Journal of Computational Physics 233 (2013) 339–358

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A space–time discontinuous Galerkin method for the incompressible Navier–Stokes equations

Sander Rhebergen [a,*], Bernardo Cockburn [a], Jaap J.W. van der Vegt [b]

[a] School of Mathematics, University of Minnesota, Minneapolis, MN 55455, USA
[b] Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 11 April 2012
Received in revised form 17 July 2012
Accepted 31 August 2012
Available online 17 September 2012

Keywords:
Space–time discontinuous Galerkin method
Incompressible Navier–Stokes equations
Deforming domains

ABSTRACT

We introduce a space–time discontinuous Galerkin (DG) finite element method for the incompressible Navier–Stokes equations. Our formulation can be made arbitrarily high-order accurate in both space and time and can be directly applied to deforming domains. Different stabilizing approaches are discussed which ensure stability of the method. A numerical study is performed to compare the effect of the stabilizing approaches, to show the method's robustness on deforming domains and to investigate the behavior of the convergence rates of the solution. Recently we introduced a space–time hybridizable DG (HDG) method for incompressible flows [S. Rhebergen, B. Cockburn, A space–time hybridizable discontinuous Galerkin method for incompressible flows on deforming domains, J. Comput. Phys. 231 (2012) 4185–4204]. We will compare numerical results of the space–time DG and space–time HDG methods. This constitutes the first comparison between DG and HDG methods.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

This is our third article in a series devoted to obtaining accurate and efficient numerical schemes for solving the incompressible Navier–Stokes equations on deforming domains. Examples of applications these schemes may be used for include fluid–structure interaction in wind turbine simulations [14] or flows with free-surfaces, such as water waves and sloshing [24].

Accurately solving partial differential equations by a numerical method on moving and deforming domains, however, is non-trivial. Many schemes fail to automatically preserve uniform flow on moving or deforming meshes. This requirement is used to derive the Geometric Conservation Law (GCL) [17] and in [13,17] it is shown that the GCL is essential for the time-accuracy of the solution.

Recently, Persson et al. [20] introduced a discontinuous Galerkin (DG) method for solving the compressible Navier–Stokes equations on deformable domains. They introduced a continuous mapping between a fixed reference configuration and the time-varying domain. The Navier–Stokes equations are then written in the reference configuration, where the equations are solved using a DG method in space and Runge–Kutta method in time. This method is simple and allows the use of explicit time-stepping, however, it does not automatically satisfy the GCL and an extra equation needs to be solved to prevent loss of accuracy.

A different approach is that of the space–time DG method (see e.g. [16,22,25,26]). The space–time formulation of the DG method results in a conservative numerical discretization on deforming meshes and it automatically satisfies the GCL. The

* Corresponding author.
E-mail addresses: srheberg@umn.edu (S. Rhebergen), cockburn@math.umn.edu (B. Cockburn), j.j.w.vandervegt@utwente.nl (J.J.W. van der Vegt).

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2012.08.052

method is implicit in time and so requires efficient solvers, e.g. multigrid methods [27,28]. In this article we follow the space–time DG approach.

Our first article in this series introduced the space–time DG formulation for the Oseen equations [26]. The scheme can be made arbitrarily high-order accurate in both space and time and is well suited for *hp*-adaptation. The method was shown to be unconditionally stable and continuity, coercivity and stability were proven. Furthermore a full *hp*-error analysis was presented.

In our second article, we introduced a space–time hybridizable DG (HDG) formulation for incompressible flows [22]. We showed that the HDG method displays optimal rates of convergence not only for the velocity and pressure fields, but also for the velocity gradient. This has not yet been achieved with DG methods. Another advantage of an HDG method over the DG method is that for higher order polynomial approximations, the computational cost of the HDG method is smaller than that of the DG method. As discussed in [22], the number of unknowns in the global system of the HDG method is of the order $\mathcal{O}(p^d)$ compared to $\mathcal{O}(p^{d+1})$ of the DG method, where p is the polynomial order and d is the dimension of the problem. It was also shown in [15] that for $p > 4$, the HDG method can be made more efficient than the continuous Galerkin method.

In this third article, we provide the Navier–Stokes extension of [26] resulting in a space–time DG method that can directly be applied to deforming domain problems. To solve the Navier–Stokes equations, we employ a Picard iteration technique. At every iteration, we therefore need to solve the Oseen equations. Solving the Oseen equations or the Navier–Stokes equations, results in the divergence-free velocity constraint being satisfied only weakly. While this is no problem for the Oseen equations [26], for the Navier–Stokes equations it means that stability of the scheme cannot be proven, unless a post-processing or modification of the weak formulation is applied. For the analysis of the Oseen equations of the space–time DG method, see [26]. The analysis of the DG method for the Oseen equations is given in [6] and for the Navier–Stokes equations in [7,8,10]. From numerical results in the literature, it appears that even though we cannot prove stability without modifying the weak formulation or applying a post-processing, the scheme is stable [3,22,23]. Our numerical results in Section 4 also show this. We will, however, also discuss two stabilization techniques. The first adds nonconservative terms to the weak formulation [11,12]. While this is an easy way of ensuring a stable scheme, local conservativity is lost. For the incompressible Navier–Stokes equations, this however might not be problematic as large gradients in the solution, which are associated with shocks and/or large discontinuities, are not present. We also introduce a space–time *BDM* projection to stabilize the method (see [7,10] for standard DG *BDM* projections). The *BDM* projection is such that if the divergence-free velocity constraint is satisfied weakly, the *BDM* projected velocity field is exactly divergence free on the element and its jump in the normal direction across element faces is zero. An advantage of this stabilization method is that local conservativity is preserved. The disadvantage, however, is that it is not trivial to obtain arbitrarily high order *BDM* spaces in higher dimensions. Let p_t and p_s denote the polynomial order in time and space, respectively. We consider here only 3 dimensional space–time *BDM* spaces for $p_t = p_s = 1$ and 2. Future work involves the expansion of space–time *BDM* spaces to more general situations, including $p_t \neq p_s$ and 4 dimensional space–time.

As mentioned above, we recently introduced a space–time HDG method for incompressible flows [22] with the idea of reducing the globally coupled degrees of freedom in order to obtain a more efficient DG method. In this article we test this statement by comparing the space–time HDG method with the natural extension of the space–time DG method for Oseen equations [26] to the Navier–Stokes equations. Comparisons between HDG and continuous Galerkin methods have been studied in [15], however, this is the first comparison between HDG and DG methods. The comparison is made for polynomial approximations in which $p_t = p_s + 1$ and $p_t = p_s$ with $p_s = 1$ and $p_s = 2$. We show that for these polynomial approximations, the space–time DG method is as efficient, if not more efficient, as the space–time HDG method in terms of L^2 errors of the velocity and pressure vs. the total number of degrees of freedom. In terms of CPU times and memory usage, the space–time HDG method seems slightly more efficient.

In [22] we found, for the space–time HDG method, that if we take the polynomial approximation of time equal to that of space, $p_t = p_s$, we were unable to obtain optimal rates of convergence for the pressure. Increasing the polynomial approximation in time to one order higher than that in space, $p_t = p_s + 1$, restored the optimal rates of convergence of the pressure. We show a similar behavior for the space–time DG method.

The outline of this article is as follows. We introduce the space–time formulation of the incompressible Navier–Stokes equations in Section 2. In Section 3 we introduce the space–time DG method for the Navier–Stokes equations, we discuss the stability of the method and introduce the space–time *BDM* projections. A numerical study in two dimensions on deforming domains will be conducted in Section 4. We will demonstrate convergence properties of the method, compare results of the different stabilization methods and compare results of the space–time DG and the space–time HDG method. For completeness, a short summary of the space–time HDG method for the Navier–Stokes equations is given in Appendix A. Concluding remarks are given in Section 5.

2. Incompressible Navier–Stokes equations

Denote by $\Omega_t \subset \mathbb{R}^d$ a bounded, time-dependent flow domain at time t in d spatial dimensions. Its boundary is denoted by $\partial\Omega_t$. Furthermore, let $\bar{x} = (x_1, x_2, \ldots, x_d)$ be the spatial variables. The time-dependent Navier–Stokes equations in Ω_t are given by

$$u_{i,t} + (u_i u_j)_{,j} - \nu(u_{i,j})_{,j} + (\delta_{ij} p)_{,j} = f_i, \quad u_{i,i} = 0, \quad \text{in } \Omega_t, \tag{1}$$

where $u \in \mathbb{R}^d$ is the velocity field, $p := p/\rho \in \mathbb{R}$ the kinematic pressure, ρ the fluid density, $\nu \in \mathbb{R}^+$ the kinematic viscosity, and $f \in \mathbb{R}^d$ a prescribed external body force. We use here index notation with the summation convention on repeated indices, the comma notation to denote partial differentiation and we let δ_{ij} denote the Kronecker delta function. Throughout this article, we will use $i, j = 1, 2, \ldots, d$ and $j = 0, 1, 2, \ldots, d$.

To solve the Navier–Stokes equations, we use a Picard iteration scheme [4,22] for which at each Picard iteration the linear Oseen problem has to be solved. For this, let the initial guess $u^{(0)}$ be a given convective divergence free velocity field and iterate over k:

$$u_{i,t}^{(k)} + (u_i^{(k)} u_j^{(k-1)})_{,j} - \nu(u_{i,j}^{(k)})_{,j} + (\delta_{ij} p)_{,j} = f_i, \quad u_{i,i}^{(k)} = 0, \quad \text{in } \Omega_t, \tag{2}$$

$(k = 1, 2, \ldots)$ until a certain convergence criterium has been met. In the following, for notational purposes, we take $u^{(k)} = u$ and $u^{(k-1)} = \bar{w}$.

Similar to what we did in [22], we consider the space–time DG method for the Oseen equation (2) noting that the Navier–Stokes equation (1) are solved by the Picard iteration. We consider the Oseen equations directly in a space–time domain $\mathcal{E} \subset \mathbb{R}^{d+1}$ in which the space–time position vector is given by $x = (x_0, \bar{x})$ with $x_0 = t$. Introducing $w = (1, \bar{w})$, the Oseen equation (2) can be written in the following space–time formulation:

$$(u_i w_j)_{,j} - \nu(u_{i,j})_{,j} + (\delta_{ij} p)_{,j} = f_i, \quad u_{i,i} = 0, \quad \text{in } \mathcal{E}. \tag{3}$$

For $0 < t < T$, the boundary of the space–time domain, $\partial\mathcal{E}$ is divided into disjoint sets, $\partial\mathcal{E} = \Gamma_m \cup \Gamma_p$, with Γ_m and Γ_p, respectively, inflow and outflow boundaries defined by:

$$\Gamma_m := \{x \in \partial\mathcal{E} : \ w_j n_j < 0\}, \quad \Gamma_p := \{x \in \partial\mathcal{E} : \ w_j n_j \geqslant 0\}.$$

Furthermore, $\Gamma_m = \Gamma_{Dm} \cup \Omega_0$, with Γ_{Dm} the part of Γ_m with a Dirichlet boundary and Ω_0 the part of Γ_m with the initial condition. We also subdivide $\Gamma_p = \Omega_T \cup \Gamma_{Dp} \cup \Gamma_N$, with Ω_T the part of $\partial\mathcal{E}$ at the final time T, Γ_{Dp} the part of Γ_p with a Dirichlet boundary condition and Γ_N the part of Γ_p with a Neumann boundary condition. Furthermore, $\Gamma_D = \Gamma_{Dm} \cup \Gamma_{Dp}$ the part of the space–time boundary with a Dirichlet boundary condition. These definitions correspond to those given in [26]. The boundary conditions are given as

$$u = u_0, \quad \text{on } \Omega_0, \qquad u = g^D, \quad \text{on } \Gamma_D, \qquad \nu \bar{n}_j u_{,j} - p = g^N, \quad \text{on } \Gamma_N, \tag{4}$$

with u_0 a given initial velocity field, g^D and g^N given data defined on (part of) the boundary. No boundary condition is imposed on Ω_T. If only Dirichlet boundary conditions are defined, we impose $\int_{\Omega_t} p \, d\bar{x} = 0$, to obtain a unique solution.

To obtain the DG weak formulation, it is necessary to consider the Oseen equations in its velocity–pressure-gradient formulation:

$$\sigma_{ij} = u_{i,j}, \quad \text{in } \mathcal{E}, \tag{5a}$$

$$(w_j u_i)_{,j} - \nu\sigma_{ij,j} + \delta_{ij} p_{,j} = f_i, \quad \text{in } \mathcal{E}, \tag{5b}$$

$$u_{i,i} = 0, \quad \text{in } \mathcal{E}. \tag{5c}$$

3. The space–time DG method

Divide the domain Ω_{t_n} into N_n non-overlapping spatial elements K^n, and similarly for the domain $\Omega_{t_{n+1}}$. A space–time element $\mathcal{K}^n$ can now be obtained by connecting K^n to K^{n+1} via linear interpolation in time. At curved boundaries, a higher order accurate interpolation may be necessary. The space–time domain $\mathcal{E}_h$, limited to the time interval (t_n, t_{n+1}), defines a space–time slab $\mathcal{E}_h^n$. The tessellation $\mathcal{T}_h^n$ of $\mathcal{E}_h^n$ consists of all space–time elements $\mathcal{K}^n$. The tessellation of $\mathcal{E}_h$ is given by $\mathcal{T}_h := \cup_n \mathcal{T}_h^n$. The element boundary $\partial\mathcal{K}^n$ of a space–time element $\mathcal{K}^n$ contains three parts K^n, K^{n+1} and $\mathcal{Q}^n = \partial\mathcal{K}^n \setminus (K^n \cup K^{n+1})$. The outward space–time normal vector on $\partial\mathcal{K}$ is denoted by $n = (n_0, \bar{n})$, with n_0 the temporal and $\bar{n}$ the spatial part of n.

Following the notation of [26], several sets of faces in the tessellation can be defined. In $\mathcal{E}$, denote the set of all faces by $\mathcal{F}$, the set of all interior faces by $\mathcal{F}_{int}$ and the set of all boundary faces on $\partial\mathcal{E}$ by $\mathcal{F}_{bnd}$. In the space–time slab $\mathcal{E}^n$, the set of all faces is denoted by $\mathcal{F}^n$, while the set of all interior faces is denoted by $\mathcal{S}_I^n$. Faces separating two space–time slabs are denoted by $\mathcal{S}_S^n$. On boundary faces, let $\mathcal{S}_D^n$ and $\mathcal{S}_N^n$ denote the sets of faces with, respectively, Dirichlet and Neumann boundary conditions. The set of Dirichlet faces can be subdivided as $\mathcal{S}_D^n = \mathcal{S}_{Dm}^n \cup \mathcal{S}_{Dp}^n$, with $\mathcal{S}_{Dm}^n$ and $\mathcal{S}_{Dp}^n$ the faces on Γ_m and Γ_p, respectively. Finally, $\mathcal{S}_{ID}^n = \mathcal{S}_I^n \cup \mathcal{S}_D^n$ and $\mathcal{S}_{IDN}^n = \mathcal{S}_I^n \cup \mathcal{S}_D^n \cup \mathcal{S}_N^n$.

We consider approximations of the velocity $u(x)$, pressure $p(x)$ and velocity gradient $\sigma(x)$, to lie in the finite element spaces $V_h^{(p_t,p_s)}, Q_h^{(p_t,p_s)}$ and $\Sigma_h^{(p_t,p_s)}$, respectively, which are defined as:

$$\begin{aligned}
V_h^{(p_t,p_s)} &:= \{v \in L^2(\mathcal{E})^d : v|_{\mathcal{K}} \circ F_{\mathcal{K}} \in (P^{(p_t,p_s)}(\hat{\mathcal{K}}))^d, \forall \mathcal{K} \in \mathcal{T}_h\}, \\
Q_h^{(p_t,p_s)} &:= \{q \in L^2(\mathcal{E}) : q|_{\mathcal{K}} \circ F_{\mathcal{K}} \in P^{(p_t,p_s)}(\hat{\mathcal{K}}), \forall \mathcal{K} \in \mathcal{T}_h\}, \\
\Sigma_h^{(p_t,p_s)} &:= \{\bar{\tau} \in L^2(\mathcal{E})^{d\times d} : \bar{\tau}|_{\mathcal{K}} \circ F_{\mathcal{K}} \in (P^{(p_t,p_s)}(\hat{\mathcal{K}}))^{d\times d}, \forall \mathcal{K} \in \mathcal{T}_h\}.
\end{aligned} \tag{6}$$

Here $F_{\mathcal{K}} : \hat{\mathcal{K}} \rightarrow \mathcal{K}$ is an iso-parametric mapping from $\hat{\mathcal{K}}$, an open unit hypercube in $\mathbb{R}^{d+1}$, to an element $\mathcal{K} \in \mathcal{T}_h$. Furthermore, $P^{(p_t,p_s)}(\hat{\mathcal{K}})$ denotes on $\hat{\mathcal{K}}$ the space of polynomials of degree at most p_t in the time direction and of degree at most p_s in the spatial coordinate direction. By $L^2(\Omega)$ we denote the space of square integrable functions on Ω.

The trace of a function $\omega(\mathcal{K}^L) \in W_h$, with W_h one of the finite element space defined in (6), on the boundary $\partial\mathcal{K}^L$ is defined as $\omega^L := \lim_{\varepsilon \downarrow 0} \omega(x - \varepsilon n^L)$, with n^L the unit outward space–time normal at $\partial\mathcal{K}^L$ [16].

The left and right traces of a function $\omega \in W_h$ at an internal face $\mathcal{S} \in \mathcal{S}^n$, with $\mathcal{S} = \partial\mathcal{K}^L \cap \partial\mathcal{K}^R$, are denoted by ω^L and ω^R, respectively. Because of the discontinuous function approximation, $\omega^L \neq \omega^R$, so that it will be useful to define the average operator: $\{\omega\} = \frac{1}{2}(\omega^L + \omega^R)$ and the jump operator: $[\![\omega]\!]_j = (\omega)^L \bar{n}_j^L + (\omega)^R \bar{n}_j^R$.

3.1. The space–time discontinuous Galerkin method

In the derivation of the space–time DG weak formulation, we will need definitions of the local and global lifting operators. For this, we follow the definitions given in [26]. The local lifting operator, $\mathcal{R}^{\mathcal{S}}$ is defined as:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \mathcal{R}_{ij}^{\mathcal{S}}(\vartheta_{ij})\tau_{ij}\,dx = \int_{\mathcal{S}} \vartheta_{ij}\{\tau_{ij}\}\,ds, \quad \forall\tau \in \Sigma_h^{(p_t,p_s)}, \quad \forall\mathcal{S} \in \mathcal{S}_{ID}^n. \tag{7}$$

The local lifting operator is zero outside the two elements connected to face $\mathcal{S}$. The global lifting operator, $\mathcal{R}$ is defined as:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \mathcal{R}_{ij}(\vartheta_{ij})\tau_{ij}\,dx = \sum_{\mathcal{S}\in\mathcal{S}_{ID}^n} \int_{\mathcal{K}} \mathcal{R}_{ij}^{\mathcal{S}}(\vartheta_{ij})\tau_{ij}\,dx, \quad \forall\tau \in \Sigma_h^{(p_t,p_s)}. \tag{8}$$

On Dirichlet boundary faces, $\mathcal{S} \in \cup_n \mathcal{S}_D^n$, let

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \mathcal{R}_{ij}^{\mathcal{S}}(\mathcal{P}g_i^D\bar{n}_j)\tau_{ij}\,dx = \int_{\mathcal{S}} g_i^D\bar{n}_j\tau_{ij}^L\,ds, \quad \forall\tau \in \Sigma_h^{(p_t,p_s)}, \quad \forall\mathcal{S} \in \mathcal{S}_D^n, \tag{9}$$

with $\mathcal{P}$ the L^2 projection on $\Sigma_h^{(p_t,p_s)}$. For the global lifting operator,

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \mathcal{R}_{ij}(\mathcal{P}g_i^D\bar{n}_j)\tau_{ij}\,dx = \sum_{\mathcal{S}\in\mathcal{S}_D^n} \int_{\mathcal{S}} g_i^D\bar{n}_j\tau_{ij}^L\,ds, \quad \forall\tau \in \Sigma_h^{(p_t,p_s)}. \tag{10}$$

Finally, let

$$\mathcal{R}_{ij}^{ID}(\vartheta_{ij}) = -\mathcal{R}_{ij}(\vartheta_{ij}) + \mathcal{R}_{ij}(\mathcal{P}g_i^D\bar{n}_j). \tag{11}$$

We will now derive the space–time DG weak formulation for the Oseen equations. For this we follow the framework described in [1,26].

The auxiliary variable. The space–time DG weak formulation for the auxiliary equation is obtained by multiplying (5a) with an arbitrary test function $\tau \in \Sigma_h^{(p_t,p_s)}$, integrating by parts in space twice over a space–time element $\mathcal{K} \in \mathcal{T}_h^n$ and summation over all elements $\mathcal{K} \in \mathcal{T}_h^n$ to obtain for all $\tau \in \Sigma_h^{(p_t,p_s)}$:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \sigma_{ij}\tau_{ij}\,dx = \sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \tau_{ij}u_{i,j}\,dx + \sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{Q}} \tau_{ij}^L(\hat{u}_i^\sigma - u_i^L)\bar{n}_j^L\,ds, \tag{12}$$

where we introduced the numerical flux $\hat{u}^\sigma$ after the first integration by parts. Note that since we integrated by parts only in space, we only have to consider the weak formulation in the space–time slab $\mathcal{E}^n$ since there are no fluxes between the different space–time slabs. It is more convenient to use integrals over the element faces $\mathcal{S}$ than element boundaries $\mathcal{Q}$. Following [1]:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{Q}} \tau_{ij}^L(\hat{u}_i^\sigma - u_i^L)\bar{n}_j^L\,ds = \sum_{\mathcal{S}\in\mathcal{S}_{IDN}^n} \int_{\mathcal{S}} \{\tau_{ij}\}[\![\hat{u}_i^\sigma - u_i]\!]_j\,ds + \sum_{\mathcal{S}\in\mathcal{S}_I^n} \int_{\mathcal{S}} \{\hat{u}_i^\sigma - u_i\}[\![\tau_{ij}]\!]_j\,ds. \tag{13}$$

For the numerical flux $\hat{u}^\sigma$, we make a similar choice as in [26]:

$$\hat{u}^\sigma = \{u\}, \quad \text{on } \mathcal{S}_I^n, \qquad \hat{u}^\sigma = g^D, \quad \text{on } \mathcal{S}_D^n, \qquad \hat{u}^\sigma = u^L, \quad \text{on } \mathcal{S}_N^n. \tag{14}$$

With this choice of the numerical flux, the weak formulation (12) is now equal to:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \sigma_{ij}\tau_{ij}\,dx = \sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \tau_{ij}u_{i,j}\,dx - \sum_{\mathcal{S}\in\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \{\tau_{ij}\}[\![u_i]\!]_j\,ds + \sum_{\mathcal{S}\in\mathcal{S}_D^n} \int_{\mathcal{S}} \tau_{ij}^L g_i^D\bar{n}_j^L\,ds. \tag{15}$$

To obtain an explicit expression for the auxiliary variable, use the definition of a global lifting operator $\mathcal{R}^{ID}$, (11) so see that:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \tau_{ij}\mathcal{R}_{ij}^{ID}([\![u_i]\!]_j)\,dx = -\sum_{\mathcal{S}\in\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \{\tau_{ij}\}[\![u_i]\!]_j\,ds + \sum_{\mathcal{S}\in\mathcal{S}_D^n} \int_{\mathcal{S}} \tau_{ij}^L g_i^D\bar{n}_j^L\,ds. \tag{16}$$

Introducing (16) into (15) and using the fact that this relation must be valid for arbitrary test functions τ, we can express $\sigma \in \Sigma_h^{(p_t,p_s)}$ as:

$$\sigma_{ij} = u_{i,j} + \mathcal{R}_{ij}^{ID}([\![u_i]\!]_j), \qquad \text{a.e. } \forall x \in \mathcal{E}^n. \tag{17}$$

The momentum equation. The space–time DG weak formulation for the momentum equation is obtained by multiplying (5b) with an arbitrary test function $v \in V_h^{(p_t,p_s)}$, integrating by parts over a space–time element $\mathcal{K} \in \mathcal{T}_h^n$ and summation over all elements $\mathcal{K} \in \mathcal{T}_h^n$ to obtain for all $v \in V_h^{(p_t,p_s)}$:

$$T_c + T_d + T_p = \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} v_i f_i \, dx, \tag{18}$$

in which

$$T_c := -\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} v_{i,j} w_j u_i \, dx + \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\partial\mathcal{K}} v_i^L \widehat{u_i^c w_j} n_j^L \, ds, \tag{19a}$$

$$T_d := \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} \nu v_{i,j} \sigma_{ij} \, dx - \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{Q}} v_i^L \nu \hat{\sigma}_{ij} \bar{n}_j^L \, ds, \tag{19b}$$

$$T_p := -\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} v_{i,j} \delta_{ij} p \, dx + \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{Q}} v_i^L \delta_{ij} \hat{p} \bar{n}_j^L \, ds, \tag{19c}$$

where $n^L = (n_0^L, \bar{n}^L)$ is the outward normal vector at $\partial\mathcal{K}$. The diffusive and pressure terms, T_d and T_p, respectively, are exactly the same as those presented in [26]. The convective term, T_c, however will need extra attention due to w not being pointwise divergence free. For $u_i w_j, p$ and σ, the numerical fluxes $\widehat{u_i^c w_j}$, $\hat{p}$ and $\hat{\sigma}$, respectively, are introduced to account for the multivalued traces at $\partial\mathcal{K}$. We next discuss the choice of numerical fluxes and the derivation of each term separately. For this we follow [26] for T_d and T_p. Afterwards we discuss the derivation of the convective term T_c.

Write the boundary integrals in (19b) and (19c) as face integrals [1] to find

$$\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{Q}} v_i^L(-\nu\hat{\sigma}_{ij} + \delta_{ij}\hat{p})\bar{n}_j^L \, ds = \sum_{\mathcal{S}\in\cup_n\mathcal{S}_{IDN}^n} \int_{\mathcal{S}} \{-\nu\hat{\sigma}_{ij} + \delta_{ij}\hat{p}\}[\![v_i]\!]_j \, ds + \sum_{\mathcal{S}\in\mathcal{S}_I^n} \int_{\mathcal{S}} \{v_i\}[\![-\nu\hat{\sigma}_{ij} + \delta_{ij}\hat{p}]\!]_j \, ds. \tag{20}$$

Following [26], the numerical fluxes for $\hat{\sigma}$ and $\hat{p}$ are chosen as:

$$\begin{aligned} &\hat{\sigma} = \{\sigma\} \quad \text{on } \mathcal{S}_I^n, \qquad \hat{\sigma} = \sigma^L \quad \text{on } \mathcal{S}_D^n, \qquad \hat{p} = \{p\} \quad \text{on } \mathcal{S}_I^n, \qquad \hat{p} = p^L \quad \text{on } \mathcal{S}_D^n, \\ &(\nu\hat{\sigma}_{ij} - \delta_{ij}\hat{p})\bar{n}_j^L = g_i^N, \quad \text{on } \mathcal{S}_N^n. \end{aligned} \tag{21}$$

Substitute (21) in (20) and replace σ by (17). The sum of the diffusion and pressure terms, $T_d + T_p$, then becomes:

$$\begin{aligned} T_d + T_p = &\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} \nu v_{i,j} u_{i,j} \, dx + \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} \nu v_{i,j} \mathcal{R}_{ij}^{ID}([\![u_i]\!]_j) \, dx - \sum_{\mathcal{S}\in\cup_n\mathcal{S}_N^n} \int_{\mathcal{S}} g_i^N v_i^L \, ds - \sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \nu\{u_{i,j}\}[\![v_i]\!]_j \, ds \\ &- \sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \nu\{\mathcal{R}_{ij}^{ID}([\![u_i]\!]_j)\}[\![v_i]\!]_j \, ds - \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} v_{i,i} p \, dx + \sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \{p\}[\![v_i]\!]_i \, ds. \end{aligned} \tag{22}$$

By (16), the second term on the right hand side of (22) is evaluated to

$$\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} \nu v_{i,j} \mathcal{R}_{ij}^{ID}([\![u_i]\!]_j) \, dx = -\sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \nu\{v_{i,j}\}[\![u_i]\!]_j \, ds + \sum_{\mathcal{S}\in\cup_n\mathcal{S}_D^n} \int_{\mathcal{S}} \nu v_{i,j}^L g_i^D \bar{n}_j^L \, ds. \tag{23}$$

The fifth term on the right hand side of (22), using (11), can be written as

$$\sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \nu\{\mathcal{R}_{ij}^{ID}([\![u_i]\!]_j)\}[\![v_i]\!]_j \, ds = -\sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \nu\{\mathcal{R}_{ij}([\![u_i]\!]_j)\}[\![v_i]\!]_j \, ds + \sum_{\mathcal{S}\in\cup_n\mathcal{S}_D^n} \int_{\mathcal{S}} \nu\mathcal{R}_{ij}(\mathcal{P}g_i^D\bar{n}_j) v_i^L \bar{n}_j^L \, ds. \tag{24}$$

To improve computational efficiency and memory use, the global lifting operator $\mathcal{R}$ is replaced by the local lifting operator $\mathcal{R}^{\mathcal{S}}$ using

$$\begin{aligned} &-\sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \nu\{\mathcal{R}_{ij}([\![u_i]\!]_j)\}[\![v_i]\!]_j \, ds + \sum_{\mathcal{S}\in\cup_n\mathcal{S}_D^n} \int_{\mathcal{S}} \nu\mathcal{R}_{ij}(\mathcal{P}g_i^D\bar{n}_j) v_i^L \bar{n}_j^L \, ds \\ &\approx -\sum_{\mathcal{S}\in\cup_n\mathcal{S}_{ID}^n} \int_{\mathcal{S}} \nu\eta\{\mathcal{R}_{ij}^{\mathcal{S}}([\![u_i]\!]_j)\}[\![v_i]\!]_j \, ds + \sum_{\mathcal{S}\in\cup_n\mathcal{S}_D^n} \int_{\mathcal{S}} \nu\eta\mathcal{R}_{ij}^{\mathcal{S}}(\mathcal{P}g_i^D\bar{n}_j) v_i^L \bar{n}_j^L \, ds, \end{aligned} \tag{25}$$

where η is a positive constant that has to be greater than the number of spatial faces of element $\mathcal{K}$ [26]. The final form of the sum $T_d + T_p$ can now be written as:

$$T_d + T_p = \sum_{K\in\mathcal{T}_h}\int_K \nu v_{i,j}u_{i,j}\,dx - \sum_{S\in\cup_n S^n_{ID}}\int_S \nu\{v_{i,j}\}[\![u_i]\!]_j\,ds + \sum_{S\in\cup_n S^n_D}\int_S \nu v^L_{i,j}g^D_i\bar{n}^L_j\,ds - \sum_{S\in\cup_n S^n_N}\int_S g^N_i v^L_i\,ds$$
$$- \sum_{S\in\cup_n S^n_{ID}}\int_S \nu\{u_{i,j}\}[\![v_i]\!]_j\,ds + \sum_{S\in\cup_n S^n_{ID}}\int_S \nu\eta\{\mathcal{R}^S_{ij}([\![u_i]\!]_j)\}[\![v_i]\!]_j\,ds - \sum_{S\in\cup_n S^n_D}\int_S \nu\eta\mathcal{R}^S_{ij}(\mathcal{P}g^D_i\bar{n}_{bj})v^L_i\bar{n}^L_j\,ds$$
$$- \sum_{K\in\mathcal{T}_h}\int_K v_{i,i}p\,dx + \sum_{S\in\cup_n S^n_{ID}}\int_S \{p\}[\![v_i]\!]_i\,ds. \tag{26}$$

Finally, we consider the convection term T_c, (19a). Writing the boundary integrals of (19a) as element face integrals [1]:

$$\sum_{K\in\mathcal{T}_h}\int_{\partial K} v^L_i\widehat{u^c_i w_j}n^L_j\,ds = \sum_{S\in\mathcal{F}}\int_S \{\widehat{u^c_i w_j}\}[\![v_i]\!]_j\,ds + \sum_{S\in\mathcal{F}_{int}}\int_S \{v_i\}[\![\widehat{u^c_i w_j}]\!]_j\,ds. \tag{27}$$

We define the convective flux $\widehat{u^c_i w_j}$ as an upwind flux:

$$\begin{aligned} &\widehat{u^c_i w_j} = \{u_i\}\{w_j\} + C^u[\![u_i]\!]_j, \quad \text{on } \mathcal{F}_{int}, \qquad \widehat{u^c_i w_j} = u_0, \quad \text{on } \Omega_0,\\ &\widehat{u^c_i w_j} = g^D_i w^L_j, \quad \text{on } \Gamma_{Dm}, \qquad \widehat{u^c_i w_j} = u^L_i w^L_j, \quad \text{on } \Gamma_p, \end{aligned} \tag{28}$$

where $C^u = \frac{1}{2}|\{w_j n_j\}|$. At the faces $K(t^+_n)$ and $K(t^-_{n+1})$ this means that:

$$\hat{u}^c = u^L \quad \text{at } K(t^-_{n+1}), \qquad \hat{u}^c = u^R \quad \text{at } K(t^+_n). \tag{29}$$

Substituting (28) in (27), the expression for T_c becomes

$$T_c = -\sum_{K\in\mathcal{T}_h}\int_K v_{i,j}w_j u_i\,dx + \sum_{S\in\mathcal{F}_{int}}\int_S (\{u_i\}\{w_j\} + C^u[\![u_i]\!]_j)[\![v_i]\!]_j\,ds + \sum_{S\subset\Gamma_p}\int_S u^L_i w^L_j n^L_j v^L_i\,ds + \sum_{S\subset\Gamma_{Dm}}\int_S g^D_i w^L_j n^L_j v^L_i\,ds$$
$$- \int_{\Omega_0}(u_0)_i v^L_i\,ds - \sum_{K\in\mathcal{T}_h}\int_K \frac{\alpha}{2}u_i v_i w_{j,j}\,dx + \sum_{S\in\mathcal{F}_{int}}\int_S \frac{\alpha}{2}\{u_i v_i\}[\![w_j]\!]_j\,ds, \tag{30}$$

where $\alpha = 0$ or $\alpha = 1$. The last two terms in (30) were added for stability reasons (see also [11,12]). We explain these terms and the choice of α in Section 3.2.

The space–time DG weak formulation for the momentum equations becomes: *Find* $u \in V_h^{(p_t,p_s)}$ *and* $p \in Q_h^{(p_t,p_s)}$, *such that for all* $v \in V_h^{(p_t,p_s)}$:

$$\mathcal{O}(u,v;w) + \mathcal{A}(u,v) + \mathcal{B}(p,v) = N(v;w) + F(v) + G(v), \tag{31}$$

in which

$$\mathcal{O}(u,v;w) := -\sum_{K\in\mathcal{T}_h}\int_K v_{i,j}w_j u_i\,dx + \sum_{S\in\mathcal{F}_{int}}\int_S (\{u_i\}\{w_j\} + C^u[\![u_i]\!]_j)[\![v_i]\!]_j\,ds + \sum_{S\subset\Gamma_p}\int_S u^L_i w^L_j n^L_j v^L_i\,ds$$
$$- \sum_{K\in\mathcal{T}_h}\int_K \frac{\alpha}{2}u_i v_i w_{j,j}\,dx + \sum_{S\in\mathcal{F}_{int}}\int_S \frac{\alpha}{2}\{u_i v_i\}[\![w_j]\!]_j\,ds, \tag{32}$$

$$\mathcal{A}(u,v) := \sum_{K\in\mathcal{T}_h}\int_K \nu v_{i,j}u_{i,j}\,dx - \sum_{S\in\cup_n S^n_{ID}}\int_S \nu\{v_{i,j}\}[\![u_i]\!]_j\,ds - \sum_{S\in\cup_n S^n_{ID}}\int_S \nu\{u_{i,j}\}[\![v_i]\!]_j\,ds + \sum_{S\in\cup_n S^n_{ID}}\int_S \nu\eta\{\mathcal{R}^S_{ij}([\![u_i]\!]_j)\}[\![v_i]\!]_j\,ds, \tag{33}$$

$$\mathcal{B}(p,v) := -\sum_{K\in\mathcal{T}_h}\int_K v_{i,i}p\,dx + \sum_{S\in\cup_n S^n_{ID}}\int_S \{p\}[\![v_i]\!]_i\,ds. \tag{34}$$

The linear forms are given by

$$N(v;w) := -\sum_{S\subset\Gamma_{Dm}}\int_S g^D_i w^L_j n^L_j v^L_i\,ds, \tag{35}$$

$$F(v) := -\sum_{S\in\cup_n S^n_D}\int_S \nu v^L_{i,j}g^D_i\bar{n}^L_j\,ds + \sum_{S\in\cup_n S^n_N}\int_S g^N_i v^L_i\,ds + \sum_{S\in\cup_n S^n_D}\int_S \nu\eta\mathcal{R}^S_{ij}(\mathcal{P}g^D_i\bar{n}_{bj})v^L_i\bar{n}^L_j\,ds + \int_{\Omega_0}(u_0)_i v^L_i\,ds, \tag{36}$$

$$G(v) := \sum_{K\in\mathcal{T}_h}\int_K v_i f_i\,dx. \tag{37}$$

From the above forms (32)–(37), only the convective form $\mathcal{O}(u,v;w)$ in (32) is different from that given in [26], which is due to w not being exactly point-wise divergence-free. All results in [26] regarding stability therefore hold for terms (33)–(37). In Section 3.2 we consider the stability of $\mathcal{O}(u,v;w)$ given by (32).

The continuity equation. The space–time DG weak formulation for the continuity equation is obtained by multiplying (5c) with an arbitrary test function $q \in Q_h^{(p_t,p_s)}$, integrating by parts in space over a space–time element $\mathcal{K} \in \mathcal{T}_h^n$ and summation over all elements $\mathcal{K} \in \mathcal{T}_h^n$ to obtain for all $q \in Q_h^{(p_t,p_s)}$:

$$-\sum_{\mathcal{K}\in\mathcal{T}_h^n}\int_{\mathcal{K}} q_{,\mathrm{j}}\, u_{\mathrm{j}}\, dx + \sum_{\mathcal{K}\in\mathcal{T}_h^n}\int_{\mathcal{Q}^n} q^L \hat{u}_{\mathrm{j}}^p \bar{n}_{\mathrm{j}}^L\, ds = 0, \tag{38}$$

with $\hat{u}^p$ the numerical flux that has to be introduced to account for multivalued traces on $\mathcal{Q}^n$. We write the boundary integral terms in (38) as face integral terms [1] so that we obtain:

$$-\sum_{\mathcal{K}\in\mathcal{T}_h^n}\int_{\mathcal{K}} q_{,\mathrm{j}}\, u_{\mathrm{j}}\, dx + \sum_{\mathcal{S}\in\mathcal{S}_{IDN}^n}\int_{\mathcal{S}} \{\hat{u}_{\mathrm{j}}^p\}[\![q]\!]_{\mathrm{j}}\, ds + \sum_{\mathcal{S}\in\mathcal{S}_I^n}\int_{\mathcal{S}} \{q\}[\![\hat{u}_{\mathrm{j}}^p]\!]_{\mathrm{j}}\, ds = 0, \tag{39}$$

The numerical flux $\hat{u}^p$ is taken as in [26]:

$$\hat{u}_{\mathrm{j}}^p = \{u_{\mathrm{j}}\} + \gamma[\![p]\!]_{\mathrm{j}} \quad \text{on } \mathcal{S}_I^n, \qquad \hat{u}^p = g^D \quad \text{on } \mathcal{S}_D^n, \qquad \hat{u}^p = u^L \quad \text{on } \mathcal{S}_N^n, \tag{40}$$

with $\gamma > 0$ a stabilizing term. Substituting the numerical flux (40) in (39) and integrating back by parts, we find that the weak formulation for the continuity equation is given by: *Find* $u \in V_h^{(p_t,p_s)}$ *such that for all* $q \in Q_h^{(p_t,p_s)}$:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n}\int_{\mathcal{K}} q u_{\mathrm{j},\mathrm{j}}\, dx - \sum_{\mathcal{S}\in\mathcal{S}_I^n}\{q\}[\![u_{\mathrm{j}}]\!]_{\mathrm{j}}\, ds + \sum_{\mathcal{S}\in\mathcal{S}_I^n}\int_{\mathcal{S}} \gamma[\![p]\!]_{\mathrm{j}}[\![q]\!]_{\mathrm{j}}\, ds + \sum_{\mathcal{S}\in\mathcal{S}_D^n}\int_{\mathcal{S}} (g_{\mathrm{j}}^D - u_{\mathrm{j}}^L)\bar{n}_{\mathrm{j}}^L q^L\, ds = 0,$$

or, in compact form taking all space–time slabs into account:

$$-\mathcal{B}(q,u) + \mathcal{C}(p,q) = H(q), \tag{41}$$

in which $\mathcal{B}$ is defined in (34) and

$$\mathcal{C}(p,q) := \sum_{\mathcal{S}\in\cup_n\mathcal{S}_I^n}\int_{\mathcal{S}} \gamma[\![p]\!]_i[\![q]\!]_i\, ds, \tag{42}$$

and

$$H(q) := -\sum_{\mathcal{S}\in\cup_n\mathcal{S}_D^n}\int_{\mathcal{S}} g_i^D \bar{n}_i^L q^L\, ds. \tag{43}$$

Space–time DG weak formulation. The space–time DG weak formulation can now be written as: *Find* $u \in V_h^{(p_t,p_s)}$ *and* $p \in Q_h^{(p_t,p_s)}$*, such that for all* $v \in V_h^{(p_t,p_s)}$ *and* $q \in Q_h^{(p_t,p_s)}$:

$$\begin{aligned} &\mathcal{O}(u,v;w) + \mathcal{A}(u,v) + \mathcal{B}(p,v) = N(v;w) + F(v) + G(v),\\ &-\mathcal{B}(q,u) + \mathcal{C}(p,q) = H(q). \end{aligned} \tag{44}$$

3.2. Stability

We now determine the stability for the trilinear form $\mathcal{O}(u,v;w)$, as given by (32). As mentioned in Section 3.1, all other terms in the space–time DG formulation (44) are the same as those given in [26] so that we do not consider these here again.

3.2.1. Controlling the energy

Consider the trilinear form $\mathcal{O}(u,v;w)$, (32), and replace v by u to find:

$$\begin{aligned} \mathcal{O}(u,u;w) := &-\sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\mathcal{K}} u_{i,j} w_j u_i\, dx + \sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}} (\{u_i\}\{w_j\} + C^u[\![u_i]\!]_j)[\![u_i]\!]_j\, ds + \sum_{\mathcal{S}\subset\Gamma_p}\int_{\mathcal{S}} u_i^L w_j^L n_j^L u_i^L\, ds\\ &- \sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\mathcal{K}} \frac{\alpha}{2} u_i u_i w_{j,j}\, dx + \sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}} \frac{\alpha}{2}\{u_i u_i\}[\![w_j]\!]_j\, ds. \end{aligned}$$

Using $u_{i,j}w_j u_i = -\frac{1}{2}u_i u_i w_{j,j} + \frac{1}{2}(u_i u_i w_j)_{,j}$ we obtain:

$$\begin{aligned} \mathcal{O}(u,u;w) := &\frac{1}{2}\sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\mathcal{K}} u_i u_i w_{j,j}\, dx + \sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}} \{u_i\}\{w_j\}[\![u_i]\!]_j\, ds + \sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}} C^u[\![u_i]\!]_j[\![u_i]\!]_j\, ds - \frac{1}{2}\sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\partial\mathcal{K}} u_i^L u_i^L w_j^L n_j^L\, ds\\ &+ \sum_{\mathcal{S}\subset\Gamma_p}\int_{\mathcal{S}} u_i^L w_j^L n_j^L u_i^L\, ds - \sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\mathcal{K}} \frac{\alpha}{2} u_i u_i w_{j,j}\, dx + \sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}} \frac{\alpha}{2}\{u_i u_i\}[\![w_j]\!]_j\, ds. \end{aligned} \tag{45}$$

Consider now the following equalities

$$-\frac{1}{2}\sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\partial\mathcal{K}} u_i^L u_i^L w_j^L n_j^L\,ds = -\frac{1}{2}\sum_{\mathcal{S}\in\mathcal{F}}\int_{\mathcal{S}}[\![u_i]\!]_j\{u_i w_j\}\,ds - \frac{1}{2}\sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}}\{u_i\}[\![u_i w_j]\!]_j\,ds = -\frac{1}{2}\sum_{\mathcal{S}\subset\Gamma_m}\int_{\mathcal{S}} u_i^L u_i^L w_j^L n_j^L\,ds$$
$$-\frac{1}{2}\sum_{\mathcal{S}\subset\Gamma_p}\int_{\mathcal{S}} u_i^L u_i^L w_j^L n_j^L\,ds - \frac{1}{2}\sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}}[\![u_i]\!]_j\{u_i w_j\} + \{u_i\}[\![u_i w_j]\!]_j\,ds + \frac{1}{2}\int_{\Omega_0} u_i^L u_i^L\,ds,$$

and

$$\{u_i w_j\} = \{u_i\}\{w_j\} + \frac{1}{4}[\![u_i]\!]_k[\![w_j]\!]_k, \quad [\![u_i w_j]\!]_j = \{u_i\}[\![w_j]\!]_j + [\![u_i]\!]_j\{w_j\}.$$

The trilinear form $\mathcal{O}(u,u;w)$ can be written as:

$$\mathcal{O}(u,u;w) := \frac{1}{2}\sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\mathcal{K}} u_i u_i w_{j,j}\,dx - \sum_{\mathcal{K}\in\mathcal{T}_h}\int_{\mathcal{K}}\frac{\alpha}{2}u_i u_i w_{j,j}\,dx - \frac{1}{2}\sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}}\{u_i u_i\}[\![w_j]\!]_j\,ds + \sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}}\frac{\alpha}{2}\{u_i u_i\}[\![w_j]\!]_j\,ds$$
$$-\frac{1}{2}\sum_{\mathcal{S}\subset\Gamma_m}\int_{\mathcal{S}} u_i^L u_i^L w_j^L n_j^L\,ds + \frac{1}{2}\sum_{\mathcal{S}\subset\Gamma_p}\int_{\mathcal{S}} u_i^L u_i^L w_j^L n_j^L\,ds + \sum_{\mathcal{S}\in\mathcal{F}_{int}}\int_{\mathcal{S}} C^u[\![u_i]\!]_j[\![u_i]\!]_j\,ds + \frac{1}{2}\int_{\Omega_0} u_i^L u_i^L\,ds, \tag{46}$$

where we used $\{u_i u_i\} = \{u_i\}\{u_i\} + \frac{1}{4}[\![u_i]\!]_k[\![u_i]\!]_k$ and $[\![u_i]\!]_j[\![u_i]\!]_k[\![w_j]\!]_k = [\![u_i]\!]_k[\![u_i]\!]_k[\![w_j]\!]_j$. We do not have control over the first four terms on the right hand side in (46). Therefore, one possibility of ensuring stability of the scheme, is to set $\alpha = 1$ and thereby eliminating these terms. By setting $\alpha = 1$, however, we give up local conservation, but gain control over the energy. The property of local conservation is important in flows in which shocks and/or large discontinuities in the solution emerge. For the incompressible Navier–Stokes equations, such solutions do not appear so that setting $\alpha = 1$ can be a justified technique for controlling the energy. We will call this the *nonconservative stabilization*. See also [11,12] for successful application of this technique in the case of standard DG.

A second possibility is to set $\alpha = 0$ and force the first and third terms on the right hand side in (46) to be zero. This can be done by requiring:

$$w_{j,j} = 0, \quad \forall\mathcal{K}\in\mathcal{T}_h, \qquad \text{and} \qquad [\![w_j]\!]_j = 0, \quad \forall\mathcal{S}\in\mathcal{F}_{int}. \tag{47}$$

This approach maintains the local conservation of the scheme and it allows control over the energy. Obtaining requirements (47) can be done by a *BDM*-projection [5], which we discuss in Section 3.2.2. In the following we denote this form of stabilization as the *BDM stabilization*.

In Section 4.1 we discuss numerical results of the *nonconservative* and *BDM stabilization* and compare these results to those obtained when no stabilization ($\alpha = 0$ and no *BDM*-projection) is used.

3.2.2. An equal order in space and time BDM-projection

As seen in Section 3.2.1, one option to obtain a stable Picard-iteration (2) is to require that the convective velocity field $w\in\mathbb{R}^{d+1}$ be exactly divergence free. Solving the space–time DG discretization (44) results in a velocity u that is not exactly divergence-free. For this reason we post-process, element-wise, the velocity $\mathrm{u} = (1,u)|_{\mathcal{K}}$ to obtain $\mathbb{P}_{\mathcal{K}}\mathrm{u}$ that satisfies (47) and set $w|_{\mathcal{K}} = \mathbb{P}_{\mathcal{K}}\mathrm{u}$, which is to be used in the next Picard-iteration. This post-processing is slightly different from that given in [7,10] since we consider here space–time hexahedron elements. We consider equal order approximations in space and time, $p_t = p_s = k$ for $d = 2$ (which requires a 3-dimensional space–time *BDM*-projection). Future work will consider *BDM*-projections in which $p_t \neq p_s$ and 4-dimensional *BDM*-projections for $d = 3$ problems.

For $\mathrm{u}\in(P^{(k,k)}(\widehat{\mathcal{K}}))^3$ we consider projections onto $(BDM^k(\widehat{\mathcal{K}}))^3$ [5]. The BDM^k space is given by

$$(BDM^k)^3 := (P^k(\widehat{\mathcal{K}}))^3 \oplus_{\ell=0}^k \mathrm{curl}_\xi(0,0,\xi_0\xi_1^{\ell+1}\xi_2^{k-\ell}) \oplus_{\ell=0}^k \mathrm{curl}_\xi(\xi_0^{k-\ell}\xi_1\xi_2^{\ell+1},0,0) \oplus_{\ell=0}^k \mathrm{curl}_\xi(0,\xi_0^{\ell+1}\xi_1^{k-\ell}\xi_2,0), \quad (k\geqslant 1),$$

where

$$\mathrm{curl}_\xi(0,0,\xi_0\xi_1^{\ell+1}\xi_2^{k-\ell}) = \left((\ell+1)\xi_0\xi_1^{\ell}\xi_2^{k-\ell},\ -\xi_1^{\ell+1}\xi_2^{k-\ell},\ 0\right)^T,$$
$$\mathrm{curl}_\xi(\xi_0^{k-\ell}\xi_1\xi_2^{\ell+1},0,0) = \left(0,\ (\ell+1)\xi_0^{k-\ell}\xi_1\xi_2^{\ell},\ -\xi_0^{k-\ell}\xi_2^{\ell+1}\right)^T,$$
$$\mathrm{curl}_\xi(0,\xi_0^{\ell+1}\xi_1^{k-\ell}\xi_2,0) = \left(-\xi_0^{\ell+1}\xi_1^{k-\ell},\ 0,\ (\ell+1)\xi_0^{\ell}\xi_1^{k-\ell}\xi_2\right)^T.$$

Here ξ_0, ξ_1 and ξ_2 are the local coordinates on the space–time reference element $\widehat{\mathcal{K}}$. Let $u\in V_h^{(k,k)}$ be computed from the discretization of (44) and let $\mathrm{u} = (1,u)$. We now define an element-wise post-processing projection $\mathbb{P}_{\mathcal{K}}$ and set $w|_{\mathcal{K}} = \mathbb{P}_{\mathcal{K}}\mathrm{u}$. Important to note is that we require the projected velocity to be in the space $w = \mathbb{P}_{\mathcal{K}}\mathrm{u}\in BDM^k$. The local post-processing is defined by

$$\int_{\partial\mathcal{K}} w_j^L n_j^L\varphi\,ds = \int_{\partial\mathcal{K}}\hat{\mathrm{u}}_j^p n_j^L\varphi\,ds, \quad \forall\varphi\in P^{(k,k)}(\widehat{\partial\mathcal{K}}), \tag{48a}$$

$$\int_{\mathcal{K}} w_j\phi_j\,dx = \int_{\mathcal{K}}\mathrm{u}_j\phi_j\,dx, \quad \forall\phi\in(P^{(k-2,k-2)}(\widehat{\mathcal{K}}))^3\ (k\geqslant 2), \tag{48b}$$

(requirement (48b) can be dropped for $k < 2$). Now, note the following. For all $q \in P^{(k-1,k-1)}$:

$$\sum_{K\in\mathcal{T}_h^n}\int_K w_{j,j}q\,dx = \sum_{K\in\mathcal{T}_h^n}\left(-\int_K w_j q_{,j}\,dx + \int_{\partial K} w_j n_j^L q^L\,ds\right) = \sum_{K\in\mathcal{T}_h^n}\left(-\int_K \mathrm{u}_j q_{,j}\,dx + \int_{\partial K}\hat{\mathrm{u}}_j^p n_j^L q^L\,ds\right)$$
$$= \sum_{K\in\mathcal{T}_h^n}\left(-\int_K \mathrm{u}_0 q_{,0}\,dx + \int_{\partial K}\hat{u}_0^p n_0^L q^L\,ds - \int_K u_j q_{,j}\,dx + \int_Q \hat{u}_j^p \bar{n}_j^L q^L\,ds\right) = \sum_{K\in\mathcal{T}_h^n}\left(-\int_K q_{,0}\,dx + \int_{\partial K} n_0^L q^L\,ds\right) = 0,$$

where the second equality holds by (48), the fourth equality holds by (38) for all $q \in P^{(k,k)}$, and so in particular also for all $q \in P^{(k-1,k-1)} \subset P^{(k,k)}$ and since $\mathrm{u}_0 = 1$, and the fifth equality follows directly by integration by parts of the volume integral. The above spaces have the following dimensions:

$$\dim\left(BDM^k(\hat{K})\right)^3 = \frac{(k+1)(k+2)(k+3)}{2} + 3(k+1),$$
$$\dim\left(P^{(k,k)}(\hat{K})\right)^3 = \frac{(k+1)(k+2)(k+3)}{2},$$
$$\dim P^{(k,k)}(\widehat{\partial K}) = \frac{1}{2}(k+1)(k+2),$$

so that we have the following equalities between the number of equations and unknowns:

$$\dim\left(BDM^k(\hat{K})\right)^3 = \dim\left(P^{(k-2,k-2)}(\hat{K})\right)^3 + 6\dim P^{(k,k)}(\widehat{\partial K}) = \frac{(k+1)(k^2+5k+12)}{2}.$$

where the 6 is due to the 6 faces of ∂K.

Piola transform. For $w = \mathbb{P}_K \mathrm{u} \in BDM^k$, the *BDM* projection is defined on the reference element. This means that on the reference element the velocity-field w is divergence-free and that the velocity jumps in the normal direction are zero. To maintain these properties on the physical element, the Piola transformation is required. The Piola transformation P_K for a vector $v \in BDM^k$ is defined as

$$v(x) = v(F_K(\xi)) = \mathrm{P}_K\hat{v}(\xi) = |DF_K(\xi)|^{-1}DF_K(\xi)\hat{v}(\xi), \tag{49}$$

where DF_K is the Jacobian of the iso-parametric mapping $F_K : \hat{K} \to K$ and $|DF_K|$ its Jacobian (see [5, p. 97]). Note that in (48a) and (48b) the Piola transform is only applied to $w \in BDM^k$.

4. Numerical test cases

We apply now the space–time DG method to different test cases. We will determine rates of convergence, compare the different stabilization methods and give a comparison between results from a space–time DG and HDG method. We recently introduced the space–time HDG method in [22]. A short summary of the method is given in Appendix A.

In the Picard iteration scheme (2), used to solve the Navier–Stokes equations, we consider the scheme to be converged if the residual of the discrete system drops nine orders in magnitude compared to the residual computed in the first Picard iteration of each time step.

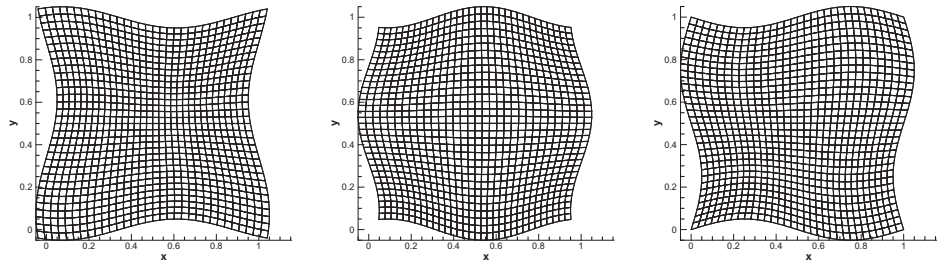
All test cases in this section are implemented using the software package hpGEM [21]. The direct LU solver available in PETSc [2] is used to solve the global linear system arising from the discretization. All integrals of the weak formulation, also those at deformed surfaces, are computed by the same technique. The integrals are mapped from physical space to a reference space (an open unit hypercube in $\mathbb{R}^{d+1}$, with d the spatial dimension). On this reference space we use a standard Gauss-integration technique. The Gauss points and weights are chosen depending on the numerical order of the (H)DG-scheme so that no numerical errors are introduced by the integration method.

4.1. Stabilization methods

In this section we will discuss the use of *BDM stabilization*, *nonconservative stabilization* and *No stabilization*. For this, we use a test case proposed in [22]. Consider the Navier–Stokes equations on a deforming space–time domain $\mathcal{E}$. The deformation of the space–time domain is based on the following transformation of a uniform space–time slab $(t, t+\Delta t)\times(0,1)^2$ for $t \geqslant 0$, $t + \Delta t \leqslant T = 1$:

$$x_i = x_i^u + A\left(\frac{1}{2} - x_i^u\right)\sin\left(2\pi\left(\frac{1}{2} - x_*^u + t^*\right)\right),$$
$$t^* = \begin{cases} t & \text{if } x_0^u = t \\ t+\Delta t & \text{if } x_0^u = t+\Delta t \end{cases}, \quad x_*^u = \begin{cases} x_2 & \text{if } i = 1 \\ x_1 & \text{if } i = 2 \end{cases},$$

where A is the amplitude. We set $A = 0.1$. On inflow boundaries we consider Dirichlet boundary conditions while Neumann boundary conditions are set on outflow boundaries. The source terms $f_i(x)$ and the boundary data on Γ_D and Γ_N are such that the exact solution is given by:



(a) Domain at $t = 0.35$. (b) Domain at $t = 0.80$. (c) Domain at $t = 1.0$.

Fig. 1. Snapshots of the deforming domain and a 32×32 mesh for test case 4.1.

Table 1
History of convergence for test case 4.1: the unsteady Navier–Stokes equations ($\nu = 0.01$) with exact solution (50) using a linear polynomial approximation at $T = 1$. The numerical parameters are $\eta = 5$ and $\gamma = 1$.

N	$\lvert u_1 - u_1^h\rvert_{L^2(\Omega)}$		$\lvert u_2 - u_2^h\rvert_{L^2(\Omega)}$		$\lvert p - p_h\rvert_{L^2(\Omega)}$		$\lvert u_1 - w_1^h\rvert_{L^2(\Omega)}$		$\lvert u_2 - w_2^h\rvert_{L^2(\Omega)}$		Its.
	Error	Order	Error	Order	Error	Order	Error	Order	Error	Order	
No stab.											
8	8.86e−3	–	6.12e−3	–	4.44e−2	–	–	–	–	–	19.0
16	1.57e−3	2.5	1.29e−3	2.2	8.72e−3	2.3	–	–	–	–	16.3
32	5.07e−4	1.6	3.74e−4	1.8	6.77e−3	0.4	–	–	–	–	12.3
64	8.55e−5	2.6	6.74e−5	2.5	1.82e−3	1.9	–	–	–	–	9.2
Noncons. stab.											
8	8.72e−3	–	5.95e−3	–	4.29e−2	–	–	–	–	–	10.7
16	1.58e−3	2.5	1.23e−3	2.3	8.14e−3	2.4	–	–	–	–	9.3
32	5.07e−4	1.6	3.72e−4	1.7	6.73e−3	0.3	–	–	–	–	7.7
64	8.59e−5	2.6	6.68e−5	2.5	1.87e−3	1.8	–	–	–	–	6.3
BDM stab.											
8	8.78e−3	–	5.89e−3	–	4.06e−2	–	1.06e−2	–	7.12e−3	–	7.3
16	1.60e−3	2.5	1.20e−3	2.3	7.53e−3	2.4	1.81e−3	2.6	1.17e−3	2.6	6.6
32	5.09e−4	1.7	3.70e−4	1.7	6.66e−3	0.2	6.17e−4	1.6	4.14e−4	1.5	6.0
64	8.64e−5	2.6	6.65e−5	2.5	1.94e−3	1.8	9.44e−5	2.7	5.89e−5	2.8	5.3

Table 2
History of convergence for test case 4.1: the unsteady Navier–Stokes equations ($\nu = 0.01$) with exact solution (50) using a quadratic polynomial approximation at $T = 1$. The numerical parameters are $\eta = 5$ and $\gamma = 1$.

N	$\lvert u_1 - u_1^h\rvert_{L^2(\Omega)}$		$\lvert u_2 - u_2^h\rvert_{L^2(\Omega)}$		$\lvert p - p_h\rvert_{L^2(\Omega)}$		$\lvert u_1 - w_1^h\rvert_{L^2(\Omega)}$		$\lvert u_2 - w_2^h\rvert_{L^2(\Omega)}$		Its.
	Error	Order	Error	Order	Error	Order	Error	Order	Error	Order	
No stab.											
8	7.34e−4	–	5.26e−4	–	1.46e−3	–	–	–	–	–	14.7
16	6.11e−5	3.6	4.36e−5	3.6	9.63e−4	0.6	–	–	–	–	11.7
32	1.17e−5	2.4	8.88e−6	2.3	1.12e−4	3.1	–	–	–	–	8.4
64	7.83e−7	3.9	6.71e−7	3.7	7.36e−5	0.6	–	–	–	–	6.2
Noncons. stab.											
8	7.18e−4	–	5.21e−4	–	1.32e−3	–	–	–	–	–	9.0
16	5.95e−5	3.6	4.30e−5	3.6	8.97e−4	0.6	–	–	–	–	7.4
32	1.16e−5	2.4	8.85e−6	2.3	1.11e−4	3.0	–	–	–	–	6.0
64	7.76e−7	3.9	6.75e−7	3.7	7.42e−5	0.6	–	–	–	–	4.8
BDM stab.											
8	7.10e−4	–	5.18e−4	–	1.22e−3	–	1.22e−3	–	8.34e−4	–	6.8
16	5.85e−5	3.6	4.25e−5	3.6	8.62e−4	0.5	8.97e−5	3.8	5.90e−5	3.8	6.2
32	1.16e−5	2.3	8.84e−6	2.3	1.13e−4	2.9	1.93e−5	2.2	1.34e−5	2.1	5.5
64	7.74e−7	3.9	6.84e−7	3.7	7.64e−5	0.6	1.19e−6	4.0	8.46e−7	4.0	5.0

$$
\begin{aligned}
u_1(x,y,t) &= -\exp(x)(y\cos(y) + \sin(y))\sin(t),\\
u_2(x,y,t) &= \exp(x) y \sin(y)\sin(t),\\
p(x,y,t) &= \exp(x)\cos(y)\cos(t) + (1-\exp(1))\sin(1)\cos(t).
\end{aligned}
\tag{50}
$$

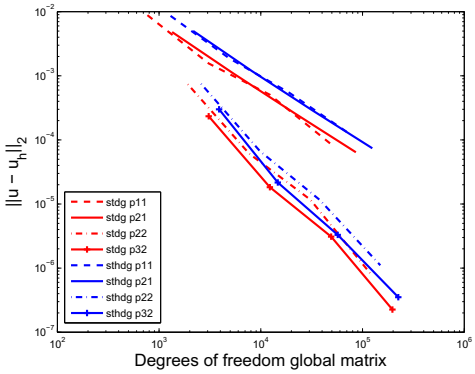
Table 3
History of convergence of the Navier–Stokes equations ($\nu = 0.01$) at $T = 1$. Computed using space–time DG ($\eta = 5, \gamma = 1$) and space–time HDG ($\delta = 0.001$). Top half $p_t = p_s = 1$. Bottom half: $p_t = p_s + 1 = 2$.

(p_t, p_s)	N	$\|u_1 - u_1^h\|_{L^2(\Omega)}$		$\|u_2 - u_2^h\|_{L^2(\Omega)}$		$\|p - p_h\|_{L^2(\Omega)}$		Condition number
		Error	Order	Error	Order	Error	Order	
DG								
(1,1)	8	8.86e−3	–	6.12e−3	–	4.44e−2	–	1.59e+3
	16	1.57e−3	2.5	1.29e−3	2.2	8.72e−3	2.3	6.34e+3
	32	5.07e−4	1.6	3.74e−4	1.8	6.77e−3	0.4	3.21e+4
	64	8.55e−5	2.6	6.74e−5	2.5	1.82e−3	1.9	2.03e+5
HDG								
(1,1)	8	8.62e−3	–	7.71e−3	–	2.05e−2	–	1.86e+4
	16	1.88e−3	2.2	1.73e−3	2.2	8.09e−3	1.3	7.85e+4
	32	5.46e−4	1.8	4.94e−4	1.8	5.56e−3	0.5	8.91e+5
	64	1.24e−4	2.1	1.19e−4	2.1	7.02e−4	3.0	7.73e+6
DG								
(2,1)	8	4.82e−3	–	3.05e−3	–	6.11e−3	–	9.08e+3
	16	1.09e−3	2.1	6.90e−4	2.1	1.57e−3	2.0	3.56e+4
	32	2.61e−4	2.1	1.68e−4	2.0	1.92e−4	3.0	1.79e+5
	64	6.42e−5	2.0	4.17e−5	2.0	1.05e−4	0.9	1.09e+6
HDG								
(2,1)	8	5.03e−3	–	4.57e−3	–	1.46e−2	–	5.46e+4
	16	1.19e−3	2.1	1.18e−3	2.0	1.98e−3	2.9	4.23e+5
	32	2.94e−4	2.0	3.02e−4	2.0	3.22e−4	2.6	1.72e+6
	64	7.43e−5	2.0	8.06e−5	1.9	1.04e−4	1.6	1.58e+9

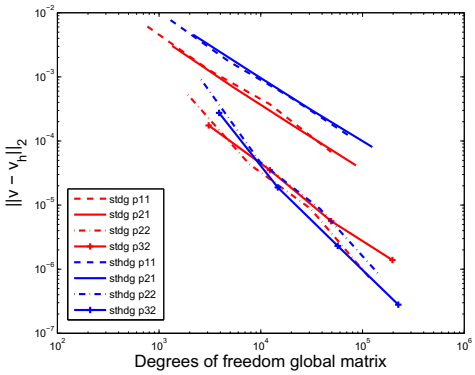
Table 4
History of convergence e of the Navier–Stokes equations ($\nu = 0.01$) at $T = 1$. Computed using space–time DG ($\eta = 5, \gamma = 1$) and space–time HDG ($\delta = 0.001$). Top half $p_t = p_s = 2$. Bottom half: $p_t = p_s + 1 = 3$. * Due to large memory requirements, we limited the computation of this value to a grid of 50×50 elements.

(p_t, p_s)	N	$\|u_1 - u_1^h\|_{L^2(\Omega)}$		$\|u_2 - u_2^h\|_{L^2(\Omega)}$		$\|p - p_h\|_{L^2(\Omega)}$		Condition number
		Error	Order	Error	Order	Error	Order	
DG								
(2,2)	8	7.34e−4	–	5.26e−4	–	1.46e−3	–	2.31e+4
	16	6.11e−5	3.6	4.36e−5	3.6	9.63e−4	0.6	9.98e+4
	32	1.17e−5	2.4	8.88e−6	2.3	1.12e−4	3.1	5.38e+5
	64	7.83e−7	3.9	6.71e−7	3.7	7.36e−5	0.6	3.38e+6
HDG								
(2,2)	8	7.50e−4	–	9.11e−4	–	7.58e−3	–	8.81e+4
	16	6.69e−5	3.5	4.74e−5	4.3	4.44e−4	4.1	2.38e+5
	32	1.16e−5	2.5	8.84e−6	2.4	6.76e−5	2.7	4.86e+6
	64	1.11e−6	3.4	7.87e−7	3.5	5.46e−5	0.3	1.44e+7
DG								
(3,2)	8	2.34e−4	–	1.75e−4	–	2.82e−4	–	1.15e+5
	16	1.83e−5	3.7	1.25e−5	3.8	3.53e−5	3.0	5.00e+5
	32	3.10e−6	2.6	2.30e−6	2.4	5.63e−6	2.6	2.69e+6
	64	2.26e−7	3.8	1.50e−7	3.9	1.30e−6	2.1	1.64e+7
HDG								
(3,2)	8	3.00e−4	–	2.75e−4	–	1.64e−3	–	1.92e+6
	16	2.17e−5	3.8	1.89e−5	3.9	8.53e−5	4.3	5.13e+5
	32	3.31e−6	2.7	2.31e−6	3.0	3.45e−6	4.6	3.23e+6
	64	3.53e−7	3.2	2.80e−7	3.0	3.96e−7	3.1	4.90e+6*

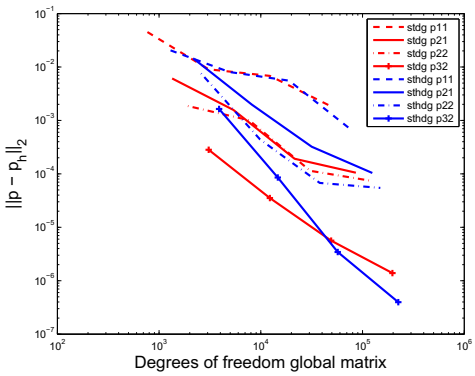
We consider a grid with $N \times N$ elements and time step $\Delta t = \sqrt{2}/N$, where N is $N = 8, 16, 32, 64$. We take $\nu = 0.01$, $\eta = 5$ and $\gamma = 1$. Fig. 1 shows the deformation of the domain and mesh with $N = 32$. Tables 1 and 2 show the rates of convergence for the velocities and pressure and, if the *BDM stabilization* is applied, also the rates of convergence of the *BDM*-projected velocity. In the last column of Tables 1 and 2, the average amount of Picard-iterations per time step needed for 9 orders of convergence of the residual, are given. For the velocity field, expected rates of convergence are obtained. The pressure, however, shows suboptimal convergence rates, as was noted also in [22]. In [22] we showed that proper rates of convergence for the pressure can be achieved if we take the polynomial approximation in time 1 order higher than the polynomial approximation in space. We will discuss this issue further in Section 4.2. Here, however, we consider only equal order polynomial approximations in space and time to make the comparison among *BDM stabilization*, *nonconservative stabilization* and *No stabilization*. It can clearly be seen from Tables 1 and 2 that using *BDM stabilization* or *nonconservative stabilization*, has a



(a) L^2 error of the x-velocity vs. D.O.F..



(b) L^2 error of the y-velocity vs. D.O.F..



(c) L^2 error of the pressure vs. D.O.F..

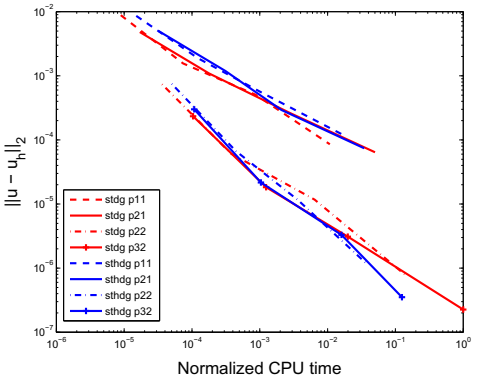
Fig. 2. The L^2-errors of the velocities and pressure against the total number of degrees of freedom of the global system.

significant positive effect on reducing the number of Picard iterations per time step compared to not using any stabilization for coarse meshes. However, as the mesh is refined, the difference in the number of Picard iterations per time step needed for convergence between the stabilization methods decreases. The errors in the L^2-norm and convergence rates using *BDM stabilization*, *nonconservative stabilization* or *No stabilization* are not significantly different.

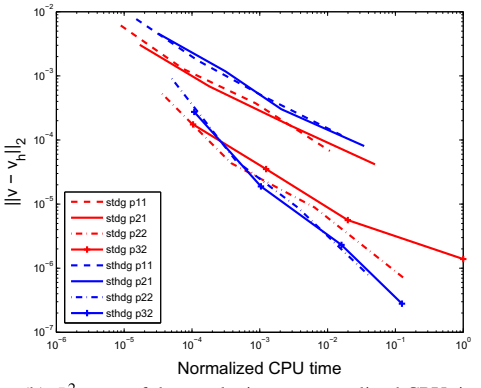
4.2. A numerical comparison between space–time DG and HDG

In this section we numerically compare the space–time DG method with the space–time HDG method. We consider the same test case of Section 4.1. We consider a grid with $N \times N$ elements and time step $\Delta t = \sqrt{2}/N$, where N is $N = 8, 16, 32, 64$. We take $\nu = 0.01$. In the space–time DG method, we set $\eta = 5$ and $\gamma = 1$. In the HDG method we set $\delta = 0.001$.

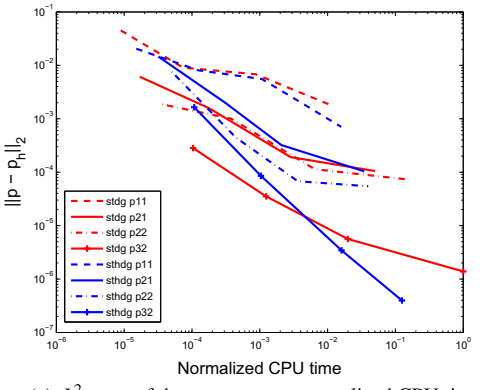
In Tables 3 and 4 rates of convergence for the velocities and pressure are given for the space–time DG and HDG methods. Both space–time DG and HDG computations show irregular behavior in the convergence rates of the pressure when $p_t = p_s$.



(a) L^2 error of the x-velocity vs. normalized CPU time.



(b) L^2 error of the y-velocity vs. normalized CPU time.

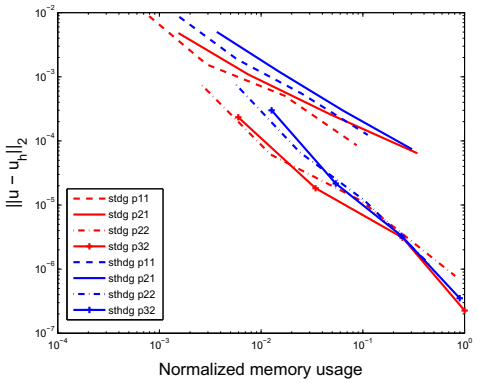


(c) L^2 error of the pressure vs. normalized CPU time.

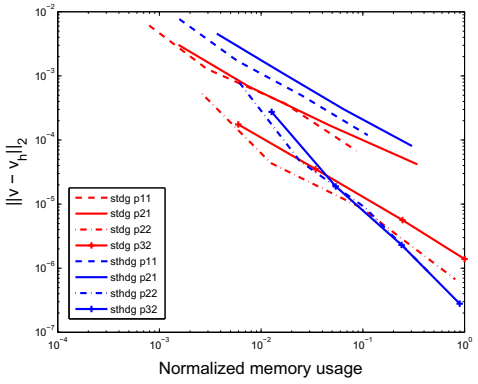
Fig. 3. The L^2-errors of the velocities and pressure against the normalized CPU times.

In [22] we found for the space–time HDG method that taking the polynomial approximation in time one order higher than the polynomial approximation in space restored the proper convergence rates of the pressure. Tables 3 and 4 show that the space–time DG method also obtains a better rate of convergence for the pressure if $p_t = p_s + 1$ compared to $p_t = p_s$ with a much better accuracy of the pressure. The analysis in [26] of the space–time DG method for the Oseen equations, shows that the pressure converges at a rate of $h^{p_s+\frac{1}{2}}$. This is the average rate of convergence we obtain here.

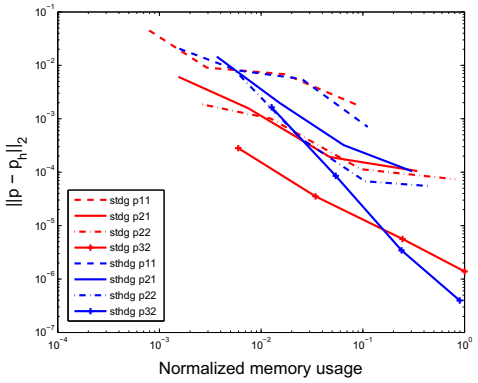
To compare the space–time DG and HDG methods, we plot the L^2-errors of the velocities and pressure against the total number of degrees of freedom of the global system, the CPU times (normalized with the biggest CPU time of our computations) and memory usage (normalized with the biggest memory usage of our computations), see Figs. 2–4. In the case of the space–time DG method, the total number of degrees of freedom of the global system is given by $3m_{p_t,p_s}N^e$, with m_{p_t,p_s} the number of basis-functions per element for a P^{p_t,p_s} polynomial approximation on the elements and N^e the total number of elements. For the space–time HDG method, the total number of degrees of freedom of the global system is given by $3m^f_{p_t,p_s}N^f$ with $m^f_{p_t,p_s}$ the number of basis-functions per face for a P^{p_t,p_s} polynomial approximation on the faces and N^f the



(a) L^2 error of the x-velocity vs. normalized memory usage.



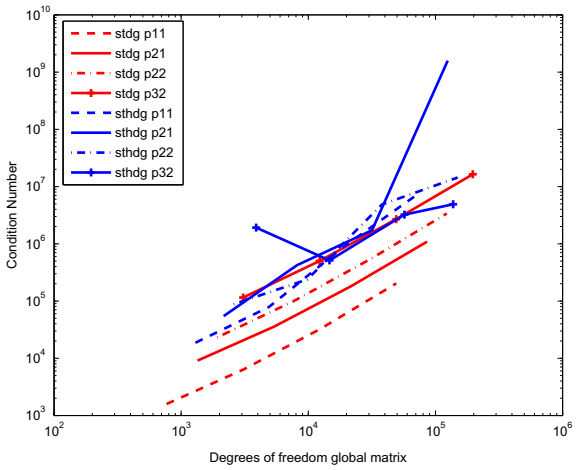
(b) L^2 error of the y-velocity vs. normalized memory usage.



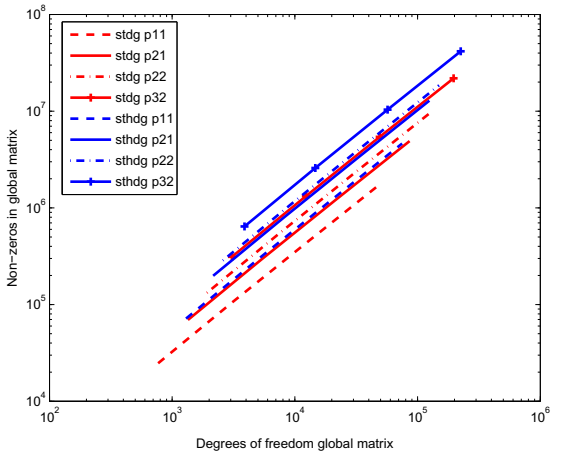
(c) L^2 error of the pressure vs. normalized memory usage.

Fig. 4. The L^2-errors of the velocities and pressure against the normalized memory usage.

number of (spatial) faces. From Fig. 2(a)–(c) it can be seen that the space–time DG method is slightly more accurate than the space–time HDG method in terms of L^2-error vs. degrees of freedom. Fig. 3(a)–(c) shows the L^2-error vs. CPU times. For the $p_s = 1$ computations, the space–time DG method has a slight advantage over space–time HDG, while the opposite is true for the $p_s = 2$ computations. For the $p_t = p_s + 1 = 3$ computation on the finest grid, the space–time HDG method performs much better than the space–time DG method. A similar conclusion can be found for the L^2-error vs. the memory usage (see Fig. 4(a)–(c)). For the $p_s = 1$ computations, the space–time DG method has a slight advantage over space–time HDG. For the $p_s = 2$ computations, both methods behave approximately the same. Furthermore, note that if $p_t = p_s + 1$ we are just as accurate for the velocity terms as when taking $p_t = p_s$ (see Figs. 2(a), (b), 3(a), (b) and 4(a), (b)) but taking $p_t = p_s + 1$ results in much better convergence rates for the pressure (see Figs. 2(c), 3(c) and 4(c)). In summary, from Figs. 2–4 we see that the main difference between the space–time DG and HDG methods is in the computation of the pressure for $p_t = p_s + 1 = 3$.

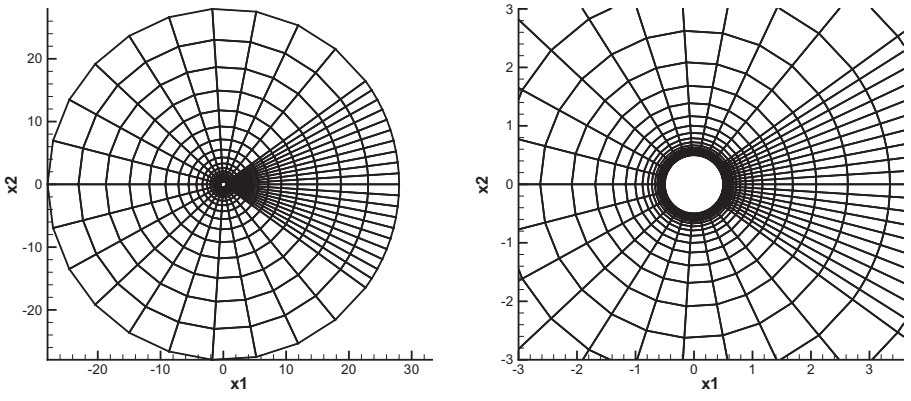


(a) Condition number vs. D.O.F..



(b) Non-zeros vs. D.O.F..

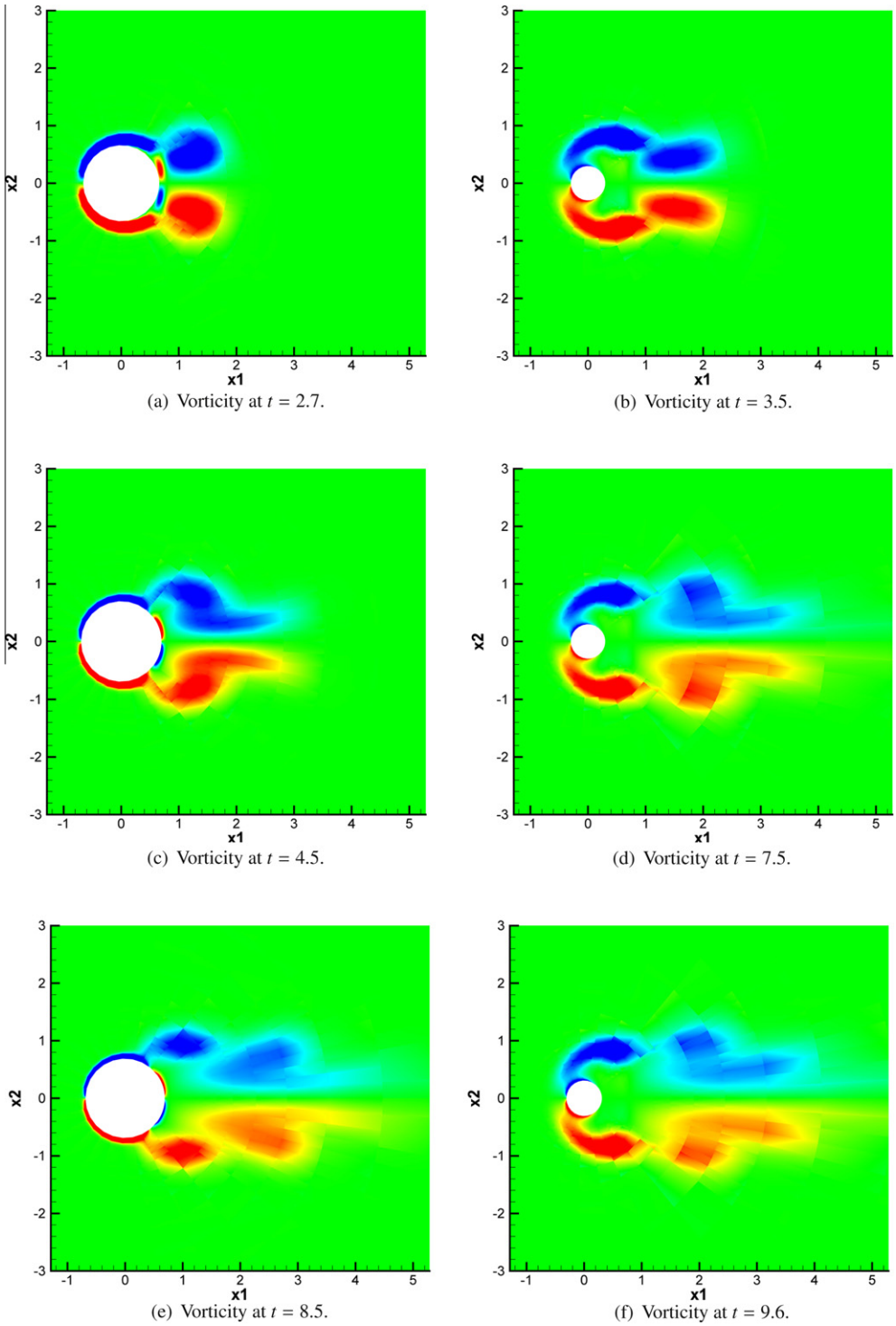
Fig. 5. The condition number and number of non-zeros in the global matrix against the total number of degrees of freedom of the global system.



(a) Mesh of cylinder. (b) Zoom around cylinder.

Fig. 6. Mesh of the cylinder at $t = 0$ for test case of Section 4.3.

The CPU-time and memory use, however, are strongly implementation dependent. To give a fair comparison we have used as much overlapping code as possible using the software package hpGEM [21] coupled to PETSc [2].



(a) Vorticity at $t = 2.7$. (b) Vorticity at $t = 3.5$.

(c) Vorticity at $t = 4.5$. (d) Vorticity at $t = 7.5$.

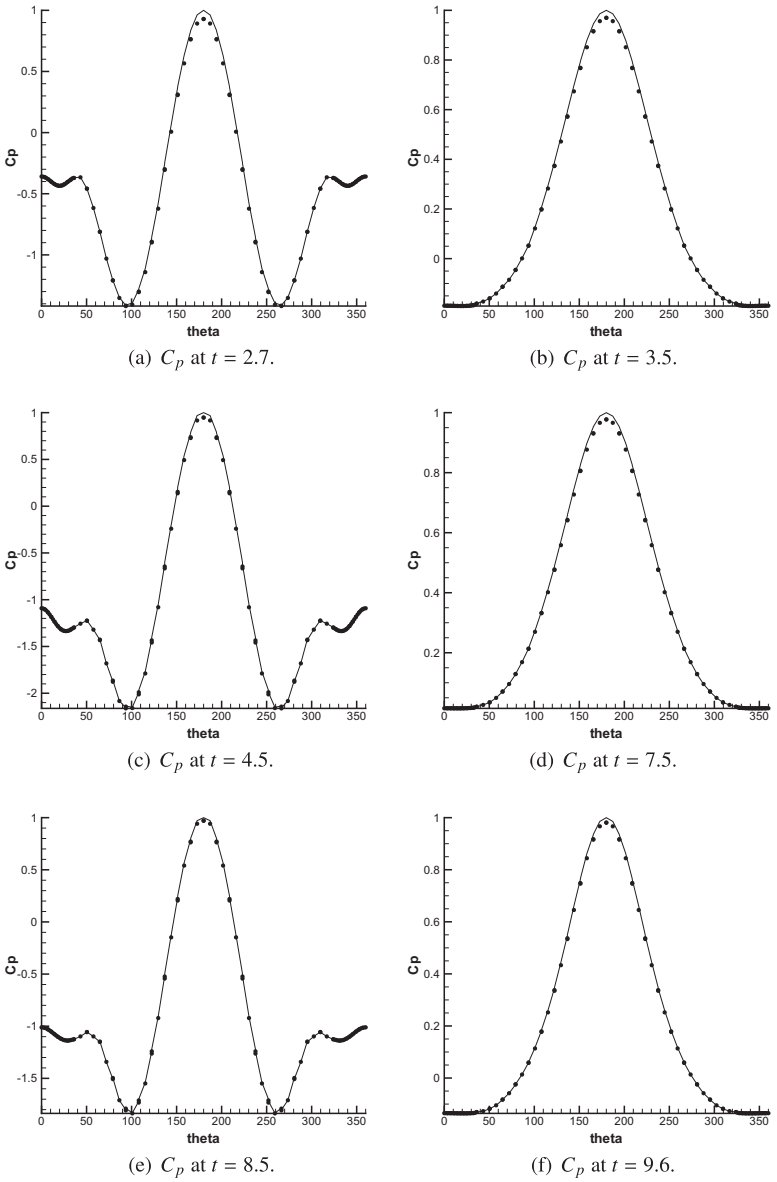
(e) Vorticity at $t = 8.5$. (f) Vorticity at $t = 9.6$.

Fig. 7. Snapshots of the vorticity for test case 4.3 on a 40×30 grid using $p_t = p_s + 1 = 3$. The pictures shown here were computed using the *nonconservative* stabilization space–time DG method.

In Fig. 5 we plot the condition number of the global system and the number of non-zeros in the global system against the total number of degrees of freedom of the global system. We see in Fig. 5(a) that the condition number of the space–time DG method is approximately an order smaller for the same polynomial approximation than that of the space–time HDG method, the exception being the $p_t = p_s + 1 = 3$ computation. This could mean that the space–time DG method is easier to solve. Note

that, with the exception of the $p_t = p_s + 1 = 2$ computation, the condition number for the space–time HDG method remains approximately the same for varying polynomial approximations. For the space–time DG method, as the polynomial approximation increases, the condition number increases. Furthermore, from Fig. 5(b) we see that, for the same number of global degrees of freedom, the global matrix of the space–time DG method is sparser than that of the space–time HDG method. This results in less memory requirements for the space–time DG method.

An advantage of the HDG method compared to the DG method is that while the DG method only provides suboptimal rates of convergence for the velocity gradient (see e.g. [6]), the HDG method results show optimal rates of convergence for the velocity gradients [18,19,22]. Due to this optimal rate of convergence for the velocity gradients, a simple element-by-element postprocessing [18,19] for the HDG method can be devised resulting in super-convergence rates for post-processed velocities. This could make the space–time HDG method more efficient than the space–time DG method. That said, if a DG method can be devised such that its velocity gradients converge with optimal rates, the HDG post-processing could also be applied to the DG method, possibly resulting also in super-convergence rates for the DG computed velocity fields. We refer to [9] in which it is explained how such a DG method may be devised. However, as it is noted in [9], the main drawback of such a DG method is that it will be very difficult to implement.



(a) C_p at $t = 2.7$. (b) C_p at $t = 3.5$.

(c) C_p at $t = 4.5$. (d) C_p at $t = 7.5$.

(e) C_p at $t = 8.5$. (f) C_p at $t = 9.6$.

Fig. 8. The pressure coefficient C_p at different time instances for test case 4.3. We use a space–time DG method with *No stabilization* (line) and *nonconservative* (dots) stabilization on a 40×30 grid using $p_t = p_s + 1 = 3$. We remark that $\theta = 0°$, $90°$, $180°$ and $270°$ denote the points $(r(t), 0), (0, r(t)), (-r(t), 0)$ and $(0, -r(t))$, respectively, on the cylinder boundary.

4.3. An expanding/contracting cylinder

We consider now a more challenging test case for the space–time DG method. For this we consider a time-dependent problem with moving boundary and deforming mesh to get information about the performance of the algorithm. Our test case is a model problem for the flow around underwater flexible hoses, e.g. used for pumping oil from reservoirs to tankers. For this purpose we will consider the flow around an expanding and contracting cylinder. We let the radius $r(t)$ of a cylinder be time-dependent and varying according to the formula $r(t) = r_0 + a_0 \sin(\pi t)$. We take $r_0 = 0.5$ and $a_0 = 0.2$. The mesh at time $t = 0$ is shown in Fig. 6. The outer radius, R, is constant and fixed at $R = 28$. The mesh consists of 40 rings and 30 sectors with refinement near the cylinder boundary and in the wake of the cylinder.

We consider the space–time DG method using *nonconservative* and *No stabilization* with $p_t = p_s + 1 = 3$. As physical parameters in this test case, we take $\Delta t = 0.005$ and $\nu = 0.01$. Numerical parameters are taken to be $\eta = 8$ and $\gamma = 10$. In the Picard iteration we require the residual to drop 9 orders in magnitude. Using a *nonconservative stabilization*, this is achieved each time step in approximately 4–5 Picard iterations. Approximately 5–6 Picard iterations are needed when *No stabilization* is used.

As initial condition, we set $u = (1, 0)$. On the inflow boundary we specify Dirichlet boundary conditions $u = (1, 0)$ while on the cylinder we specify $u = (0, 0)$. A Neumann boundary condition is specified on the outflow in which we set $g^N = 0$.

Contour plots of the vorticity and the contraction and expansion of the cylinder are shown in Fig. 7 at different time instances. Note that the vorticity is computed using the gradient of the velocity so that its order is one less than that of the pressure and velocity fields. We show here the pictures computed using the *nonconservative* stabilization space–time DG method, however, the pictures of the space–time DG method with *No stabilization* are identical. In Fig. 8 we compare the pressure coefficient $C_p = (p - p_\infty)/(\frac{1}{2}\rho_\infty u_\infty^2)$, $p_\infty = 0, \rho_\infty = 1, u_\infty = 1$ along the cylinder boundary at different time instances computed using the *nonconservative* and the *No stabilization* options. We see that the computed C_p is almost identical for both methods, except for a slight deviation near the stagnation point, $(-r(t), 0)$. To quantify, with respect to $\max(C_p) - \min(C_p)$, the difference is at most 3%.

5. Conclusions

We have introduced a space–time DG finite element method for the incompressible Navier–Stokes equations. To ensure stability of the method we introduced a space–time *BDM*-projection and the more simpler stabilization method of adding nonconservative terms to the weak formulation. From numerical experiments, using the *BDM stabilization* or the *nonconservative stabilization* improves slightly the efficiency of the method compared to *No stabilization*, since less Picard iterations are needed per time step to converge to a given tolerance. However, the finer the mesh, the less significant this effect becomes.

We numerically investigated also the use of using different orders of polynomial approximation in time, p_t, and space, p_s. We found that when $p_t = p_s$, the rates of convergence of the pressure were very irregular and suboptimal. Increasing the polynomial approximation in time such that $p_t = p_s + 1$ improved the rates of convergence of the pressure, showing less irregularity, although it is difficult to say whether the rates of convergence for the pressure are optimal. Rates of convergence for the velocity fields are optimal. We showed that despite that the global number of degrees of freedom increase when $p_t = p_s + 1$ compared to $p_t = p_s$, using $p_t = p_s + 1$ is more efficient since the error of the solution in the L^2-norm is smaller for an equal amount of global number of degrees of freedom. The CPU times and memory usage are also less when using $p_t = p_s + 1$ compared to $p_t = p_s$ to obtain a specified L^2-error of the pressure.

Solutions of computations using the space–time DG method are compared with those obtained from a space–time HDG method. We show that in terms of the L^2 errors of the velocity and pressure vs. total number of degrees of freedom, CPU times and memory usage, both methods perform approximately equally well. A reason for using the space–time DG method is that more research has been done in efficiently solving the global systems arising from the discretization (see e.g. [27,28] for multigrid methods for space–time DG methods). Also, the space–time DG method is less complex than the space–time HDG method. Reasons to use the space–time HDG method, however, are that the space–time HDG method also computes the velocity gradient with optimal rates of convergence. This has not been achieved yet with standard DG methods. It is due to these optimal convergence rates that post-processing techniques can be developed for the HDG method so that post-processed velocities converge with super-convergence rates [18,19].

Our final numerical computation shows the method's ability to cope with a more challenging test case. We compute the flow around an expanding and contracting cylinder. The radius of the cylinder is time-dependent so that the domain is continuously deforming. Results of the space–time DG method without stabilization are compared to using a *nonconservative stabilization*. The number of Picard-iterations per time step are slightly less for the space–time DG method with *nonconservative stabilization*. The numerical solutions, however, are almost identical. Only in the computation of the pressure coefficient, C_p, did we notice a slight deviation in the solution around the stagnation point, $\theta = 180°$.

Acknowledgments

Sander Rhebergen gratefully acknowledges funding by a Rubicon Fellowship from the Netherlands Organisation for Scientific Research (NWO) and the Marie Curie Cofund Action. Bernardo Cockburn was supported in part by the National Science Foundation (Grant DMS-1115331) and by the University of Minnesota Super Computing Institute.

Appendix A. The space–time hybridizable discontinuous Galerkin method

In [22] we introduced a space–time hybridizable discontinuous Galerkin (HDG) method for incompressible flows. For completeness we summarize here the weak formulation and refer to [22] for more details.

Besides the discontinuous finite element spaces defined in (6), the space–time HDG method also requires spaces for the approximate trace of the velocities and pressure:

$$M_h^{(p_t,p_s)} = \{\mu \in L^2(\mathcal{F})^d : \mu|_{\mathcal{S}} \circ F_{\mathcal{S}} \in (P^{(p_t,p_s)}(\hat{\mathcal{S}}))^d, \forall \mathcal{S} \in \mathcal{F}\},$$
$$\Psi_h^{(p_t,p_s)} = \{\psi \in L^2(\mathcal{F}) : \psi|_{\mathcal{S}} \circ F_{\mathcal{S}} \in P^{(p_t,p_s)}(\hat{\mathcal{S}}), \forall \mathcal{S} \in \mathcal{F}\}.$$

The space–time HDG weak formulation for the Oseen equations, needed in each Picard-step, is given by: *find an approximation* $(\sigma, u, p) \in \Sigma_h^{(p_t,p_s)} \times V_h^{(p_t,p_s)} \times Q_h^{(p_t,p_s)}$ *such that for all* $\mathcal{K} \in \mathcal{T}_h$*:*

velocity gradient equation:

$$\sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} \sigma_{ij}\tau_{ij}\,dx + \sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{K}} u_i \tau_{ij,j}\,dx - \sum_{\mathcal{K}\in\mathcal{T}_h^n} \int_{\mathcal{Q}_{\mathcal{K}}^n} \hat{u}_i \bar{n}_j \tau_{ij}\,ds = 0, \tag{A.1a}$$

momentum equation:

$$\begin{aligned} &-\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} u_i w_j v_{i,j}\,dx + \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} (\nu\sigma_{ij} - \delta_{ij}p) v_{i,j}\,dx + \sum_{\mathcal{S}\in\cup_n\mathcal{S}_S^n} \int_{\mathcal{S}} [\![v_i]\!]_j \left(\{u_i\} w_j + C_S [\![u_i]\!]_j \right) d\bar{x} \\ &\quad + \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{Q}_{\mathcal{K}}^n} \left(\hat{u}_i w_j n_j - \nu\sigma_{ij}\bar{n}_j + \delta_{ij}\hat{p}\bar{n}_j + S_{ir}(u_r - \hat{u}_r) \right) v_i\,ds \\ &\quad = \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} f_i v_i\,dx + \int_{\Omega_0} u_i^0 v_i\,d\bar{x}, \end{aligned} \tag{A.1b}$$

mass equation:

$$-\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{K}} u_i q_{,i}\,dx + \sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{Q}_{\mathcal{K}}^n} \left(\hat{u}_i \bar{n}_i + S^p(p - \hat{p}) \right) q\,ds = 0, \tag{A.1c}$$

for all $(\tau, v, q) \in \Sigma_h^{(p_t,p_s)} \times V_h^{(p_t,p_s)} \times Q_h^{(p_t,p_s)}$. Here $S \in \mathbb{R}^{n\times n}$ is a second order tensor of stabilization parameters and S^p a scalar stabilization parameter. By defining $S = \theta \mathbb{I}_d$ and $S_p = \left(4\theta - 2w_j n_j\right)^{-1}$, where $\mathbb{I}_d \in \mathbb{R}^{d\times d}$ is the identity matrix, $\theta = \max(|w_j n_j|, \delta) + \nu/\ell, \ell$ a representative diffusive length scale and $\delta > 0$ is a given constant, the existence and uniqueness of the Oseen problem can be proven to exist [22]. Furthermore, in (A.1), the approximation of the traces $\hat{u}$ and $\hat{p}$ are required. These are determined such that continuity of the normal component of the numerical trace of the total "momentum"-flux and "mass"-flux is enforced. This is achieved by requiring

$$\begin{aligned} &\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{Q}_{\mathcal{K}}^n} \left(\hat{u}_i w_j n_j - \nu\sigma_{ij}\bar{n}_j + \delta_{ij}\hat{p}\bar{n}_j + S_{ir}(u_r - \hat{u}_r) \right) \mu_i\,ds = 0, \\ &\sum_{\mathcal{K}\in\mathcal{T}_h} \int_{\mathcal{Q}_{\mathcal{K}}^n} \left(\hat{u}_i \bar{n}_i + S^p(p - \hat{p}) \right) \psi\,ds = 0, \end{aligned} \tag{A.2}$$

for all $(\mu, \psi) \in (M_h, \Psi_h)$. The Dirichlet and Neumann boundary conditions are imposed by:

$$\begin{aligned} &\int_{\mathcal{S}} \hat{u}_i \mu_i\,ds = \int_{\mathcal{S}} g_i^D \mu_i\,ds, \quad \forall \mathcal{S} \in \cup_n \mathcal{S}_{Dm}^n, \\ &\int_{\mathcal{S}} S_{ir}(u_r - \hat{u}_r)\mu_i\,ds = 0, \quad \forall \mathcal{S} \in \cup_n \mathcal{S}_{Dp}^n, \\ &\int_{\mathcal{S}} \mu_i \left(\delta_{ij}\bar{n}_j\hat{p} - \nu\sigma_{ij}\bar{n}_j + S_{ir}(u_r - \hat{u}_r) \right) ds = \int_{\mathcal{S}} \mu_i \left(\delta_{ij}\bar{n}_j p_N - b_i^N \right) ds, \quad \forall \mathcal{S} \in \cup_n \mathcal{S}_N^n, \\ &\int_{\mathcal{S}} S^p(p - \hat{p})\psi\,ds = 0, \quad \forall \mathcal{S} \in \cup_n \left(\mathcal{S}_D^n \cup \mathcal{S}_N^n \right). \end{aligned}$$

References

[1] D.N. Arnold, F. Brezzi, B. Cockburn, L.D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (2002) 1749–1779.

[2] S. Balay, J. Brown, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L. Curfman McInnes, B.F. Smith, H. Zhang, PETSc Web page, 2011. Available from: <http://www.mcs.anl.gov/petsc>.

[3] F. Bassi, A. Crivellini, D.A. Di Pietro, S. Rebay, An implicit high-order discontinuous Galerkin method for steady and unsteady incompressible flows, Comput. Fluids 36 (2007) 1529–1546.
[4] M. Benzi, M.A. Olshanskii, An augmented Lagrangian-based approach to the Oseen problem, SIAM J. Sci. Comput. 28 (2006) 2095–2113.
[5] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, Springer Verlag, 1991.
[6] B. Cockburn, G. Kanschat, D. Schötzau, The local discontinuous Galerkin method for the Oseen equations, Math. Comp. 73 (2003) 569–593.
[7] B. Cockburn, G. Kanschat, D. Schötzau, A locally conservative LDG method for the incompressible Navier–Stokes equations, Math. Comp. 74 (2004) 1067–1095.
[8] B. Cockburn, G. Kanschat, D. Schötzau, A note on discontinuous Galerkin divergence-free solutions of the Navier–Stokes equations, J. Sci. Comput. 31 (2007) 61–73.
[9] B. Cockburn, J. Guzmán, H. Wang, Superconvergent discontinuous Galerkin methods for second-order elliptic problems, Math. Comp. 78 (2008) 1–24.
[10] B. Cockburn, G. Kanschat, D. Schötzau, An equal-order DG method for the incompressible Navier–Stokes equations, J. Sci. Comput. 40 (2009) 188–210.
[11] E. Ferrer, R.H.J. Willden, A high order discontinuous Galerkin finite element solver for the incompressible Navier–Stokes equations, Comput. Fluids 46 (2011) 224–230.
[12] V. Girault, F. Wheeler, Discontinuous Galerkin methods, Comput. Methods Appl. Sci. 16 (2008) 3–26.
[13] H. Guillard, C. Farhat, On the significance of the geometric conservation law for flow computations on moving meshes, Comput. Methods Appl. Mech. Eng. 190 (2000) 1467–1482.
[14] M.-C. Hsu, I. Akkerman, Y. Bazilevs, High-performance computing of wind turbine aerodynamics using isogeometric analysis, Comput. Fluids 49 (2011) 93–100.
[15] R.M. Kirby, S.J. Sherwin, B. Cockburn, To CG or to HDG: a comparative study, J. Sci. Comput. Available from: http://dx.doi.org/10.1007/s10915-011-9501-7.
[16] C.M. Klaij, J.J.W. van der Vegt, H. van der Ven, Space-time discontinuous Galerkin method for the compressible Navier–Stokes equations, J. Comput. Phys. 217 (2006) 589–611.
[17] M. Lesoinne, C. Farhat, Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations, Comput. Methods Appl. Mech. Eng. 134 (1996) 71–90.
[18] N.C. Nguyen, J. Peraire, B. Cockburn, A hybridizable discontinuous Galerkin method for Stokes flow, Comput. Methods Appl. Mech. Eng. 199 (2010) 582–597.
[19] N.C. Nguyen, J. Peraire, B. Cockburn, An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations, J. Comput. Phys. 228 (2011) 1147–1170.
[20] P.-O. Persson, J. Bonet, J. Peraire, Discontinuous Galerkin solution of the Navier–Stokes equations on deformable domains, Comput. Methods Appl. Mech. Eng. 198 (2009) 1585–1595.
[21] L. Pesch, A. Bell, W.E.H. Sollie, V.R. Ambati, O. Bokhove, J.J.W. van der Vegt, hpGEM – a software framework for discontinuous Galerkin finite element methods, ACM Trans. Math. Softw. 33 (2007).
[22] S. Rhebergen, B. Cockburn, A space-time hybridizable discontinuous Galerkin method for incompressible flows on deforming domains, J. Comput. Phys. 231 (2012) 4185–4204.
[23] K. Shahbazi, P.F. Fischer, C.R. Ethier, A high-order discontinuous Galerkin method for the unsteady incompressible Navier–Stokes equations, J. Comput. Phys. 222 (2007) 391–407.
[24] S. Ushijima, Three-dimensional arbitrary Lagrangian–Eulerian numerical prediction method for non-linear free surface oscillation, Int. J. Numer. Methods Fluids 26 (1998) 605–623.
[25] J.J.W. van der Vegt, H. van der Ven, Space–time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. I. General formulation, J. Comput. Phys. 182 (2002) 546–585.
[26] J.J.W. van der Vegt, J.J. Sudirham, A space–time discontinuous Galerkin method for the time-dependent Oseen equations, Appl. Numer. Math. 58 (2008) 1892–1917.
[27] J.J.W. van der Vegt, S. Rhebergen, hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part I. Multilevel analysis, J. Comput. Phys. 231 (2012) 7537–7563.
[28] J.J.W. van der Vegt, S. Rhebergen, hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part II. Optimization of the Runge-Kutta smoother, J. Comput. Phys. 231 (2012) 7564–7583.