INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 0000; **00**:1–13
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld

Numerical comparison of hybridized discontinuous Galerkin and finite volume methods for incompressible flow

T. Ahnert*and G. Bärwolff

Institut für Mathematik, Technische Universität Berlin, Germany

SUMMARY

A numerical comparison of a hybridizable discontinuous Galerkin method proposed by Nguyen et al. and the well established finite volume method of second order in space represented by the icoFoam and simpleFoam solver of OpenFOAM is given. The hybridizable discontinuous Galerkin method has been reformulated as Picard iteration, hybridized and implemented from scratch. The methods are introduced and four numerical standard simulations are used in order to benchmark and evaluate the solver - the Taylor-Green vortex, the 180 degrees fence case as well as a two-dimensional stationary and non-stationary DFG benchmark. The numerical examples suggests hybridized discontinuous Galerkin methods are a competitive alternative to finite volume solvers for incompressible fluid simulations due to high accuracy and better stability properties. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: discontinuous Galerkin method; finite volume method; incompressible Navier-Stokes equations; DFG benchmark; implicit Euler

1. INTRODUCTION

In silicio experiments of fluids are an important part of today's scientific progress. They are used as a solution to engineering problems as well as to understand biological, physical and chemical processes. The flow and behavior of a fluid under external forces can be described by the Navier-Stokes equations, which can be used for the simulation of fluid phenomena. There is no general known analytical solution of the Navier-Stokes equations, it is therefore necessary to solve it using numerical methods. Well known methods are the finite element method, the finite volume method and the finite difference method. A method that is recently moving in the focus of research is the discontinuous Galerkin method, that became popular due to its unique features. It allows for high order solutions, high parallel computations, locally conservative, discontinuous ansatz functions, explicit as well as implicit solutions and non-conform meshes.

Standard discontinuous Galerkin methods (DG) have the problem of not being suitable for diffusion problems or higher order problems, due to bigger matrix system that need to be solved in every step when compared to standard finite element methods. To conquer this disadvantage so-called hybrid discontinuous Galerkin methods have been developed, that allow for the reduction of the original algebraic system to a reduced system for the faces only and thus reduces the degree of freedoms considerably [1]. One particular method for the Navier-Stokes equations that uses such a mechanism has been proposed by Nguyen et al. [2]. It is based on a formulation that has been used for several different applications by the same authors, see e.g. [3], [4], [5], [6] and [7] and even a space-time formulation [8].

*Correspondence to: Tobias Ahnert, Institut für Mathematik, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany

We implemented this hybridized discontinuous Galerkin method with an Oseen iteration in the Trilinos framework [9]. Due to high interest in the performance of the hybridized discontinuous Galerkin method, we like to compare it to a standard second order finite volume method (FV) provided by the icoFoam and simpleFoam solver from the OpenFOAM package [10]. We will show absolute and relative errors, compare to known experimental results and give time consumption needed by the solver in order to reach a certain accuracy. Further, we compare to known numerical values from the literature.

This work has the following structure: We begin by introducing the solvers used for the comparison by stating their respective mathematical formulation in section 2. Then we proceed to evaluate their properties numerically in section 3, where we show timings, accuracy and stability of the methods. The chosen cases are the Green-Taylor vortex, a 180 degrees fence case and the DFG benchmark [11], [12]. We conclude this work with a discussion of the results in section 4.

2. SOLVER USED FOR THE COMPARISON

In the literature there are many different methods usable for the incompressible Navier-Stokes equations. We chose one representative for each, the discontinuous Galerkin methods and the finite volume methods. We going to introduce the two methods used in this work by stating their mathematical formulation.

2.1. *Some notation*

The notation used in this work is similar to the original paper by Nguyen et al. [2].

Let $\Omega \subset \mathbb{R}^N$, $N = \{2, 3\}$ be an open connected subset with Lipschitz boundary $\partial\Omega$. $\mathcal{K}$ names a disjoint regular collection of elements $K \in \mathcal{K}$ with $\overline{\Omega} = \cup_{K\in\mathcal{K}} K$. The collection of all faces of $\mathcal{K}$ will be named $\mathcal{E}$ and we like to introduce the set of inner faces $\mathcal{E}_i := \{F \in \mathcal{E} : F \cap \Omega \neq \emptyset\}$ and outer faces $\mathcal{E}_\partial := \mathcal{E} \setminus \mathcal{E}_i$.

A scalar quantity will be named by a lower case letter, a vector quantity by a lower case bold letter and tensorial quantity by an upper case bold letter in the following sections, e.g. $a \in \mathbb{R}, \mathbf{a} \in \mathbb{R}^N, \mathbf{A} \in \mathbb{R}^{N\times N}$ is a scalar, vector and tensor, respectively. A tensor can be thought of as a $N \times N$ matrix, since we use only tensors of rank 2 without coordinate transformation. The components of a vector quantity $\mathbf{a}$ will be denoted by $a_i, \dots, a_N$ and the components of a tensorial quantity $\mathbf{A}$ by $A_{1,1}, \dots, A_{N,N}$. The symbol $\otimes$ represents a tensorial outer product, i.e. $(\mathbf{a} \otimes \mathbf{b})_{i,j} := a_i \cdot b_j$.

The standard $L^2(D)$ norm and scalar product on $D \subset \mathbb{R}^N$ is denoted by $\|\cdot\|_D$ and $(\cdot,\cdot)_D$, respectively and further the standard $L^2(D)$ scalar product on $D \subset \mathbb{R}^{N-1}$ is symbolized by $\langle\cdot,\cdot\rangle_D$. The symbol $\mathbb{P}_k(D)$ should denote the space of all polynomials up to order k on a domain D.

2.2. *Hybridized discontinuous Galerkin method*

We will present the hybridized discontinuous Galerkin method for the incompressible Navier-Stokes equations using a Picard iteration and based on [2] and similar to [8]. Suppose we like to discretize the incompressible Navier-Stokes equations consisting of the momentum and mass balance of an incompressible fluid with Dirichlet boundary condition, i.e.

$$\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} &= -\nabla \cdot (\mathbf{u} \otimes \mathbf{u} + p\mathbf{I} - \nu\nabla \otimes \mathbf{u}) + \mathbf{f}, \text{ in } \Omega \\
\nabla \cdot \mathbf{u} &= 0, \text{ in } \Omega \\
\mathbf{u} &= \mathbf{g}, \text{ on } \partial\Omega \\
\mathbf{u} &= \mathbf{u}_0, \text{ at } t = 0
\end{aligned} \tag{1}$$

where we have to fix the pressure by demanding

$$\int_\Omega p = 0 \, \mathrm{d}\mathbf{x}$$

and $\mathbf{g} \in L^2(\partial\Omega)$ has to adhere the condition $\int_{\partial\Omega} \mathbf{g} \cdot \mathbf{n} \, \mathrm{d}\mathbf{x} = 0$, otherwise the divergence-free condition is unsatisfiable.

Next, we introduce the finite dimensional spaces we are going to use for spatial discretization, that is

$$\begin{aligned}
P_h &:= \{v \in L^2(\mathcal{K}) : v|_K \in \mathbb{P}_k(K), \forall K \in \mathcal{K}\} \\
U_h &:= \{v \in L^2(\mathcal{K})^N : v|_K \in \mathbb{P}_k(K)^N, \forall K \in \mathcal{K}\} \\
L_h &:= \{v \in L^2(\mathcal{K})^{N\times N} : v|_K \in \mathbb{P}_k(K)^{N\times N}, \forall K \in \mathcal{K}\} \\
\mathcal{U}_h &:= \{v \in L^2(\mathcal{E})^N : v|_e \in \mathbb{P}_k(e)^N, \forall e \in \mathcal{E}\} \\
\mathcal{U}_h(\mathbf{a}) &:= \{v \in \mathcal{U}_h : v = \mathbf{a} \text{ on } \mathcal{E}_\partial\} \\
\mathcal{P}_h &:= \{v \in L^2(\partial\mathcal{K}) : v|_{\partial K} \in \mathbb{P}_0(\partial K), \forall K \in \mathcal{K}\}.
\end{aligned}$$

Note the spaces $\mathcal{U}_h$ and $\mathcal{P}_h$ are living on the boundaries of the elements and are used for the hybridization process of the discontinuous Galerkin method. The pressure space P_h and the velocity spaces L_h and U_h are of the same polynomial order. The method is therefore a so-called equal order method.

The discretized hybridizable discontinuous Galerkin method with an implicit Euler method for (1) is then find $(\mathbf{L}_h^m, \mathbf{u}_h^m, p_h^m, \widehat{\mathbf{u}}^m) \in (L_h, U_h, P_h, \mathcal{U}_h(\mathbf{g}))$ such that for all $(\mathbf{M}_h, \mathbf{v}_h, q_h, \widehat{\mathbf{v}}) \in (L_h, U_h, P_h, \mathcal{U}_h(\mathbf{0}))$

$$\begin{aligned}
(\mathbf{L}_h^m, \mathbf{M}_h)_\mathcal{K} + (\mathbf{u}_h^m, \nabla\cdot\mathbf{M}_h)_\mathcal{K} - \langle \widehat{\mathbf{u}}^m, \mathbf{n}\cdot\mathbf{M}_h\rangle_{\partial\mathcal{K}} &= 0 \\
(\frac{\mathbf{u}_h^m}{\Delta t}, \mathbf{v}_h)_\mathcal{K} - (\mathbf{w}_h^m \otimes \mathbf{u}_h^m + p_h^m \mathbf{I} - \nu \mathbf{L}_h^m, \nabla\otimes\mathbf{v}_h)_\mathcal{K} & \\
+\langle \mathbf{n}\cdot\widehat{\mathbf{F}}(\widehat{\mathbf{w}}^m), \mathbf{v}_h\rangle_{\partial\mathcal{K}} &= (\mathbf{f}^m, \mathbf{v}_h)_\mathcal{K} + (\frac{\mathbf{u}_h^{m-1}}{\Delta t}, \mathbf{v}_h)_\mathcal{K} \\
-(\mathbf{u}_h^m, \nabla q_h)_\mathcal{K} + \langle \mathbf{n}\cdot\widehat{\mathbf{u}}^m, q_h\rangle_{\partial\mathcal{K}} &= 0 \\
\langle \mathbf{n}\cdot\widehat{\mathbf{F}}(\widehat{\mathbf{w}}^m), \widehat{\mathbf{v}}\rangle_{\partial\mathcal{K}} &= 0 \\
\langle \mathbf{n}\cdot\mathbf{g}, 1\rangle_{\partial\mathcal{K}} &= 0 \\
(p_h, 1)_\mathcal{K} &= 0 \\
\mathbf{u}_h^0 &= \mathbf{u}_0
\end{aligned} \tag{2}$$

where $\widehat{\mathbf{F}}$ is given by

$$\begin{aligned}
\widehat{\mathbf{F}}(\widehat{\mathbf{w}}) &:= -\nu\widehat{\mathbf{L}_h} + \widehat{p_h}\mathbf{I} + \widehat{\mathbf{w}}\otimes\widehat{\mathbf{u}} \\
&:= -\nu\mathbf{L}_h + p_h\mathbf{I} + \widehat{\mathbf{w}}\otimes\widehat{\mathbf{u}} + \mathbf{n}\otimes\mathbf{s}(\mathbf{u}_h, \widehat{\mathbf{u}}, \mathbf{w}_h, \widehat{\mathbf{w}})
\end{aligned} \tag{3}$$

m denotes the mth time step of size Δt^m and $\mathbf{w}_h, \widehat{\mathbf{w}}$ is introduced to linearize the equation. The term $\mathbf{s}$ is a stabilization term defined as

$$\mathbf{s}(\mathbf{u}_h, \widehat{\mathbf{u}}, \mathbf{w}_h, \widehat{\mathbf{w}}) = [(\mathbf{u}_h - \widehat{\mathbf{u}}) \cdot \mathbf{S}(\mathbf{w}_h, \widehat{\mathbf{w}})]$$

and $\mathbf{S}(\mathbf{w}_h, \widehat{\mathbf{w}})$ is a stabilization tensor, that needs to be chosen appropriately. One can prove that the conditions $\mathbf{A}$ shall be positive definite and $\mathbf{u}$ must be divergence free, with

$$\mathbf{A} := \mathbf{S} - \frac{1}{2}\mathbf{n}\otimes\mathbf{u}$$

is sufficient, although in practice it converges even without $\mathbf{u}$ being divergence free. For more information on $\mathbf{S}$ see [13] and a proof of a more general result see [8]. We choose $\mathbf{S} = \mathbf{I}$ for all cases, but the non-stationary DFG benchmark, where we choose $\mathbf{S} = 2\mathbf{I}$. The incompressible Navier-Stokes equations can be solved by iteration of the system of equations (2) with $\mathbf{w}_h^m := \mathbf{u}_h^{m-1}$ and $\widehat{\mathbf{w}}^m := \widehat{\mathbf{u}}^{m-1}$. This is the well-known Picard iteration by use of the Oseen equations.

2.2.1. Hybridized formulation The incompressible Navier-Stokes equations can be directly solved using formulation (2) by use of a suitable time discretization, which must be an implicit method due to the differential algebraic character of equations (2). Unfortunately, the degree of freedoms are too high in comparison to other standard methods. The hybridizable discontinuous Galerkin methods allow us to solve for the approximate flux and the mean of the pressure only [2]. The velocity and pressure values can then be reconstructed from the flux and the mean pressure by local reconstruction. This whole process of reformulating is called hybridization (cf. [1]) and will be formulated for the stationary case of (2).

The stationary formulation of (2) with Dirichlet boundary condition is to find $(\mathbf{L}_h, \mathbf{u}_h, p_h, \widehat{\mathbf{u}}) \in (L_h, U_h, P_h, \mathcal{U}_h(\mathbf{g}))$ such that for all $(\mathbf{M}_h, \mathbf{v}_h, q_h, \widehat{\mathbf{v}}) \in (L_h, U_h, P_h, \mathcal{U}_h(\mathbf{0}))$ equations

$$\begin{aligned}
(\mathbf{L}_h, \mathbf{M}_h)_{\mathcal{K}} + (\mathbf{u}_h, \nabla\cdot\mathbf{M}_h)_{\mathcal{K}} - \langle\widehat{\mathbf{u}}, \mathbf{n}\cdot\mathbf{M}_h\rangle_{\partial\mathcal{K}} &= 0\\
-(\mathbf{w}_h\otimes\mathbf{u}_h + p_h\mathbf{I} - \nu\mathbf{L}_h, \nabla\otimes\mathbf{v}_h)_{\mathcal{K}} + \langle\mathbf{n}\cdot\widehat{\mathbf{F}}(\widehat{\mathbf{w}}), \mathbf{v}_h\rangle_{\partial\mathcal{K}} &= (\mathbf{f}, \mathbf{v}_h)_{\mathcal{K}}\\
-(\mathbf{u}_h, \nabla q_h)_{\mathcal{K}} + \langle\mathbf{n}\cdot\widehat{\mathbf{u}}, q_h\rangle_{\partial\mathcal{K}} &= 0\\
\langle\mathbf{n}\cdot\widehat{\mathbf{F}}(\widehat{\mathbf{w}}), \widehat{\mathbf{v}}\rangle_{\partial\mathcal{K}} &= 0\\
\langle\mathbf{n}\cdot\mathbf{g}, 1\rangle_{\partial\mathcal{K}} &= 0\\
(p_h, 1)_{\mathcal{K}} &= 0,
\end{aligned}\tag{4}$$

hold, where $\widehat{\mathbf{F}}$ is again given by (3).

Next, we introduce the so-called local solver $\mathcal{L}_h$ that maps $(\mathbf{c}, \boldsymbol{\lambda}, \rho) \in L^2(\Omega)^2 \times \mathcal{U}_h \times \mathcal{P}_h$ to the functions $(\mathfrak{L}_h, \mathfrak{u}_h, \mathfrak{p}_h) \in L_h \times U_h \times P_h$ fulfilling

$$\begin{aligned}
(\mathfrak{L}_h, \mathbf{G}_K)_K + (\mathfrak{u}_h, \nabla\cdot\mathbf{G}_K)_K &= \langle\boldsymbol{\lambda}, \mathbf{n}\cdot\mathbf{G}_K\rangle_{\partial K}\\
(\nu\mathfrak{L}_h - \mathfrak{p}_h\mathbf{I} - \mathbf{w}_h\otimes\mathfrak{u}_h, \nabla\otimes\mathbf{v}_K)_K &\\
+\langle\mathbf{n}\cdot(-\nu\mathfrak{L}_h + \mathfrak{p}_h\mathbf{I}) + \mathfrak{u}_h\cdot\mathbf{S}(\mathbf{w}_h, \widehat{\mathbf{w}}), \mathbf{v}_K\rangle_{\partial K} &= \langle-\mathbf{n}\cdot(\widehat{\mathbf{w}}\otimes\boldsymbol{\lambda})\\
&\quad + \boldsymbol{\lambda}\cdot\mathbf{S}(\mathbf{w}_h, \widehat{\mathbf{w}}), \mathbf{v}_K\rangle_{\partial K} + (\mathbf{c}, \mathbf{v})_K\\
-(\mathfrak{u}_h, \nabla q_K)_K &= -\langle\mathbf{n}\cdot\boldsymbol{\lambda}, q_K - \overline{q_K}\rangle_{\partial K}\\
\overline{\mathfrak{p}_h} &= \rho
\end{aligned}$$

for each $K \in \mathcal{K}$ and all $(\mathbf{G}_K, \mathbf{v}_K, p_K) \in \mathbb{P}_k(K)^{N\times N} \times \mathbb{P}_k(K)^N \times \mathbb{P}_k(K)$ with $\overline{a}$ denoting the average of a on ∂K, i.e.

$$\overline{a} = \frac{1}{|\partial K|}\int_{\partial K} a(\mathbf{x})\,\mathrm{d}\mathbf{x}.$$

The pure Dirichlet problem has an under-defined pressure, so we need to introduce $\rho \in \mathcal{P}_h$, in order to fix the pressure by a constant. The local solver is a linear differential equation in all three parameters, therefore we can set

$$\begin{aligned}
(\mathfrak{L}_h^c, \mathfrak{u}_h^c, \mathfrak{p}_h^c) &:= \mathcal{L}_h(\mathbf{c}, \mathbf{0}, 0)\\
(\mathfrak{L}_h^{\boldsymbol{\lambda}}, \mathfrak{u}_h^{\boldsymbol{\lambda}}, \mathfrak{p}_h^{\boldsymbol{\lambda}}) &:= \mathcal{L}_h(\mathbf{0}, \boldsymbol{\lambda}, 0)\\
(\mathfrak{L}_h^{\rho}, \mathfrak{u}_h^{\rho}, \mathfrak{p}_h^{\rho}) &:= \mathcal{L}_h(\mathbf{0}, \mathbf{0}, \rho).
\end{aligned}$$

Just as in [2], we can state theorem 2.1, which tells us how to solve formulation (4) effectively.

Theorem 2.1
Let $(\mathbf{L}_h, \mathbf{u}_h, p_h, \widehat{\mathbf{u}})$ solve formulation (4). Then we have

$$\begin{aligned} \mathbf{L}_h &= \mathfrak{L}_h^c + \mathfrak{L}_h^{\boldsymbol{\lambda}} + \mathfrak{L}_h^\rho \\ \mathbf{u}_h &= \mathfrak{u}_h^c + \mathfrak{u}_h^{\boldsymbol{\lambda}} + \mathfrak{u}_h^\rho \\ p_h &= \mathfrak{p}_h^c + \mathfrak{p}_h^{\boldsymbol{\lambda}} + \mathfrak{p}_h^\rho \\ \widehat{\mathbf{u}} &= \boldsymbol{\lambda}, \end{aligned}$$

where $(\boldsymbol{\lambda}, \rho) \in \mathcal{U}_h(\mathbf{g}) \times \mathcal{P}_h$ is the solution of the following weak formulation

$$\begin{aligned} a_h(\boldsymbol{\lambda}, \widehat{\mathbf{v}}) + b_h(\rho, \widehat{\mathbf{v}}) &= c_h(\widehat{\mathbf{v}}) \\ b_h(\sigma, \boldsymbol{\lambda}) &= 0 \end{aligned}$$

for all $(\widehat{\mathbf{v}}, \sigma) \in \mathcal{U}_h(0) \times \mathcal{P}_h$. Here the forms are given by

$$\begin{aligned} a_h(\boldsymbol{\lambda}, \widehat{\mathbf{v}}) &= \langle \mathbf{n} \cdot (-\nu \mathfrak{L}_h^{\boldsymbol{\lambda}} + \mathfrak{p}_h^{\boldsymbol{\lambda}} \mathbf{I} + \widehat{\mathbf{w}} \otimes \boldsymbol{\lambda}), \widehat{\mathbf{v}} \rangle_{\partial \mathcal{K}} + \langle (\mathfrak{u}_h^{\boldsymbol{\lambda}} - \boldsymbol{\lambda}) \cdot \mathbf{S}(\mathbf{w}_h, \widehat{\mathbf{w}}), \widehat{\mathbf{v}} \rangle_{\partial \mathcal{K}} \\ b_h(\sigma, \widehat{\mathbf{v}}) &= \langle \sigma, \mathbf{n} \cdot \widehat{\mathbf{v}} \rangle_{\partial \mathcal{K}} \\ c_h(\widehat{\mathbf{v}}) &= -\langle \mathbf{n} \cdot (-\nu \mathfrak{L}_h^{\mathbf{c}} + \mathfrak{p}_h^{\mathbf{c}}) + \mathfrak{u}_h^{\mathbf{c}} \cdot \mathbf{S}(\mathbf{w}_h, \widehat{\mathbf{w}}), \widehat{\mathbf{v}} \rangle_{\partial \mathcal{K}} \end{aligned}$$

for all $(\boldsymbol{\lambda}, \widehat{\mathbf{v}}, \sigma) \in \mathcal{U}_h \times \mathcal{U}_h \times \mathcal{P}_h$.

This can be simply proven with the same lines as for the Newton-Raphson iteration case in [2]. For the detailed proof see [13].

Hybridized discontinuous Galerkin methods are relatively new methods that just become popular in the last couple of years. Their advantages are the reduced degree of freedom, higher convergence order and (in the case of Nguyen et al.'s solver) local conservation properties via a post-processing step, cf. [2].

2.3. *Implementation of the DG method*

We implemented the described discontinuous Galerkin method in C++. The program is just using the matrix assembly routine and XML functions of the Trilinos framework [9] as well as the Visualization Toolkit (VTK) [14] for output. The mathematical core components like the matrix assembly, the polynomial creation/integration and mesh creation are written completely from scratch. We did so to better evaluate the implementation problems that might arise from the discontinuous Galerkin approach. Since we have used low order polynomials, i.e. not higher than order 3, we used Lagrangian polynomials with a Gaussian integration scheme of order 5.

In our view the discontinuous Galerkin method is harder to implement than standard finite element methods due to the added complexity of boundary integrals and variables from the $\mathcal{U}_h$ and $\mathcal{P}_h$ spaces. Further, the hybridization process adds another complexity due to the need of implementing local solvers.

We tried different iterative solvers like GMRES and BiCG with multiple choices of standard preconditions like incomplete LU, diagonal preconditioners and alike, but all of them failed to converge quickly for the arising matrix system, so we finally settled with the direct matrix solver UMFPACK [15]. The lack of suitable and easily usable preconditioners for hybridizable discontinuous Galerkin methods is a major issue for practical use in engineering applications. Figure 3 shows a typical matrix structure created by the hybridized DG solver.

For now, we just considered triangulated meshes with linear boundary elements. This way we could easily pre-compute reference matrices for the domain and boundary elements. For more complicated elements the matrix assembling can get rather inefficient due to the high number of degrees of freedom and hence the high number of integrations needed. Further, we used uniformly distributed lagrangian polynomials as ansatz functions. In retrospect this choice of ansatz functions is not optimal, because it results in an increases condition number. For different polynomial ansatz functions in use with discontinuous Galerkin methods see e.g. [16] and [17].

2.4. *icoFoam finite volume method*

The incompressible finite volume tool icoFoam of the OpenFOAM package [10] uses a standard discretization for the momentum equation combined with the PISO method (c.f. [18]) in order to compute the pressure. In this work, we use the implicit Euler method to fulfill the equation

$$\begin{aligned}\int_K \frac{\mathbf{v}^{n+1}-\mathbf{v}^n}{\Delta t}\,\mathrm{d}\mathbf{x} + \int_{\partial K} (\mathbf{n}\cdot\mathbf{\Phi}^{n+1})\mathbf{v}^{n+1}\,\mathrm{d}\mathbf{s} - \\ \int_K \nabla\cdot\mu(\nabla\otimes\mathbf{v}^{n+1})\,\mathrm{d}\mathbf{x} = -\int_K \nabla p^{n+1}\,\mathrm{d}\mathbf{x} \ \ \forall K\in\mathcal{K} \\ \nabla\cdot\mathbf{v}^{n+1} = 0,\end{aligned} \tag{5}$$

where $\mathbf{\Phi}$ is the velocity interpolated to the faces and the upper index symbolizes the time step. OpenFOAM is using a kind of Rhie-Chow interpolation for flux fields, which we will symbolize by Π.

2.4.1. Velocity predictor step To solve the nonlinear equation system we use the following prediction correction method based on the Chorin projection idea [19].

First, we compute an approximate velocity by usage of the old pressure and velocity values, so the momentum equation from system (1) becomes

$$\begin{aligned}\int_K \frac{\tilde{\mathbf{v}}-\mathbf{v}^n}{\Delta t}\,\mathrm{d}\mathbf{x} + \int_{\partial K} (\mathbf{n}\cdot\mathbf{\Phi}^{n})\tilde{\mathbf{v}}\,\mathrm{d}\mathbf{s} - \\ \int_K \nabla\cdot\mu(\nabla\otimes\tilde{\mathbf{v}})\,\mathrm{d}\mathbf{x} = -\int_K \nabla p^{n}\,\mathrm{d}\mathbf{x} \ \ \forall K\in\mathcal{K}\end{aligned} \tag{6}$$

in a finite volume context, where we set the volume forces to zero.

Then the algebraic equation for a single cell $i\in\mathcal{E}$ of (6) becomes

$$a_i\tilde{\mathbf{v}}_i + \sum_{n\in\mathcal{N}(i)} a_n\tilde{\mathbf{v}}_n = \mathbf{b}_i - \nabla p_i \tag{7}$$

in discretized form, where $a_i\in\mathbb{R}$ are the coefficients for $\tilde{\mathbf{v}}_i$ and $\mathbf{b}$ represents the right hand side of the algebraic equation without the pressure.

2.4.2. Pressure correction loop Let A be the diagonal matrix containing all a_i from equation (7), that is for $k=\lfloor j/3\rfloor$ let $(A)_{jj}=a_k$ and $(A)_{ij}=0$ for $i\neq j$ and further let H be the vector containing all $a_n\tilde{\mathbf{v}}_n$ and the right hand side $\mathbf{b}_i$, that is

$$H_{3i+j} = (-\sum_{n\in\mathcal{N}(i)} a_n\tilde{\mathbf{v}}_n + \mathbf{b}_i)_j \text{ for } j\in\{1,2,3\}.$$

This H-operator is common for OpenFOAM based implementations. We then compute a Jacobi step for $\tilde{\mathbf{v}}$ with

$$\tilde{\mathbf{v}}_{\text{jac}} = A^{-1}H$$

Next, we compute $\mathbf{\Phi}=\Pi\tilde{\mathbf{v}}_{\text{jac}}$ followed by

$$\nabla\cdot(A^{-1}\nabla p) = \nabla\cdot\mathbf{\Phi}$$

to compute the new pressure p. The face flux $\mathbf{\Phi}$ is then corrected by

$$\mathbf{\Phi} = \Pi(A^{-1}H - A^{-1}\nabla p)$$

followed by the correction of the predicted velocity

$$\tilde{\mathbf{v}} := \tilde{\mathbf{v}}_{\text{jac}} - A^{-1}\nabla p.$$

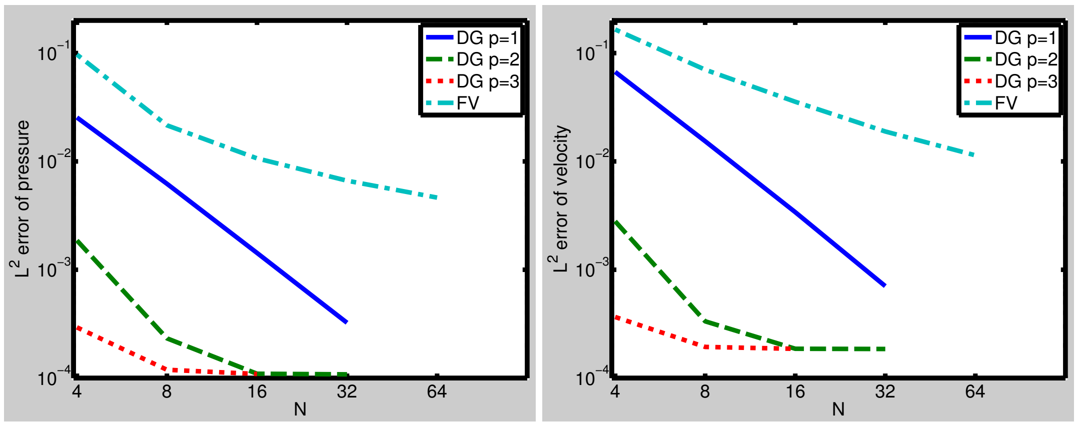


Figure 1. The absolute errors of the DG and the FV method for the Taylor-Green vortex. Several polynomial degrees are shown for the DG method.

The pressure correction loop is repeated until the pressure converges or a maximum number of rounds is reached. The resulting values for $\tilde{\mathbf{v}}$, p and $\mathbf{\Phi}$ are then used as solution for time step $n+1$. In other words the result fulfills the equations (5) at the end of the pressure correction loop.

The icoFoam solver has been chosen for its very broad use in engineering applications due to the popularity of the OpenFOAM package. Further, it is very common for finite volume methods to use an operator splitting approach for the pressure as done by the PISO loop in icoFoam. The simpleFoam solver works the same as the icoFoam solver, but has no time depending part. Hence, its application are stationary solutions of the incompressible Navier-Stokes equations.

3. NUMERICAL EXAMPLES

3.1. *Taylor vortex*

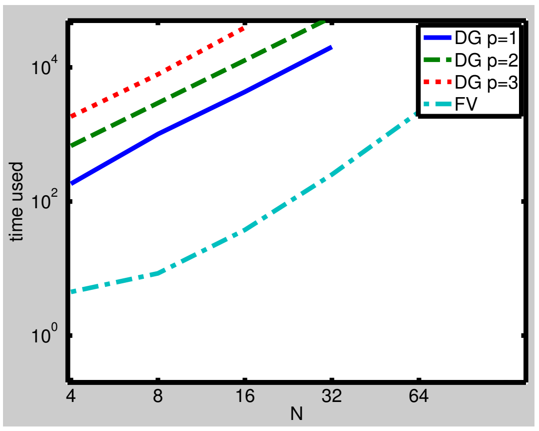
We solved the well known Taylor-Green vortex with analytic solution

$$\begin{aligned} u_1 &= -\cos(\pi x_1)\sin(\pi x_2)\exp(\frac{-2\pi^2 t}{\mathrm{Re}}) \\ u_2 &= \sin(\pi x_1)\cos(\pi x_2)\exp(\frac{-2\pi^2 t}{\mathrm{Re}}) \\ p &= -\frac{1}{4}(\cos(2\pi x_1)+\cos(2\pi x_2))\exp(\frac{-4\pi^2 t}{\mathrm{Re}}) \end{aligned}$$

on the full periodic domain $[0,2]\times[0,2]$. Similar simulations have been done before in [2], but without comparison to a different solver. We used the non-stationary DG and the icoFoam solvers for this case. We chose a time step of size $\mathrm{dt}=0.001$. In the following sections p will denote the polynomial order used in the DG method and $N=1/h$, with h being a typical element size.

Figure 1 shows the convergence of the absolute errors of the discontinuous Galerkin method and the finite volume method. The DG method converges with order $k+1$ at polynomial degree k and the FV method with order 1 when measured in a L^2-norm against the analytical result.

The time used for the computation of the solution is shown in Figure 2. As one can see our implementation is quite slow compared to icoFoam for low order polynomials. The situation changes with higher order polynomial ansatz and very high accuracy as shown in Figure 2. As one can also see from the figures, the time error becomes dominant for very accurate solutions in the DG solver. Surprisingly, when computed with too big time steps, the two solvers showed different behavior, i.e. icoFoam shows instabilities in the results and the DG solver computes a valid solution, whose accuracy is limited by the time-marching error.



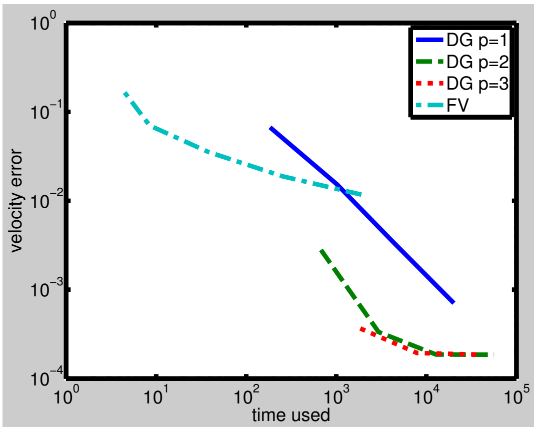
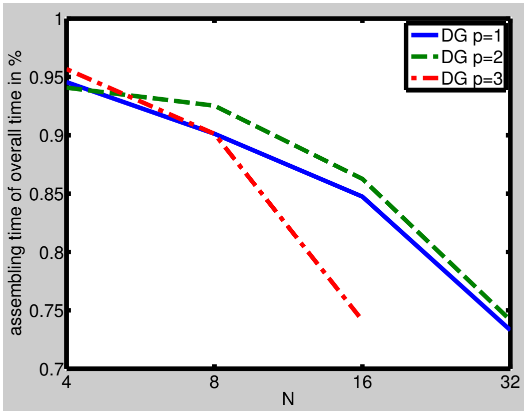


Figure 2. Left: The time used by DG and FVM for the given accuracy. Right: The time vs error plot for the Taylor-Green vortex. The finite volume solver is much faster for small grids, but when using a high polynomial order the DG solver becomes faster for the same accuracy.



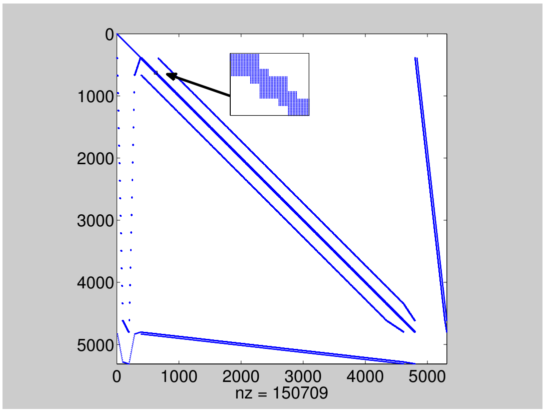


Figure 3. Left: The overall time used for the assembling of the matrix in percent for the DG solver in the Taylor-Green vortex simulation. Right: An example for the matrix structure created by the hybridized DG solver.

The time used to assemble the matrix in relation to the overall time is shown in Figure 3. One can clearly see our solver takes too much time assembling the matrix. However, due to the fact the assembling is of a lower computational order than the matrix solver, the problem becomes less problematic for bigger problems.

3.2. Fence

As a test for the steady case solvers we chose the so-called fence case, where a small bar is located in a laminar stream creating vortices on both sides of it. The geometry is chosen as in Tanedas experiments [20], i.e. the length of the domain is 10 and the height of the bar is $\frac{1}{30}$th of the length. The Reynolds number is 0.014, where the characteristic length scale is the height of the bar, which has been chosen as $\frac{1}{3}$ and we chose a pipe height of 3 with a quadratic inlet condition. We performed comparison of relative errors, with experimental pictures and we compare the vortices distance given by Taneda [20].

The result of the DG simulation is shown in figures 5. The distance from the center of the vortices to the corner is measured in experiments as 0.54 the height of the fence [20]. The distance value for the finite volume solver has been measured as 0.5026 and for the discontinuous Galerkin method as 0.5065. Both values are smaller than the experimental value, but fairly close to each other. This

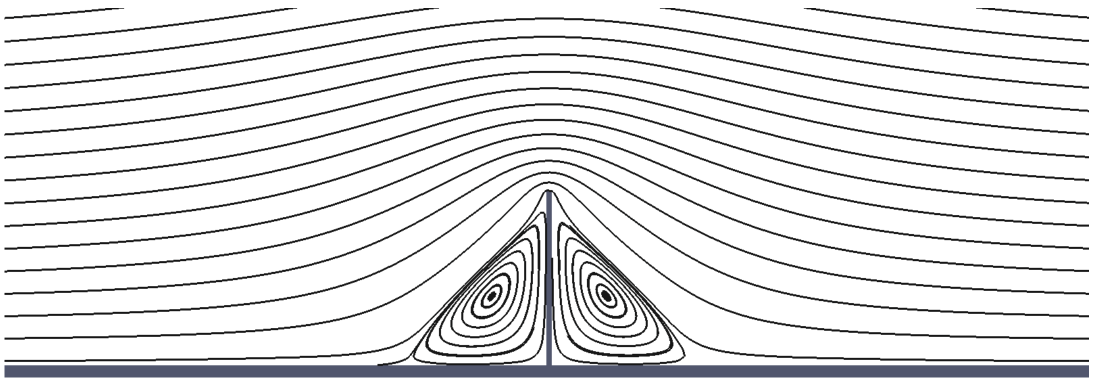


Figure 4. Streamline plot of flow through the magnified fence domain. Simulation created with the DG solver.

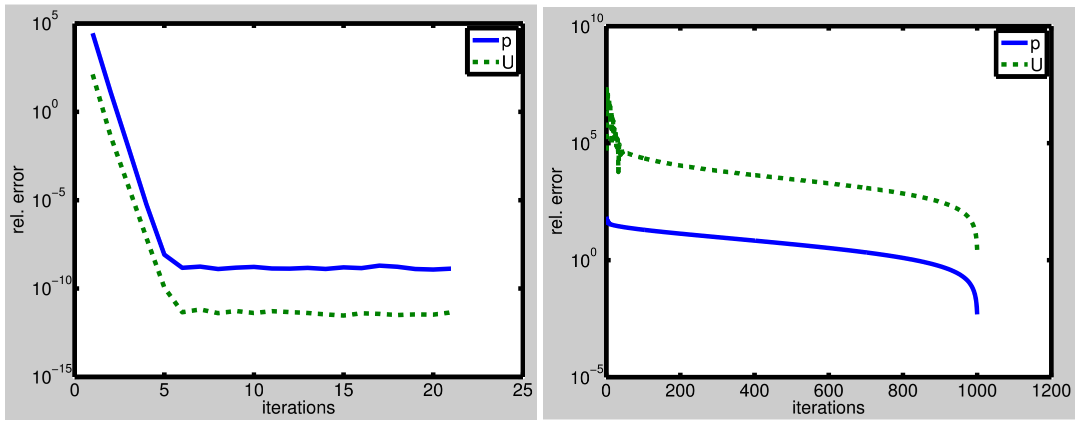


Figure 5. Shown are the relative errors (absolute change to previous iteration) for the fence case. On the left hand side for the discontinuous Galerkin method and on the right hand side for the finite volume method.

leads to the conclusion that both simulations have the same limiting factors like the finite size of the channel, being a 2D simulation only and the mesh quality.

The relative error of the simulations performed is shown in 4. One can tell from this plot that the discontinuous Galerkin method needs just a few (less than 10) iterations steps to fully converge, whereas the finite volume solver has not fully converged up to 1000 steps for the same mesh and boundary conditions. We tested with different values for the finite volume solution methods/schemes and always received similar results. Therefore, we think the DG solver is more suited for stationary problems. We have seen the same behavior for the stationary DFG benchmark.

3.3. *DFG benchmark*

The DFG benchmark (cf. [11]) is a well known benchmark, that uses different directly measurable characteristic values. The geometry is shown in Figure 6 and is of size 2.2×0.41. The circle has radius $R = 0.05$ and has its center located at point $(0.2, 0.2)$. We chose the same unstructured Delauney grid for both the finite volume solvers and the discontinuous Galerkin solver, but extruded it for OpenFOAM (because it needs 3D meshes). We chose a characteristic element size of $\frac{1}{32}$ for all of the domain, but the circle, where we chose $\frac{1}{256}$ as a refinement. In order to highlight the extra option of polynomial refinement in the DG solver, we solved for three different polynomial orders ($p \in \{1, 2, 3\}$). The finite volume solver has been used with the original mesh and two more refined meshes, that simply have halved and quartered characteristic element ($N \in \{1, 2, 3\}$) sizes, respectively.

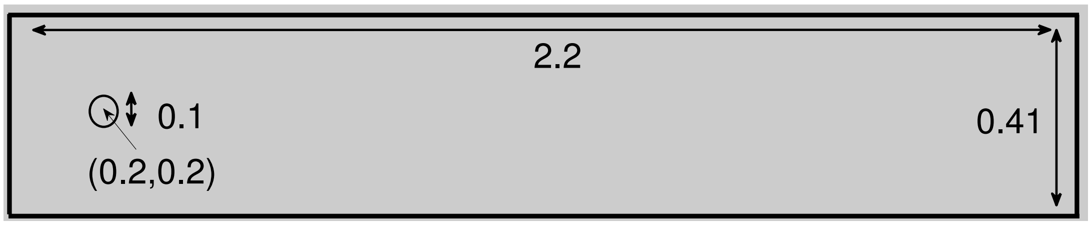


Figure 6. The DFG benchmark domain as proposed by [11].

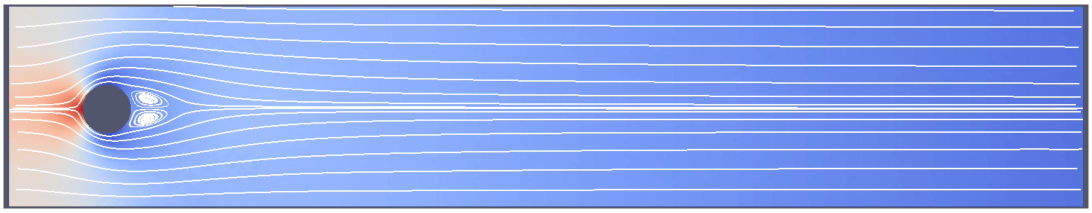


Figure 7. Pressure plot with streamlines of DFG benchmark computed with DG solver and polynomial degree 1.

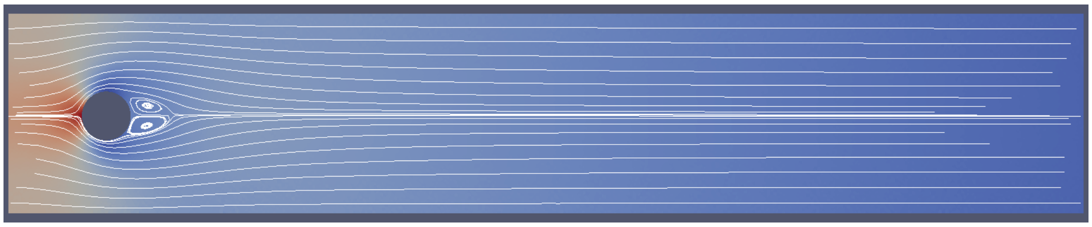


Figure 8. Pressure plot and streamlines of DFG benchmark computed with simpleFoam solver.

A quadratic inlet profile defined as

$$U_{\text{in}}(0, y) = (4 \cdot U_{\max} \cdot y(0.41 - y)/0.41^2, 0)$$

has been chosen at the left hand side and a pressure outlet condition on the right hand side of the domain. The viscosity is $\nu = 0.001$ and $\rho = 1$.

3.3.1. Stationary case The stationary case is defined with maximum inlet velocity as $U_{\max} = 0.3$. To more accurately compare the solutions, we compute the drag and lift coefficients around the circle. The drag and lift coefficients c_d and c_l are defined as

$$c_d = \frac{F_w}{\rho \overline{U}^2 R} \qquad\qquad c_l = \frac{F_a}{\rho \overline{U}^2 R}$$

and forces

$$\begin{pmatrix} F_w \\ F_a \end{pmatrix} = \int_S [-pI + \rho\nu(\nabla \otimes \mathbf{u})]\mathbf{n}\,\mathrm{d}s.$$

with S being the surface of the circle, $\mathbf{n}$ the normal of S and $\overline{U} = 0.2$ is the average inlet velocity. The Reynolds number is defined as $\mathrm{Re} = \frac{\overline{U}2R}{\nu}$ and has value $\mathrm{Re} = 20$ for this case. Figures 7 and 8 show the pressure and velocity plots for discontinuous Galerkin and finite volume ansatz, respectively.

Table I shows the result of the coefficients. Comparison with the values from [11] shows the c_d and c_l inside the original bounds of $c_d \in [5.5700, 5.5900]$ and $c_l \in [0.0104, 0.0110]$. The DG results are as good as the OpenFOAM results on the mesh, but it took all of the DG runs less than 15

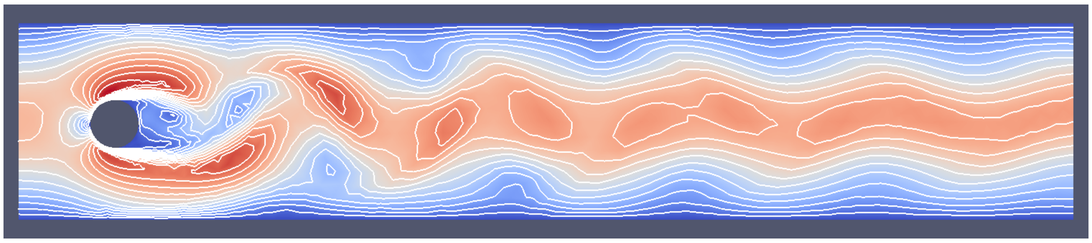


Figure 9. Velocity magnitude plot of non-stationary DFG benchmark computed with DG solver. A contour plot was used to highlight the different velocity regimes of the vortex street.

iterations to converge, whereas OpenFOAM needed well-over between 300 and 4500 iterations to fully converge, depending on the mesh size. The absolute computation time used by both solvers is about the same and in the realm of just a few minutes on a standard workstation.

Two other values presented in Table I are the pressure difference Δp between the points $(0.15, 0.2)$ and $0.25, 0.2$, i.e. the end points of the circle as well as the length of the recirculation area L_a, i.e. maximum x value of recirculation area minus 0.25 - the end of the circle. On can see the DG solver is a good alternative to FV methods and is in the error bounds from [11].

The numbers get even closer for higher resolutions around the cylinder, which can be explained by the fact, that we do not use curvilinear elements for the circle. Thus, a higher resolution make the circle more round and therefore the numbers increasingly more accurate. To demonstrate this effect, we computed a high resolution solution and marked it with (hr) in Table I.

3.3.2. Non-stationary case The non-stationary case has a maximum inlet velocity $U_{\max} = 1.5$, i.e. $\overline{U} = 1.0$. The Reynolds number has value $\mathrm{Re} = 100$ and it has a non-stationary time-periodic solution as shown in Figure 9. In order to gain the periodic solution we first started with an initial condition of zero velocity and pressure, then computed the time period $T = [0, 6]$, at which end the discontinuous Galerkin as well as the OpenFOAM solver reached the stable periodic solution. The start time t_n of a period n is defined as the time of recuring maximum lift coefficient. We took the lift and drag coefficients for a whole period and computed the pressure difference Δp at the point in time at the center of two consecutive period beginnings, i.e. $t_p = (t_{n+1} + t_n)/2$. Further, the Strouhal number $\mathrm{St} = \frac{2Rf}{\overline{U}}$ has been measured with f the frequency of separation [11], i.e. $f = 2/(t_{n+1} - t_n)$. The constant time step size used is $\Delta t = 0.0005$ for both methods.

The results are given in Table II and are in good agreement to the numbers given in [11]. Again the DG solver's accuracy is at least as good as the finite volume method. However, due to the slow assembling routine, the time consumed by the DG solver was considerably larger than for the icoFoam solver, i.e. about a factor of 12 for a case with the same number of degree of freedoms in the velocity (DG p=1 vs icoFoam N=2). However, we note that this comparison is the worst case for our DG solver, since it performs better for higher polynomial degrees. This might be reduce by a factor of 5 by a thorough optimization, but still in that case icoFoam stays faster for non-stationary problems, partly due to the need of a direct solver for the DG method. From the experience of the Taylor-Green vortex we expect a different behavior for high accuracy cases with higher polynomial degree.

4. DISCUSSION

Summarizing the previous results, we note discontinuous Galerkin methods are a viable alternative to standard finite volume methods. Although our implementation is slow at assembling the matrices (its the main time consumed by every iteration), the higher the expected accuracy the more competitive it gets, due to the option of an higher polynomial degree. Further, the discontinuous Galerkin method allows one to specify the boundary conditions in a more mathematical rigorous way, than the finite volume method, i.e. if purely velocity boundary conditions are given, icoFoam and simpleFoam need artificial pressure conditions due to their PISO algorithm, whereas in discontinuous Galerkin methods this is not necessary. The major advantages of finite volume methods are the local conservative nature of these methods, our discontinuous Galerkin method also possesses this property via a post-processing step, described in [2], which also gives an additional order of convergence for the velocity. The benchmarks show the physical accuracy of both methods to be about the same for most problems. The convergence rate is higher for DG methods for high polynomial orders. With support to the use of and easy to implement higher polynomial ansatzes and the resulting high accuracy the DG method could be a real alternative to high resolution spectral methods to resolve also all fundamental scales of turbulent flows.

The hybridized discontinuous Galerkin methods are not capable of using explicit time marching schemes, whereas finite volume methods are able to use them. Their are explicit discontinuous Galerkin methods, but they have the disadvantage of possessing a higher number of degrees of freedom in comparison with finite volume methods, see [16] for an introduction. Therefore, it is always necessary to solve an unsymmetric matrix equation. Iterative solvers like GMRES and BiCG with preconditioners like the incomplete lu-decomposition failed to converge quickly for the arising equation. We think this can be enhanced by preconditioners that use the hybridized discontinuous Galerkin matrix structure. We finally settled with a direct solver, i.e. UMFPACK [15], which solved the arising matrix system robust and fast. We do not use matrix condensation currently and there is still room for optimization possible for the polynomial ansatz functions.

The local solver solutions must be preserved until the reconstruction is done. Hence, the memory footprint of the DG solver is much larger than for the finite volume method. Although the exact footprint might be implementation specific, we think a higher memory consumption is a general characteristic of hybridized DG solver.

The steady state finite volume solver simpleFoam needs under-relaxation in order to converge. The discontinuous Galerkin method had no need for stability mechanisms other than the $\mathbf{S}$ tensor and converged with a smaller number of iterations. Generally, the DG method seemed more stable as for large CFL numbers (near 1) the finite volume solver failed converging and the DG solver produced valid results up to the time marching error, although both used the same implicit Euler schemes. In our experience the stability tensor $\mathbf{S}$ can be chosen much smaller than the proposed values. Thus, a sharper stability and accuracy result would be convenient.

Possible future work is to expand the DG solver to 3D problems, make it much more competitive by optimizing the matrix assembly routine and compare with 3D test cases. Further, discontinuous Galerkin methods are said to offer good parallel structures we would like to exploit and benchmark in the future.

5. ACKNOWLEDGEMENTS

One of the authors (TA) gratefully acknowledges the support by the Federal Ministry of Education (BMBF) and the state government of Berlin (SENBWF) in the framework of the program “Spitzenforschung und Innovation in den Neuen Ländern” (Grant Number 03IS2151). Furthermore, the authors (TA, GB) gratefully acknowledge the support by the Technische Universität Berlin.

REFERENCES

1. Cockburn B, Gopalakrishnan J, Lazarov R. Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis* 2009; **47**(2):1319–1365, doi:10.1137/070706616.
2. Nguyen N, Peraire J, Cockburn B. An implicit high-order hybridizable discontinuous galerkin method for the incompressible navier–stokes equations. *Journal of Computational Physics* 2011; **230**(4):1147 – 1170, doi: 10.1016/j.jcp.2010.10.032.
3. Nguyen N, Peraire J, Cockburn B. A hybridizable discontinuous galerkin method for stokes flow. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(9–12):582 – 597, doi:10.1016/j.cma.2009.10.007.
4. Nguyen N, Peraire J, Cockburn B. Hybridizable discontinuous galerkin methods for the time-harmonic maxwell’s equations. *Journal of Computational Physics* 2011; **230**(19):7151 – 7175, doi:10.1016/j.jcp.2011.05.018.

5. Nguyen N, Peraire J, Cockburn B. High-order implicit hybridizable discontinuous galerkin methods for acoustics and elastodynamics. *Journal of Computational Physics* 2011; **230**(10):3695 – 3718, doi:10.1016/j.jcp.2011.01.035.
6. Nguyen N, Peraire J, Cockburn B. An implicit high-order hybridizable discontinuous galerkin method for linear convection–diffusion equations. *Journal of Computational Physics* 2009; **228**(9):3232 – 3254, doi:10.1016/j.jcp.2009.01.030.
7. Nguyen N, Peraire J, Cockburn B. An implicit high-order hybridizable discontinuous galerkin method for nonlinear convection–diffusion equations. *Journal of Computational Physics* 2009; **228**(23):8841 – 8855, doi:10.1016/j.jcp.2009.08.030.
8. Rhebergen S, Cockburn B. A space–time hybridizable discontinuous galerkin method for incompressible flows on deforming domains. *Journal of Computational Physics* 2012; **231**(11):4185 – 4204, doi:10.1016/j.jcp.2012.02.011.
9. Heroux MA, Bartlett RA, Howle VE, Hoekstra RJ, Hu JJ, Kolda TG, Lehoucq RB, Long KR, Pawlowski RP, Phipps ET, *et al.*. An overview of the trilinos project. *ACM Trans. Math. Softw.* 2005; **31**(3):397–423, doi: http://doi.acm.org/10.1145/1089014.1089021.
10. Openfoam 2.1.1. URL `http://www.openfoam.org.`, last checked 2-2013.
11. Turek S, Schäfer M. Benchmark computations of laminar flow around a cylinder. *Flow Simulation with High-Performance Computers II*, Hirschel EH (ed.). No. 52 in NNFM, Vieweg, 1996.
12. John V, Matthies G. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *International Journal for Numerical Methods in Fluids* 2001; **37**(8):885–903, doi:10.1002/fld.195.
13. Ahnert T. Discontinuous galerkin method for the incompressible navier-stokes equation. master thesis, Technical University Berlin 2012-1-10.
14. Vtk 5.10. URL `http://www.vtk.org.`, last checked 2-2013.
15. Davis TA. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.* Jun 2004; **30**(2):165–195, doi:10.1145/992200.992205.
16. Hesthaven J, Warburton T. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics Series, Springer-Verlag New York, 2008.
17. Gassner GJ, Lörcher F, Munz CD, Hesthaven JS. Polymorphic nodal elements and their application in discontinuous galerkin methods. *Journal of Computational Physics* 2009; **228**(5):1573 – 1590, doi:10.1016/j.jcp.2008.11.012.
18. Issa R. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics* 1986; **62**(1):40 – 65, doi:10.1016/0021-9991(86)90099-9.
19. Chorin AJ. Numerical solution of the navier-stokes equations. *Math. Comp* 1968; **22**(104):745–762.
20. Taneda S. Visualization of separating stokes flows. *Journal of the Physical Society of Japan* 1979; **46**(6):1935–1942, doi:10.1143/JPSJ.46.1935.

Method	c_d	c_l	L_a	Δp
DG p=1	5.53246	0.012776	0.082346	0.1183123
DG p=2	5.57557	0.0106114	0.081156	0.1175033
DG p=3	5.57487	0.0108511	0.081250	0.1174782
DG p=1 (hr)	5.5739	0.01096	0.0845	0.1174
simpleFoam N=1	5.63391	−0.000663409	0.07943048	0.1132308
simpleFoam N=2	5.65306	0.0117287	0.08216164	0.116371
simpleFoam N=3	5.64141	0.0146669	0.08249009	0.1169138
lower bound	5.5700	0.0104	0.0842	0.1172
upper bound	5.5900	0.0110	0.0852	0.1176

Table I. Measured values for stationary DFG benchmark for the finite volume and the discontinuous Galerkin method as well as the reference bounds from [11].

Method	$c_{d\,max}$	$c_{l\,max}$	Δp	St
DG p=1	3.17162	0.993902	2.485552	0.2962963
DG p=2	3.2268	1.00068	2.481253	0.2989537
DG p=3	3.22473	0.997769	2.479035	0.2994012
icoFoam N=1	3.14361	0.901461	2.47068	0.2631579
icoFoam N=2	3.22667	0.991152	2.469584	0.2928258
icoFoam N=3	3.21026	0.962411	2.463615	0.295421
lower bound	3.2200	0.9900	2.4600	0.2950
upper bound	3.2400	1.0100	2.5000	0.3050

Table II. Measured values for non-stationary case of DFG benchmark for the finite volume and the discontinuous Galerkin method as well as the reference bounds from [11].