An adaptive operator splitting of higher order for the Navier-Stokes equations

Jörg Frochte[1] and Wilhelm Heinrichs[2]

[1] Arbeitsgruppe Ingeneurmathematik, Universität Duisburg-Essen Campus Essen, Universitätsstr. 3, 45117 Essen `jfrochte@ing-math.uni-essen.de`
[2] Arbeitsgruppe Ingeneurmathematik, Universität Duisburg-Essen Campus Essen, Universitätsstr. 3, 45117 Essen `wheinric@ing-math.uni-essen.de`

Summary. This article presents an operator splitting for solving the time-dependent incompressible Navier-Stokes equations with Finite Elements. By using a postprocessing step the splitting method shows a reduction factor higher than second order. 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.

1 Introduction

We consider the incompressible time dependent Navier-Stokes equations

$$\frac{\partial v}{\partial t} + (v \cdot \nabla)v - \nu \nabla^2 v + \nabla p = f \quad \text{ in } \Omega,\ t \in [0, t_{\text{end}}] \tag{1}$$

$$\nabla \cdot v = 0 \text{ in } \Omega,\ t \in [0, t_{\text{end}}] \quad ; \quad v = v_0 \text{ for } t = 0, \text{ in } \Omega \ , \tag{2}$$

$$v = h \text{ on } \partial\Omega,\ t \in [0, t_{\text{end}}] \ . \tag{3}$$

The solution of these equations on the time intervall $[0, t_{\text{end}}]$ 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 $\partial\Omega$. To solve the Navier-Stokes equations we use a splitting technique with a postprocessing. The algorithm without postprocessing, called base splitting algorithm, is related to the one published by Haschke and Heinrichs [2] for spectral methods. For linear Finite Elements in contradiction to the solution itself the convergence rate of the gradient is only of first order. To avoid this and to compute boundary conditions for the pressure which are always a challenge for splitting techniques a new gradient recovery technique is developed.

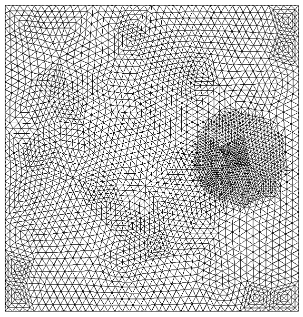

Fig. 1. Mesh with 3233 unknowns

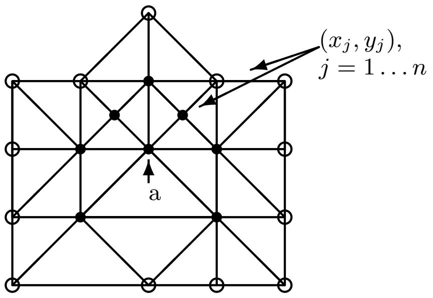

Fig. 2. Database of G_T/TBR and the Z^2 gradient recovery technique

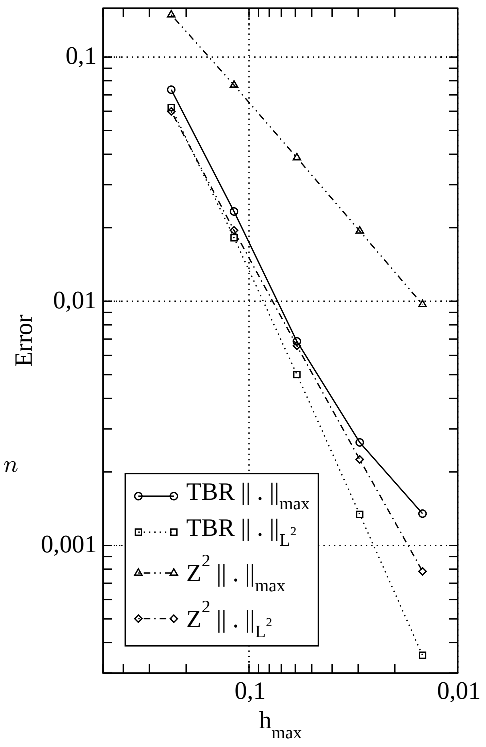

Fig. 3. Error TBR and Z^2

2 The Taylor based gradient recovery technique

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:

$$u_h(x_j, y_j) - u_h(x_a, y_a) = \mathbf{u_x}(\mathbf{x_a}, \mathbf{y_a})(x_j - x_a) + \mathbf{u_y}(\mathbf{x_a}, \mathbf{y_a})(y_j - y_a)$$
$$+\tfrac{1}{2}(\mathbf{u_{xx}}(\mathbf{x_a}, \mathbf{y_a})(x_j - x_a)^2 + \mathbf{u_{xy}}(\mathbf{x_a}, \mathbf{y_a})(x_j - x_a)(y_j - y_a) + \mathbf{u_{yy}}(\mathbf{x_a}, \mathbf{y_a})(y_j - y_a)^2)$$

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 [7] uses only the information from the nodes with filled circles. The greater database together with a proper weighting [1] improves the results, especially on adaptive refined meshes and at the edges of Ω. 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 Laplace equation, $-\nabla^2 u = f$. Figure 3 illustrates the fact that the TBR technique shows higher reduction rates in the L^2 norm. 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 $G_T u_h$.

3 The stabilized 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 [2] one time step of the splitting follows the scheme:

Time step in the base splitting

1. Compute a guess $(\bar{p}^{n+1})$ for the pressure
2. Based on the pressure compute an intermediate velocity $\tilde{v}^{n+1}$
3. Solve the Laplace equation $(**)$ $-\nabla^2 p_{update} = -\frac{\beta_0}{\triangle t}\nabla\cdot\tilde{v}^{n+1}$; $p_{update} = 0$ on $\partial\Omega$ for the pressure and velocity update
4. Apply the update by $p^{n+1} = \bar{p}^{n+1} + p_{update}$; $v^{n+1} = \tilde{v}^{n+1} + \frac{\triangle t}{\beta_0}\nabla p_{update}$

In difference to [2] $\bar{p}^{n+1}$ is the solution of the following Laplace equation:

$$-\nabla^2\bar{p} = -\nabla\cdot f + \nabla\cdot((v\cdot\nabla)v) \tag{4}$$

$$\underbrace{\Leftrightarrow}_{\nabla\cdot v=0} -\nabla^2\bar{p}^{n+1} = -(f^n_{1x} + f^n_{2y}) + v^n_{1x}v^n_{1x} + 2v^n_{2x}v^n_{1y} + v^n_{2y}v^n_{2y}\ . \tag{5}$$

All partial derivations on the right side were built with TBR. The Neumann boundary conditions are taken directly from the Navier-Stokes equations (1):

$$\nabla p = f - (\underbrace{\frac{\partial v}{\partial t} + (v\cdot\nabla)v}_{(*)} - \nu\nabla^2 v) \quad \text{on } \partial\Omega \tag{6}$$

The $(*)$ is zero for homogeneous zero Dirichlet boundary conditions. In the case that other boundary conditions are given $\frac{\partial v}{\partial t}$ is approximated with a BDF scheme of third order and the partial derivations are computed using G_T. The Laplace term is approximated by $G_T^2 v_1 = v_{1_{yy}} - v_{2_{yx}}$, $G_T^2 v_2 = v_{2_{xx}} - v_{1_{xy}}$. This formulation is more accurate than $v_{i_{xx}} + v_{i_{yy}}$ $(i = 1, 2)$. The reconstruction of second order derivations at the edges still causes more problems than the

recovery of the first order derivations. But generally the quite small kinematic viscosity ν reduces the influence of this term heavily. With this procedure it is possible to add fitted boundary conditions for the pressure to the splitting and also to prevent an unstable behaviour of the algorithm for solutions of the type $v(t,x,y,z) = z(t)g(x,y,z)$. If $\bar{p}^{n+1}$ is simply set equal to p^n as in [2] the pressure update step would bump the same mesh based errors stepwise into the approximated pressure function. For small time step sizes the factor $\frac{\beta_0}{\triangle t}$ on the right side of the Laplace Equation amplifies this effect which is prevented with the above displayed procedure.
With the coefficients of the BDF scheme $\beta_j (j = 1..3)$ we set
$\tilde{f} = f - G_T \bar{p}^{n+1} - \frac{1}{\triangle t} \sum_{j=1}^{3} \beta_j v^{m+1-j}$ and so the intermediate velocity can be computed explicitly

$$\left(-\nu \nabla^2 + \frac{\beta_0}{\triangle t} I\right) \tilde{v}_i^{m+1} = \tilde{f}^{n+1} - (v_e \cdot \nabla) v_e \qquad (7)$$

or implicitly

$$\left(-\nu \nabla^2 + \frac{\beta_0}{\triangle t} I\right) \tilde{v}_i^{m+1} + (v_e \cdot \nabla)\, \tilde{v}_i^{m+1} = \tilde{f}^{n+1} \qquad (8)$$

using a kind of Picard iteration ($v_e = \tilde{v}_i^{m+1}$) with the initial value $v_e = v^n$ and the stop criterion $\|\tilde{v}_i^{m+1} - \tilde{v}_{i-1}^{m+1}\| < \varepsilon_{Pic} = 10^{-3}$. The self-evident boundary conditions are taken from (3). The Finite Element spaces for the velocity and the pressure are chosen to fulfil the $\inf - \sup -$condition, so we used triangle Taylor-Hood-Elements with linear and in the context of the postprocessing also with quadratic base functions.

4 The multi-grid postprocessing

The main reason for most splittings not to reach an order higher than two in time is that it seems not possible to compute a stable pressure approximation $\bar{p}$ of second order to compute $\tilde{v}$. With such an approximation the analysis done by Heinrichs in [3] would advise at least for the Stokes equations to get a scheme of third order. With a postprocessing step there is a stable way to compute an approximation of an order higher than one that can be used to compute $\tilde{v}$. To do this we use a set of nested Finite Elements 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 $X_h = V_h \times V_h$ and $X_H = V_H \times V_H$. and set $V_{h,0}$ resp. $V_{h/2,0}$ 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 for the use of linear base functions, so set $H = h/4$ and a full implicit treatment of the nonlinear term. Other variations based on this idea can be found in [1].

```
0. Compute an initial pressure $p_h^0$ at for $t = 0$ with (5)

            **Time step with build-in postprocessing**

1. Solve the PDE (8) for the intermediate velocity $\tilde{v}_{h/2}^{n+1}$ using $p_h^n$
2. Solve the Laplace equation $(**)$ 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 (5) and use $\hat{v}_{h/2}^{n+1}$ on the right side to get $\bar{p}_{h/2}^{n+1}$
5. Solve the PDE (8) 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 Laplace equation $(**)$ in $V_{h/2}$ for the pressure and velocity
   update: $-\nabla^2 p_{h/2_{update}} = -\frac{\beta_0}{\triangle t} \nabla \cdot \tilde{v}_{h/4}^{n+1}$ ; $p_{h/2_{update}} = 0$ on $\partial\Omega$
7. Apply the update to the velocity and the pressure
   $p_{h/2}^{n+1} = \bar{p}_{h/2}^{n+1} + p_{h/2_{update}}$ ; $v_{h/4}^{n+1} = \tilde{v}_{h/4}^{n+1} + \frac{\triangle t}{\beta_0} \nabla p_{h/2_{update}}$
8. Compute the restrictions for the next splitting step:
   $v_{h/2}^{n+1} = I_{h/2}\, v_{h/4}^{n+1}$ , $p_h^{n+1} = I_h\, p_{h/2}^{n+1}$
```

The prolongation between the Finite Element spaces is done with the common prolongation and restriction from Multigridsolvers. Only in step 8 the interpolation operator is used. Because of the way 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 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. Adaptivity in space e. g. has been tested with the well-known Driven Cavity Problem, see [1] for further details.

5 Numerical Results

The splitting with and without postprocessing was tested on various test-problems. Exemplarily the results of the test-problem **IV** from [1] are displayed. Here $\Omega = \{(x, y) \in \mathbb{R}^2 | 1 \leq r \leq 2\}$, $r = \sqrt{x^2 + y^2}$ is a spool. The right side f is fitted so that the solution for the velocity is

$$v_1(x, y, t) = -y(0.25 - (r - 1.5)^2) \sin(2\pi t) \ ,$$
$$v_2(x, y, t) = x(0.25 - (r - 1.5)^2) \sin(2\pi t)$$

Table 1. Splitting with and without postprocessing by comparison ; $\nu = 1/5000$

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

and for the pressure $p(x,y,t) = y\sin(x)\sin(2\pi t)$. At first glance the splitting with build-in postprocessing seems to be more expensive than the one without. But as table 1 shows the splitting technique with postprocessing is with the same number of unknowns in all numerical tests faster than the one without.

5.1 'Flow around a cylinder'

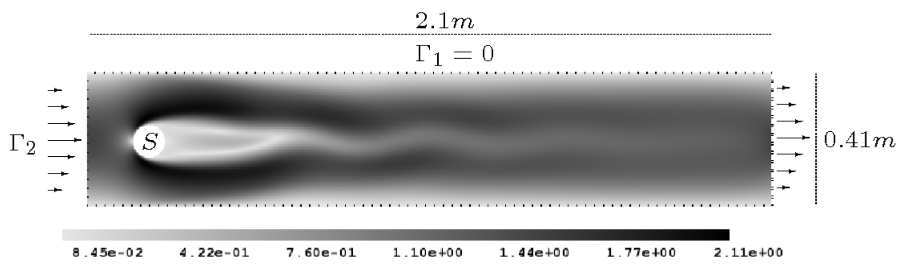

Fig. 4. 2D-3 4sec.

A very popular benchmark problem the splitting was tested with is the 'Flow around a cylinder' defined by Schäfer and Turek in [6]. For the outflow Γ_3 we used like [5] the same boundary conditions as for the inflow. To compute the drag (c_d) and the lift (c_l) coefficient we used an ansatz first published for the stationary Navier-Stokes equations in [4]. Applying it to the unstationary Navier-Stokes equations leads to the following equations:

$$c_d = -20 \int_\Omega \frac{\partial}{\partial t} v \cdot u_d + \nu \nabla v : \nabla u_d + (v \cdot \nabla) v \cdot u_d - p(\nabla \cdot u_d)\, d\Omega \qquad (9)$$

$$c_l = -20 \int_\Omega \frac{\partial}{\partial t} v \cdot u_l + \nu \nabla v : \nabla u_l + (v \cdot \nabla) v \cdot u_l - p(\nabla \cdot u_l)\, d\Omega\,. \qquad (10)$$

Type 2D-3 (unsteady)

There are different variations of the 'Flow around a cylinder'-Problem defined in [6]. For the 2D-3 the velocity is simulated over 8 seconds. The figures 5

Table 2. '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

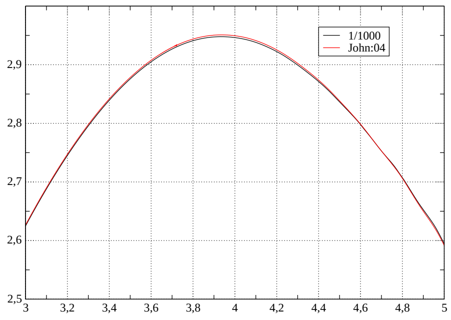

Fig. 5. 2D-3 : c_d

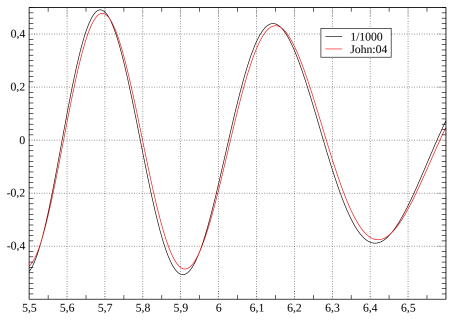

Fig. 6. 2D-3 : c_l

and 6 show the results with 139344 unknowns for the velocity and 35048 for the pressure compared to the results computed by John in [5] with quadratic Taylor-Hood-Elements and 399616 unknowns in v and 50240 in p. John used a fractional-step-θ-scheme with a step size of 1/800. The intervals for the benchmark values defined in [6] are $c^{\text{ref}}_{d,\max} = [2.93, 2.97]$ and $c^{\text{ref}}_{l,\max} = [0.47, 0.49]$. Table 2 shows the good results which could be computed with a quite low number of unknowns.

Type 2D-2 (periodic, unsteady)

The variation 2D-2 from [6] usually needs small time step sizes.

Table 3. 2D-2: Results for an adaptive chosen time step size

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

An error indicator

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

based on $v_{h/2}$ computed with $\triangle t$ and $\triangle t/2$ was used to choose a proper time step size in consideration of the stability of the BDF scheme. With this choice of time step sizes the splitting algorithm with the presented postprocessing and 555680 unknowns in v and 139344 in p computed the benchmark values displayed in table 3 with an average time step size $\bar{\triangle} t$. 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].

6 Conclusions

The presented algorithm with build-in postprocessing shows an error reduction in the L^2 norm of an order $n > 2$ 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. Therefor 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 Finite Elements of a higher order.
Acknowledgment: We would like to thank Volker John for providing his benchmark data for the 'Flow around a cylinder' 2D-3 case.

References

1. Jörg Frochte: Ein Splitting-Algorithmus höherer Ordnung für die Navier-Stokes-Gleichung auf der Basis der Finite-Element-Methode [Diss./Phd-Thesis], Universität Duisburg-Essen (to be published in Dec. 2005)
2. 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)
3. Wilhelm Heinrichs: Splitting techniques for the unsteady Stokes equations, SIAM J. Numer. Anal., **35**, 1646–1662 (1998)
4. Volker John and Gunar 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)
5. Volker 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)
6. 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)
7. 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)