H2x}^{n+1}v_{H1y}^{n+1})\|_0 \\
&\leq \underbrace{\|v_{1x}^{n+1}v_{1x}^{n+1} - v_{H1x}^{n+1}v_{H1x}^{n+1}\|_0}_{I.} + \underbrace{\|v_{2y}^{n+1}v_{2y}^{n+1} - v_{H2y}^{n+1}v_{H2y}^{n+1}\|_0}_{II.} \\
&+ \underbrace{2\|v_{2x}^{n+1}v_{1y}^{n+1} - v_{H2x}^{n+1}v_{H1y}^{n+1}\|_0}_{III.}
\end{aligned}$$

With the reverse triangle inequality and (18) we achieve:

$$\|v_{H1x}^{n+1}\|_0 - \|v_{1x}^{n+1}\|_0 \leq \|v_{1x}^{n+1} - v_{H1x}^{n+1}\|_0 \leq \mathcal{E} \Leftrightarrow \|v_{H1x}^{n+1}\|_0 \leq \|v_{1x}^{n+1}\|_0 + \mathcal{E} \tag{25}$$

We start with the estimation for $I.$:

$$\begin{aligned}
&\|v_{1x}^{n+1}v_{1x}^{n+1} - v_{H1x}^{n+1}v_{H1x}^{n+1}\|_0 = \|(v_{1x}^{n+1} - v_{H1x}^{n+1})(v_{1x}^{n+1} + v_{H1x}^{n+1})\|_0 \\
\leq\ &\|v_{1x}^{n+1} - v_{H1x}^{n+1}\|_0\|v_{1x}^{n+1} + v_{H1x}^{n+1}\|_0 \\
\underbrace{\leq}_{(18)}\ &\mathcal{E} \cdot \|v_{1x}^{n+1} + v_{H1x}^{n+1}\|_0 \underbrace{\leq}_{(25)} \mathcal{E} \cdot (\underbrace{2\|v_{1x}^{n+1}\|_0}_{=C_{1V}} + \mathcal{E}) \leq \mathcal{E}(C_{1V} + \mathcal{E})
\end{aligned}$$

The estimation for $II.$ can be done analogous with $\|v_{H2y}^{n+1}\|_0 \leq \|v_{2y}^{n+1}\|_0 + \mathcal{E}$ and $C_{2V} = 2\|v_{2y}^{n+1}\|_0$. Thus with $C_{3V} = \max\{\|v_{1y}^{n+1}\|_0, 2\|v_{2x}^{n+1}\|_0\}$ it follows:

$$\begin{aligned}
\|v_{2x}^{n+1}v_{1y}^{n+1} - v_{H2x}^{n+1}v_{H1y}^{n+1}\|_0 &= \|v_{2x}^{n+1}v_{1y}^{n+1} - v_{H2x}^{n+1}v_{1y}^{n+1} + v_{H2x}^{n+1}v_{1y}^{n+1} - v_{H2x}^{n+1}v_{H1y}^{n+1}\|_0 \\
&\leq \|v_{2x}^{n+1}v_{1y}^{n+1} - v_{H2x}^{n+1}v_{1y}^{n+1}\|_0 + \|v_{H2x}^{n+1}v_{1y}^{n+1} - v_{H2x}^{n+1}v_{H1y}^{n+1}\|_0 \\
&\leq \|v_{1y}^{n+1}\|_0\|v_{2x}^{n+1} - v_{H2x}^{n+1}\|_0 + \underbrace{\|v_{H2x}^{n+1}\|_0}_{\leq 2\|v_{2x}^{n+1}\|_0 + \mathcal{E}} \|v_{1y}^{n+1} - v_{H1y}^{n+1}\|_0 \\
&\leq \|v_{1y}^{n+1}\|_0\mathcal{E} + \|v_{2x}^{n+1}\|_0\mathcal{E} + \mathcal{E}^2 = \mathcal{E}(C_{3V} + \mathcal{E})
\end{aligned}$$

With a constant $C_V = 3\max\{C_{1V}, C_{2V}, C_{3V}\}$ that only depends on the first order derivations of the exact solution v we get under the above conditions

$$\|\nabla^2(p^{n+1} - p_H^{n+1})\|_0 \leq \mathcal{E}(C_{1V} + C_{2V} + C_{3V} + 3\mathcal{E}) = \mathcal{E}(C_V + 3\mathcal{E}) \tag{26}$$

Summarizing we derive that for a constant mesh the time dependent error will be reduced of second order. The following numerical results substantiate that with this pressure approximation it is possible to achieve reduction rates of higher order in time.

6 Numerical results for the splitting with built-in postprocessing

Again we test the splitting on the test problems **I** and **II** to display the benefits of the postprocessing technique. Because the semi-implicit and the full implicit approach differ much in their scope and behaviour they will be discussed in different subsections.
The Picard iterations per time step (PIPS) are given in the different tables are the sum of the Picard iterations in the base splitting and the postprocessing step.

6.1 Results for the full implicit approach

At first glance the algorithm of the splitting with built-in postprocessing seems to be more expensive than the one without. But as table 1 shows the splitting technique with post-processing is with the same number of unknowns in all numerical tests faster than the one without. The reason is the lower number of Picard iterations per time step (PIPS) in W_H. Beyond this the accuracy is highly increased, e.g. as we can see from table 1 for $\triangle t = 1/32$

Δt	Degrees of freedom	with Postprocessing $\|u-u_h\|_{L^2}$	Quot.	without Postprocessing $\|u-u_h\|_{L^2}$	Quot.	Speed-up
		velocity (v_1)				
1/8	29408	1.216e-01	-	1.222e-01	-	1.34
1/16	29408	1.768e-02	6.880	4.035e-02	3.029	1.12
1/32	116672	2.254e-03	7.843	6.779e-03	5.952	1.07
1/64	116672	3.026e-04	7.448	2.247e-03	3.018	1.35
1/64	464768	2.811e-04	8.018	-	-	-
		pressure (p)				
1/8	7472	1.907e-02	-	5.087e-01	-	1.34
1/16	7472	2.827e-03	6.746	1.145e-01	4.443	1.12
1/32	29408	4.260e-04	6.636	2.735e-02	4.187	1.07
1/64	29408	3.348e-04	1.273	8.960e-03	3.052	1.35
1/64	116672	1.008e-04	4.226	-	-	-

Table 1: Splitting with and without postprocessing by comparison; test problem II ; $\nu = 1/5000$

the resulting velocity error is more than thrice higher than with the base splitting, for the pressure it is about a factor 64.

In table 1 we also observe the effect described in section 4. The the measuring L_2-error for $\Delta t \to 0$ converges against the error in space. For the $\Delta t = 1/64$ we can see that with a number of unknowns comparable to the base splitting a further reduction in p is impossible. Because of the coarser mesh for the pressure, p is the first value influenced by this effect. With another refinement it is possible to achieve a reduction rate beyond second order in p again. To achieve proper reduction rates for $\Delta t = 1/128$ two more global refinements were needed and because of this it is left out. For a higher accuracy elements with a higher order in space need to be used. For the test problem I in table 2 and 3 we observe the same behaviour. So

Δt	Degrees of freedom (DoF)	PIPS	v_1 $\|f-f_h\|_0$	Quotient	v_2 $\|f-f_h\|_0$	Quotient
1/8	16641	12.7	6.284e-02	-	7.320e-02	-
1/16	66049	6.6	5.516e-03	10.139	6.930e-03	10.056
1/32	66049	4.9	6.066e-04	9.093	6.554e-04	10.057
1/64	263169	4.3	1.107e-04	5.480	1.215e-04	5.392
1/128	263169	4.0	2.413e-05	4.588	2.415e-05	5.033

Table 2: Test Problem I; The L_2 error in v for Re = 2000 and a full implicit splitting with postprocessing with different $\triangle t$-h-combinations

Δt	DoF	PIPS	$\|f-f_h\|_0$	Quotient
1/8	4225	12.7	2.206e-01	-
1/16	16641	6.6	2.362e-02	9.341
1/32	16641	4.9	6.951e-03	3.398
1/64	66049	4.3	1.835e-03	3.787
1/128	66049	4.0	4.774e-04	3.845

Table 3: Test Problem I; The L_2 error in p for Re = 2000 and a full implicit splitting with postprocessing with different $\triangle t$-h-combinations

on testproblem I and II for a sufficiently small error in space we can observe reduction rates beyond the second order.

6.2 Results for the semi-implicit approach

Because of the explicit treatment of the nonlinear term this technique is less stable than the full implicite approach. In the numerical tests it turns out that it is limited to a Reynolds number less that 2000.

Beyond this as table 4 shows the approach has to consider a kind of CFL condition. On a grid that is too fine for a given time step size the method diverges.

Δt	DoF	$\|f-f_h\|_{L^2}$	DoF	$\|f-f_h\|_{L^2}$
1/32	7472/29408	4.071e-03	29408/116672	div.
1/64	7472/29408	9.901e-04	29408/116672	5.148e-04
1/128	7472/29408	7.322e-04	29408/116672	2.226e-04

Table 4: CFL condition; L_2-Error in v_1 ; test problem 2; $\nu = 1/5000 \Leftrightarrow Re \approx 1925$

In table 5 the results on test problem I are displayed. On the one hand with the semi-implicit approach it is possible to compute very good results with an explicit and cheap technique. On the other hand we see that the ratio of mesh size and Δt influence the results. After every mesh refinment the reduction quotient is reduced.

Even with if handicaps compared to the very robust full implicite aproach the semi-implicite aproach is intessting for problems with low Reynolds numbers, because the low

Δt	DoF in the base/postprocessing-step	PIPS	$\|u-u_h\|_{L^2}$	Quotient	$\|u-u_h\|_{\max}$
			v_1		
1/4	1089/4225	7.8	1.389e-01	-	5.210e-01
1/8	1089/4225	5.8	1.815e-02	7.655	7.262e-02
1/16	4225/16641	4.9	6.287e-03	2.886	2.486e-02
1/32	4225/16641	4.4	8.757e-04	7.180	3.743e-03
1/64	16641/66049	4.0	3.019e-04	2.901	1.544e-03
1/128	16641/66049	4.0	5.240e-05	5.761	2.334e-04
			v_2		
1/4	1089/4225	7.8	1.360e-01	-	6.167e-01
1/8	1089/4225	5.8	2.094e-02	6.495	6.530e-02
1/16	4225/16641	4.9	6.378e-03	3.283	2.465e-02
1/32	4225/16641	4.4	8.840e-04	7.215	3.705e-03
1/64	16641/66049	4.0	3.036e-04	2.911	1.540e-03
1/128	16641/66049	4.0	5.219e-05	5.817	2.323e-04
			p		
1/4	289/1089	7.8	5.741e-01	-	1.788
1/8	289/1089	5.8	7.465e-02	7.691	2.392e-01
1/16	1089/4225	4.9	2.771e-02	2.694	1.394e-01
1/32	1089/4225	4.4	4.109e-03	6.743	2.776e-02
1/64	4225/16641	4.0	1.875e-03	2.192	1.757e-02
1/128	4225/16641	4.0	2.716e-04	6.904	7.304e-03

Table 5: Influence of the mesh refinement on the half implicit/explicit splitting on test problem I with $Re = 10.0$; all partial derivations on the right side, including the non-linear term, were constructed with the gradient recovery technique

CPU costs. For the quite smooth artificial test problems the upper boundary was a Reynolds number of 2000. On the Standard CFD Problems in section 6.3 it was able to solve the 'Driven Cavity'-problem up to a Reynolds numbers of 1000. It was impossible to achieve proper results on the 'Flow around a cylinder'-problem.

6.3 Standard CFD Problems

To test the splitting on some standard CFD problems we use the 'Flow around a cylinder'- and the 'Driven Cavity'-problem.

6.3.1 'Driven Cavity'

The goal in this very common benchmark problem is to compute the flow of a 2D driven cavity at various Reynolds numbers. The domain itself is the unit square. As one can see from figure 10 the cavity is driven by a translating plate at the top of the cavity given by $\bar{v}_1$. Choosing $\bar{v}_1 = -1$ we obtain the (unregularized) Driven Cavity Problem. The boundary conditions of the Driven Cavity Problem induce singularities at the top corners of the unit square, so the mesh in this region was a priori refined as displyed in figure 11. Depending

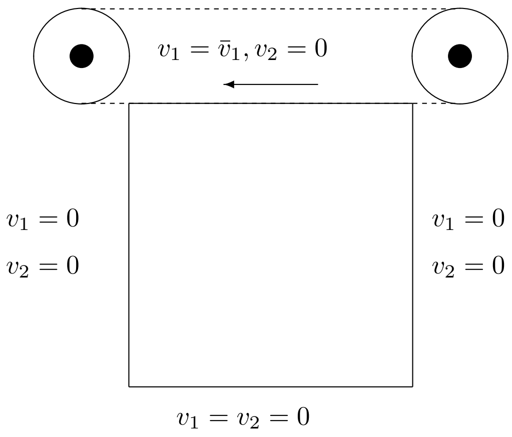

Figure 10: The Driven Cavity Problem

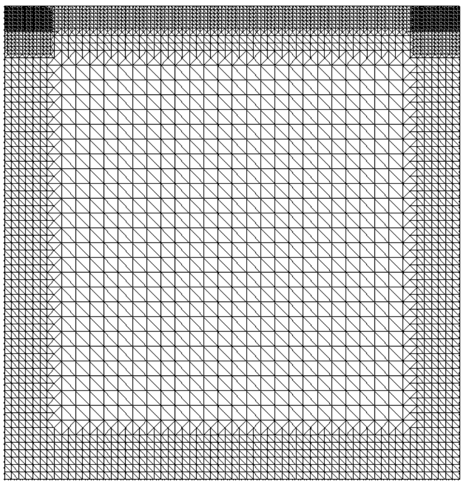

Figure 11: Mesh for the Driven Cavity Problem

max $\|\psi\|$ (Streamfunction)		
	Botella & R. Peyret (1998)	adaptive mesh refinement; half implicit
x-coordinate	0.4692	0.46875
y-coordinate	0.5652	0.5625
max $\|\psi\|$	0.1189366	0.122

Table 6: Results of the implicit splitting with explicit postprocessing using quadratic Taylor-Hood-Elements to solve the Driven Cavity Problem with $Re = 1000$

on the Reynolds number the Driven Cavity Problem converged against a stationary solution, which properties are used as benchmark values. At this benchmark problem we also tested the variation of the splitting algorithm which uses an explicit treatment of the nonlinear term with quadratic Taylor-Hood-Elements. The explicit treatment leads to a symmetric, positive definite problem which is much easier to be solved. On the other hand it is less robust than the full implicit one and so it is restricted to a Reynolds number less than 2000 for the Driven Cavity Problem. Table 6 shows the result achieved with this technique on an adaptive refined mesh started with 58153 Degrees of Freedom for v and 14725 for p at the postprocessing mesh. The final mesh has 323248 Degrees of Freedom for v and 81187 for p. As error indicator it is possible to use the described gradient recovery technique similar to the Z^2 technique described in [24]. A mesh refinement in combination with a splitting scheme that uses the fact that $\nabla \cdot v = 0$ possibly has disadvantages. The reason is that added unknowns need interpolated data for the former time steps. This interpolated data may disturb the divergence-freedom of v which in the step 6 of the algorithm influences mainly the pressure for a small time step size because of the factor $1/\triangle t$. This is a quite common phenomenon for splitting techniques, see e.g. [21], p.21ff. The velocity is not affected because the intermediate velocity is computed using the pressure constructed in step 1 in which no factor amplifies this effect. So for the Driven Cavity this is not of real importance because mainly we are interested in the velocity.

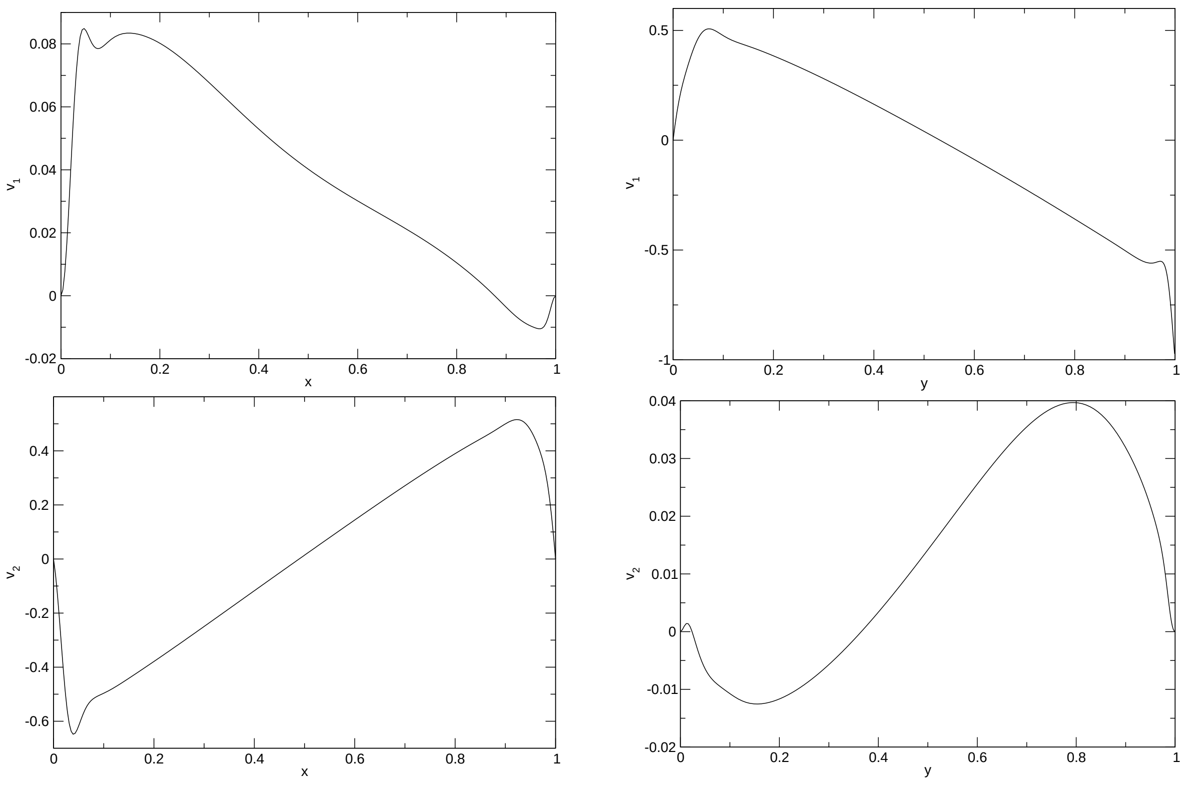

Figure 12: Results of the full implicit splitting with postprocessing for Driven Cavity Problem with $Re = 5000$ and 58153 (pre)/ 231121 (post) degrees of freedom in v ; the velocity components v_1, v_2 along $x = 0.5$ and $y = 0.5$

One benchmark value for the Driven Cavity Problem is the maximum of the Streamfunction ψ which is defined by

$$\begin{aligned} -\nabla^2\psi &= \frac{\partial u_2}{\partial x} - \frac{\partial u_1}{\partial y} \quad \text{in } \Omega \\ \psi &= 0 \quad \text{on } \partial\Omega \, . \end{aligned}$$

In table 6 it is compared with the value computed by Botella & R. Peyret using spectral methods on a problem with substrated singularities so that only the smooth part of the solution has to be computed. The result of this procedure is a very accurate solution.

To compute the stationary solution for higher Reynolds numbers we used the full implicit splitting with postprocessing on a constant mesh. The values of the velocity along $x = 0.5$ and $y = 0.5$ are displayed in figure 12 and are similar to the ones published in [8].

6.3.2 'Flow around a cylinder'

A very popular benchmark problem is the 'Flow around a cylinder' defined by Schäfer and Turek within the DFG high priority program *flow simulation with high-performance computers* in [19]. Three variations of this problem exist, the geometry is displayed in figure 13. We will present results on two of them. In [19] the inflow at Γ_2 is given in the definition. For the

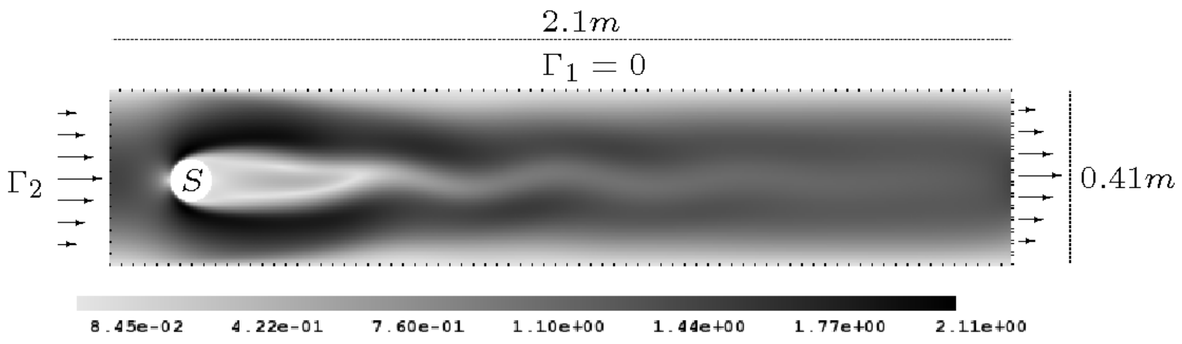

Figure 13: The geometry of the 'Flow around a cylinder' with the velocity after 4 seconds in the 2D-3 case

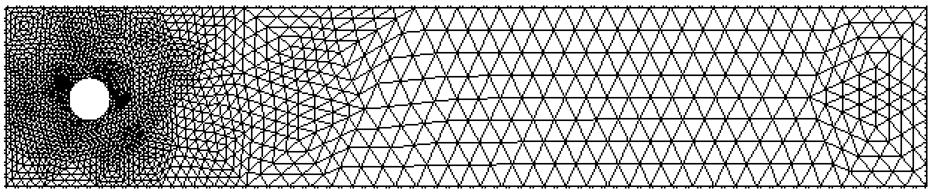

Figure 14: The mesh for the 'Flow around a cylinder', a priori refined around the obstacle

outflow Γ_3 we used like John in [12] the same boundary conditions as for the inflow. At the other boundaries no-slip conditions are given. The kinematic viscosity ν is $1/1000$ and the diameter of S is $0.1m$. Since the problem is two-dimensional, it is well known that a weak solution $(v;p)$ exists and this solution is unique.

The benchmark values are the drag and the lift coefficient and the difference of the pressure at two points at the edge of the obstacle. So for all of them we need a high accuracy at $\partial\Omega$ in v and p which underlines again the importance of the introduced fitted boundary conditions for the pressure in section 2.

The most difficult benchmark value for all algorithms in [19] is the lift coefficient. It takes small time steps and a lot of unknowns to get proper results. The drag coefficient and the pressure are easier to compute. To compute the drag (c_d) and the lift (c_l) coefficient we used an approximation first published for the stationary Navier-Stokes equations in [11]. Applying it to the unstationary Navier-Stokes equations leads to the following equations:

$$c_d = -20\int_\Omega \frac{\partial}{\partial t}v\cdot u_d + \nu\nabla v : \nabla u_d + (v\cdot\nabla)v\cdot u_d - p(\nabla\cdot u_d)\, d\Omega \tag{27}$$

$$c_l = -20\int_\Omega \frac{\partial}{\partial t}v\cdot u_l + \nu\nabla v : \nabla u_l + (v\cdot\nabla)v\cdot u_l - p(\nabla\cdot u_l)\, d\Omega\ . \tag{28}$$

Alternative ways to compute c_d and c_l can e.g. be found in [21].

6.3.3 Type 2D-3 (unsteady)

For the 2D-3 variation the velocity is simulated over 8 seconds. For the in- and outflow we obtained:

$$v(2.2,y,t) = v(0,y,t) = 0.41^{-2}\sin(\pi t/8)(1.2y(0.41-y),0)\ , \quad 0\le y\le 0.41 \tag{29}$$

Table 7: 'Flow around a cylinder' with postprocessing

$\triangle t$	$t(c_{d,\max})$	$c_{d.\max}$	$t(c_{l,\max})$	$c_{l,\max}$	$p_{\text{diff}}(8s)$
1/400	3.93	2.9509076	5.695	0.49461359	-0.11086049
1/1000	3.934	2.9478232	5.688	0.49117886	-0.11053843
1/1200	3.93	2.9465880	5.686667	0.49084030	-0.11048193
John:04	3.93625	2.9509216	5.6925	0.47811979	-0.11158097

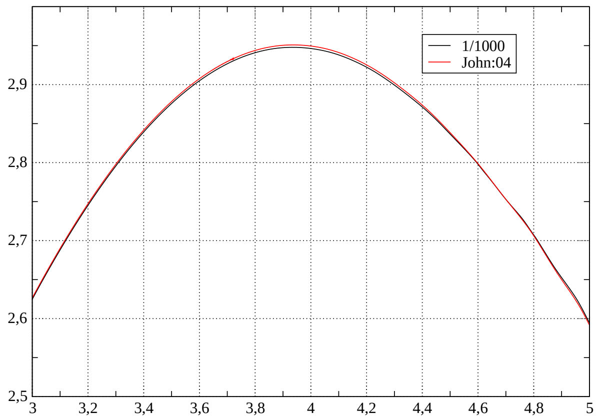

Figure 15: 2D-3 : c_d

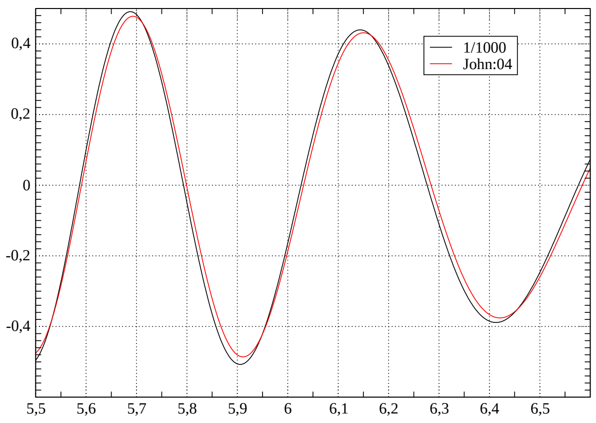

Figure 16: 2D-3 : c_l

So after a while the inflow increases and two vortices start to develop behind the cylinder. Figure 13 shows this at $t = 4$. For $t \in [4, 5]$ the vortices separate from the cylinder and a vortex street develops. The vortices are still visible at $t = 8$. Figures 15 and 16 show the results with 139344 unknowns for the velocity and 35048 for the pressure compared to the results computed by John in [12] with quadratic Taylor-Hood-Elements and 399616 unknowns in v and 50240 in p. John used a fractional-step-θ-scheme with a macro step size of 1/800, which means a substep size of 1/2400, see [21], p.162 for details. The intervals for the benchmark values defined in [19] are $p_{\text{diff}}(8s) \in [-0.115, -0.105]$ $c^{\text{ref}}_{d,\max} = [2.93, 2.97]$ and $c^{\text{ref}}_{l,\max} = [0.47, 0.49]$. Table 7 shows the good results which could be computed with a quite low number of unknowns and a time step size up to $\triangle t = 1/1200$.

6.3.4 Type 2D-2 (periodic, unsteady)

For the type 2D-2 we use the splitting algorithm with the presented full implicit postprocessing and 555680 unknowns in v and 139344 in p. The variation 2D-2 from [19] usually needs small time step sizes over the whole simulation. It has as inflow condition

$$v(0, y, t) = 0.41^{-2}(6y(0.41 - y), 0)\ , \quad 0 \le y \le 0.41 \tag{30}$$

and we choose again $v(2.2, y, t) = v(0, y, t)$. Not depending on the start configuration $(v; p)$ at $t = 0$ these conditions create a periodic behaviour of the solution. Beyond the drag and the lift coefficient we are now interested in the Strouhal number given by

$$St = \frac{fd}{\bar{V}}\ .$$

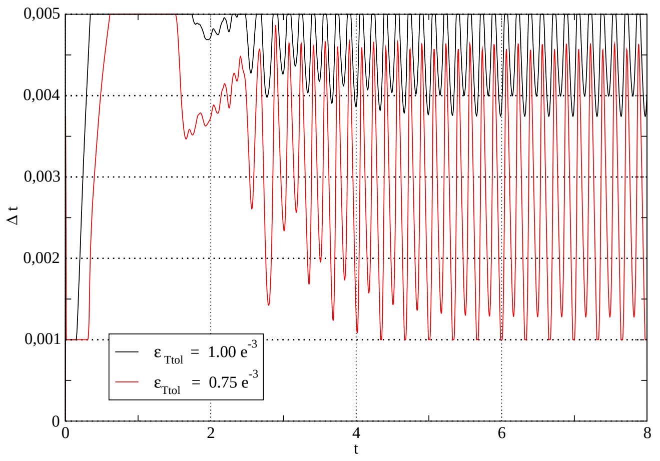

Figure 17: 2D-2 : $\triangle t$ chosen with two different parameters ε_{Ttol} and a secure restriction $0.001 < \triangle t < 0.005$

While f is the frequency of vortex shedding, d is the characteristic length and $\bar{V}$ is in this case $\bar{V} := 2/3 \cdot v(0, 0.41/2, t)$. Hence we derive with the period $\mathcal{P} = 1/f$

$$St = 0.1 \cdot f = \frac{1}{10\mathcal{P}} \ .$$

ε_{Ttol}	$\bar{\triangle t}$	$c_{d.\,\max}$	$c_{l,\max}$	Strouhal
$1.0 \cdot 10^{-3}$	0.004246	3.2439	1.0104	0.29811
$7.5 \cdot 10^{-4}$	0.002479	3.2285	1.0031	0.30022

Table 8: 2D-2: Results for an adaptive chosen time step size

The error indicator

$$e_t(t^m) \approx \frac{4\|v_{\frac{\triangle t_m}{2}}(t^m) - v_{\triangle t^m}(t^m)\|}{3\|v_{\frac{\triangle t_m}{2}}(t^m)\|} \qquad \triangle t_{m+1} = \sqrt{\frac{\varepsilon_{Ttol}}{e_t(t^m)}}\triangle t_m$$

is based on the approximation in the coarser space $v_{h/2}$ to minimize the CPU costs. We computed $v_{h/2}$ with a time step size $\triangle t$ and $\triangle t/2$ was used to choose a proper time step size in consideration of the stability of the BDF scheme, see e.g. [4]. The benchmark values displayed in table 8 are computed with the choice of time step sizes displayed in figure 17 which leads to an average time step size $\bar{\triangle t}$ shown in table 8. ε_{Ttol} is a parameter for the time error tolerance for the choice of the time step size. The tolerance intervals for c_d are [3.2200,3.2400], for c_l [0.9900,1.0100] and for the Strouhal number [0.2950, 0.3050].

7 Conclusions

The presented algorithm with built-in postprocessing shows an error reduction in the L^2 norm of an order greater than two in time. It was successfully tested on analytic problems as well as on standard CFD problems. A very interesting aspect of the postprocessing with nested grids is that in all numerical experiments it caused no additional CPU costs compared to the base splitting. The algorithm can be supplemented with adaptive control methods. The control of the time step size can be implemented in the first substep of the splitting with postprocessing to reduce CPU costs. An extension of the techniques to three-dimensional problems could be done straightforward, because neither the in chapter 4 presented base splitting nor the postprocessing technique makes any use of the number of dimensions. The gradient recovery technique is as well not especially designed for the two dimensional case. The definition of the neighbourhood of a has to be extended to transfer this method to a higher dimensional case. Nevertheless, it is possible that the bigger neighbourhood in three dimensions will make the gradient recovery technique less attractive so that a fallback to Z^2-like techniques has to be performed. A port of this technique to three dimensions is one of the future prospects. Beyond this further future prospects could be e.g. the integration of more levels together with the fourth order BDF scheme for the postprocessing and the use of higher order finite elements.

Acknowledgment: We would like to thank Volker John for providing his benchmark data for the 'Flow around a cylinder' 2D-3 case.

References

[1] O. Botella and R. Peyret: Benchmark spectral results on the lid-driven cavity flow, Computers & Fluids , **27** , 421–433 (1998)

[2] Susanne C. Brenner and L. Ridgway Scott: The Mathematical Theory of Finite Element Methods, Springer New York (1994)

[3] T. A. Davis: UMFPACK - an unsymmetric-pattern multifrontal method with a column pre-ordering strategy, ACM Trans. Math. Software, Vol. 30, **2**, 196–199 , (2004)

[4] R. D. Grigorieff: Stability of Multistep-Methods on Variable Grids. Numerische Mathematik, **42**, 359–377 (1983)

[5] M. J. Grote and T. Huckle: Parallel Preconditioning with Sparse Approximate Inverse, SIAM J. Sci. Computing **18**, 838 – 853 (1997)

[6] J. Frochte: Ein Splitting-Algorithmus höherer Ordnung für die Navier-Stokes-Gleichung auf der Basis der Finite-Element-Methode [Diss./Phd-Thesis], Universität Duisburg-Essen (2005)

[7] J. Frochte and W. Heinrichs: An adaptive operator splitting of higher order for the Navier-Stokes equations, Numerical mathematics and advanced applications. Proceedings of ENUMATH 2005, p.871-879, Springer Berlin (2006)

[8] U. Ghia and K. Ghia and C.T. Shin: High-Re solutions of Navier-Stokes equations and a multi-grid method, Journal of Comp. Phys., **48** , 387–411 (1982)

[9] H. Haschke and W. Heinrichs: Splitting techniques with staggered grids for the Navier-Stokes equations in the 2d case, Journal of Comp. Phys., **168**, 131–154 (2001)

[10] W. Heinrichs: Splitting techniques for the unsteady Stokes equations, SIAM J. Numer. Anal., **35**, 1646–1662 (1998)

[11] V. John and G. Matthies: Higher order Finite Element discretizations in a benchmark problem for the 3D Navier Stokes equations, Internation Journal for Numerical Methods in Fluid Mechanics, **40**, 775–798 (2002)

[12] V. John: Reference values for drag and lift of a two-dimensional time dependent flow around a cylinder, Internation Journal for Numerical Methods in Fluids, **44**, 777–788 (2004)

[13] G. E. Karniadakis and Moshe Israeli and S. A. Orszag: High-order splitting methods for the incompressible Navier-Stokes equations. J. Comput. Phys. 97, No.2, 414-443 (1991).

[14] P. Knabner and L. Angermann: Numerik partieller Differentialgleichungen", Springer Berlin (2000)

[15] G.I. Marchuk: Splitting and alternating direction methods. Ciarlet, P. G. (ed.) et al. Handbook of numerical analysis. Volume I: Finite difference methods (Part 1), solution of equations in R^n (Part 1). Amsterdam: North-Holland. 197-462 (1990)

[16] Y. Maday and A. T. Patera and E. M. Ronquist: An operator-integration-factor splitting method for time-dependent problems: Application to incompressible fluid flow. J. Sci. Comput. 5, No.4, 263-292 (1990)

[17] P.D. Minev and F.N. van de Vosse and L.J.P Timmermans and A.A. van Steenhoven: A second-order splitting algorithm for thermally-driven flow problems. Int. J. Numer. Methods Heat Fluid Flow 6, No.2, 51-60 (1996)

[18] H.-G. Roos and M. Stynes and L. Tobiska: Numerical Methods for Singularly Perturbed Differential Equations, Springer Berlin (1996)

[19] M. Schäfer and S. Turek: The benchmark problem 'flow around a cylinder'. In: Hirchel EH (ed.) Notes on Numerical Fluid Mechanics. Vieweg Verlag Braunschweig, (1996)

[20] L.J.P. Timmermans and P.D. Minev and F.N. van de Vosse: An approximate projection scheme for incompressible flow using spectral elements. Int. J. Numer. Methods Fluids 22, No.7, 673-688 (1996).

[21] S. Turek: Efficient Solvers for Incompressible Flow Problems. Springer, Berlin (1999)

[22] Vidar Thomée: Galerkin Finite Element Methods for Parabolic Problems, Springer Berlin (1997)

[23] N.N.Yanenko, The method of fractional steps, Springer-Verlag. Berlin (1971) . Russian ed.: Nauka, Novosibirsk, 1967

[24] O. C. Zienkiewicz and J. Z. Zhu: The superconvergent patch recovery and a posteriori error estimates. Part I & II, Int. J. Num. Meth. Engrg., **33**, 1331–1382 (1992)