Fast matrix-free evaluation of hybridizable discontinuous Galerkin operators

Martin Kronbichler[1], Katharina Kormann[2], and Wolfgang A. Wall[1]

[1] Institute for Computational Mechanics, Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany, kronbichler@lnm.mw.tum.de, wall@lnm.mw.tum.de

[2] Max Planck Institute for Plasma Physics, Boltzmannstr. 2, 85748 Garching, Germany, katharina.kormann@ipp.mpg.de

Abstract. This paper proposes a new algorithm for fast matrix-free evaluation of linear operators based on hybridizable discontinuous Galerkin discretizations with sum factorization, exemplified for the convection-diffusion equation on quadrilateral and hexahedral elements. The matrix-free scheme is based on a formulation of the method in terms of the primal variable and the trace. The proposed method is shown to be up to an order of magnitude faster than the traditionally considered matrix-based formulation in terms of the trace only, despite using more degrees of freedom. The impact of the choice of basis on the evaluation cost is discussed, showing that Lagrange polynomials with nodes co-located with the quadrature points are particularly efficient.

1 Introduction

Discontinuous Galerkin methods are highly attractive discretization schemes for a wide variety of problems, in particular for problems with strong transport character as they allow for upwinding and are amenable to efficient explicit time stepping. They are also increasingly used in problems dominated by elliptic terms where linear systems need to be solved. Whereas classical discontinuous schemes have often been criticized as being less efficient than continuous ones due to more degrees of freedom and a wider stencil for delivering the same accuracy, the hybridizable discontinuous Galerkin (HDG) method developed by Cockburn and co-workers [3,10] has addressed this deficiency by expressing the connectivity in the final system matrix only through a variable for the trace of the solution on the mesh skeleton, into which all volume terms are absorbed by a static-condensation-like technique. This reduced number of unknowns has been shown to render the method highly efficient when compared to continuous and other discontinuous Galerkin discretizations, in particular for higher polynomial degrees [4,12].

When using modern iterative solvers with matrix-free implementations, however, matrix-based HDG is still lagging behind formulations in the primal variable amenable to matrix-free operator evaluation [7,8]. This work closes this gap by developing a new matrix-free scheme for a particular HDG variant in the primal variable u and the trace $\widehat{u}$ that was recently proposed by

Cockburn [2], relying on the algorithms from the generic finite element library `deal.II` [5,6]. We show that operator evaluation is more than an order of magnitude faster with the proposed implementation than matrix-based HDG for moderate polynomial degrees of four to five in three spatial dimensions. The beneficial properties of the new method motivate future work on efficient preconditioners for iterative solvers with this scheme.

This article is structured as follows. Sect. 2 introduces the HDG discretization and details the two possible matrix formulations. In Sect. 3 the sum factorization ingredients and efficient bases are presented. Sect. 4 shows our computational results and gives an outlook to future work.

2 Hybridizable discontinuous Galerkin discretization

We consider the linear convection-diffusion equation on a bounded computational domain $\Omega \subset \mathbb{R}^d$ in d space dimensions,

$$\begin{aligned} &\nabla \cdot (\boldsymbol{c} u) - \nabla \cdot (\kappa \nabla u) = f \quad \text{in } \Omega, \\ &u = g_{\mathrm{D}} \quad \text{on } \partial\Omega_{\mathrm{D}}, \qquad (-k\nabla u + \boldsymbol{c} u)\cdot \boldsymbol{n} = g_{\mathrm{N}} \quad \text{on } \partial\Omega_{\mathrm{N}}, \end{aligned} \tag{1}$$

where $\boldsymbol{c}$ is the direction of transport, $\kappa > 0$ is the diffusivity, and f is a given forcing. The solution u is found subject to a Dirichlet value g_D on $\partial\Omega_{\mathrm{D}} \subset \partial\Omega$ and a Neumann condition on $\partial\Omega_{\mathrm{N}} = \partial\Omega \setminus \partial\Omega_{\mathrm{D}}$.

We assume a tesselation $\mathcal{T}_h$ that partitions the computational domain Ω into n_e elements Ω_e of characteristic size h. We specialize algorithms for quadrilateral or hexahedral elements on which sum factorization is straightforwardly implemented. An element Ω_e is the image of the reference domain $[-1,1]^d$ under a polynomial mapping of degree k. We denote by $V_{h,k}$ the space of admissible solutions that are polynomials of tensor degree k. The HDG method [10], which rewrites the convection-diffusion equation (1) into a system in the primal variable u and the flux $\boldsymbol{q} = -\kappa\nabla u$, uses this space for both u and each component of $\boldsymbol{q}$. A third variable $\widehat{u}$ is introduced on the mesh skeleton, the faces $\mathcal{F}_h$ of the triangulation, with the associated solution space $M_{h,k}$ as the $(d-1)$-dimensional tensor product polynomials on the faces $\mathcal{F}_h$ that are zero on the Dirichlet boundaries $\partial\Omega_{\mathrm{D}}$. The resulting weak form is to find a triple $(\boldsymbol{q}, u, \widehat{u}) \in V_{h,k}^d \times V_{h,k} \times M_{h,k}$ such that it holds

$$\begin{aligned} &(\boldsymbol{w}, \kappa^{-1}\boldsymbol{q})_{\mathcal{T}_h} - (\nabla\cdot\boldsymbol{w}, u)_{\mathcal{T}_h} + \langle \boldsymbol{w}\cdot\boldsymbol{n}, \widehat{u}\rangle_{\partial\mathcal{T}_h} = -\langle \boldsymbol{w}\cdot\boldsymbol{n}, g_{\mathrm{D}}\rangle_{\partial\Omega_{\mathrm{D}}} && \forall \boldsymbol{w} \in V_{h,k}^d, \\ &-(\nabla v, \boldsymbol{c}u + \boldsymbol{q})_{\mathcal{T}_h} + \langle v, (\boldsymbol{c}\widehat{u} + \boldsymbol{q})\cdot\boldsymbol{n} + \tau(u-\widehat{u})\rangle_{\partial\mathcal{T}_h} = (v, f)_{\mathcal{T}_h} && \forall v \in V_{h,k}, \\ &\langle \mu, (\boldsymbol{c}\widehat{u} + \boldsymbol{q})\cdot\boldsymbol{n} + \tau(u-\widehat{u})\rangle_{\partial\mathcal{T}_h} = \langle \mu, g_{\mathrm{N}}\rangle_{\partial\Omega_{\mathrm{N}}} && \forall \mu \in M_{h,k}. \end{aligned} \tag{2}$$

We denote by $(\cdot,\cdot)_{\mathcal{T}_h}$ the bilinear form associated with cell integrals in the usual finite element fashion and by $\langle\cdot,\cdot\rangle_{\partial T_h}$ the bilinear form for face integrals, where each interior face is visited twice, involving different fields u and $\boldsymbol{q}$ but the same $\widehat{u}$ on the two sides. The stabilization parameter τ can

come in various flavors adapted to the problem nature [10]. We exemplify the implementation with a centered scheme as $\tau = |\boldsymbol{c}\cdot\boldsymbol{n}| + \frac{\kappa}{\ell}$ with $\ell = 5$.

The weak forms in (2) give rise to the following matrices, see also [10],

$$\begin{aligned} A_{ij} &= (\boldsymbol{\varphi}_i, \kappa^{-1}\boldsymbol{\varphi}_j)_{\mathcal{T}_h}, & B_{ij} &= -(\nabla\varphi_i, \boldsymbol{\varphi}_j)_{\mathcal{T}_h} + \langle \varphi_i, \boldsymbol{\varphi}_j\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h}, \\ C_{ij} &= \langle \psi_i, \boldsymbol{\varphi}_j\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h}, & D_{ij} &= -(\nabla\varphi_i, \boldsymbol{c}\varphi_j)_{\partial T_h} + \langle \varphi_i, \tau\varphi_j\rangle_{\partial\mathcal{T}_h}, \\ E_{ij} &= \langle \varphi_i, (\boldsymbol{c}\cdot\boldsymbol{n}-\tau)\psi_j\rangle_{\partial\mathcal{T}_h}, & G_{ij} &= \langle \psi_i, \tau\varphi_j\rangle_{\partial\mathcal{T}_h}, \\ H_{ij} &= \langle \psi_i, (\boldsymbol{c}\cdot\boldsymbol{n}-\tau)\psi_j\rangle_{\partial\mathcal{T}_h}, \end{aligned} \tag{3}$$

where $\boldsymbol{\varphi}_i$ denotes vector-valued shape functions spanning $V_{h,k}^d$, φ_i are basis functions spanning $V_{h,k}$, and ψ_i are the basis functions of $M_{h,k}$. The right hand side of (2) gives rise to the three vectors

$$R_i = \langle \boldsymbol{\varphi}_i\cdot\boldsymbol{n}, g_{\mathrm{D}}\rangle_{\partial\Omega_{\mathrm{D}}}, \quad F_i = (\varphi_i, f)_{\mathcal{T}_h}, \quad L_i = \langle \psi_i, g_{\mathrm{N}}\rangle_{\partial\mathcal{T}_h}. \tag{4}$$

With these ingredients, the discrete HDG method seeks for the coefficient vectors $(\boldsymbol{Q}, \boldsymbol{U}, \widehat{\boldsymbol{U}})$ through the linear system

$$\begin{pmatrix} A & -B^{\mathrm{T}} & C^{\mathrm{T}} \\ B & D & E \\ C & G & H \end{pmatrix} \begin{pmatrix} \boldsymbol{Q} \\ \boldsymbol{U} \\ \widehat{\boldsymbol{U}} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{R} \\ \boldsymbol{F} \\ \boldsymbol{L} \end{pmatrix}. \tag{5}$$

Formulation in terms of the trace $\widehat{u}$ An attractive feature of the HDG method is the possibility to statically condense out some of the degrees of freedom before solving the linear system [3]. This can significantly speed up the solution of the final linear system in implicit methods, besides the beneficial convergence properties of HDG fluxes [3,10]. In equation (5), the matrices A, B, D are block-diagonal over elements because the coupling is expressed solely in terms of the trace variable $\widehat{u}$. Thus, they can be inverted element by element through a Schur complement, eliminating $\boldsymbol{q}$ and u from the system. This reduces the matrix-vector product complexity per element from $\mathcal{O}(k^{2d})$ to $\mathcal{O}(k^{2d-2})$. The resulting linear system in the trace reads

$$K\widehat{\boldsymbol{U}} = \boldsymbol{T}, \tag{6}$$

where the matrix K and right hand side vector $\boldsymbol{T}$ are given by

$$K = H - \begin{pmatrix} C & G \end{pmatrix} \begin{pmatrix} A & -B^{\mathrm{T}} \\ B & D \end{pmatrix}^{-1} \begin{pmatrix} C^{\mathrm{T}} \\ E \end{pmatrix}, \; \boldsymbol{T} = \boldsymbol{L} - \begin{pmatrix} C & G \end{pmatrix} \begin{pmatrix} A & -B^{\mathrm{T}} \\ B & D \end{pmatrix}^{-1} \begin{pmatrix} -\boldsymbol{R} \\ \boldsymbol{F} \end{pmatrix}.$$

The trace formulation has been used in a large number of works on HDG, including the performance comparisons [4,12]. The static condensation with inverse matrices, however, mandates to resort to an explicit sparse matrix storage.[1]

[1] Note that we do not consider the narrow case where the mesh is Cartesian, $\boldsymbol{c}$ is axis-aligned and element-wise constant, and κ is constant, when the matrix for $(\boldsymbol{q}, u)$ is separable and can be expressed as a Kronecker product of 1D matrices with the inverse given by the fast diagonalization method [9].

Formulation in terms of $\boldsymbol{u}$ and $\widehat{\boldsymbol{u}}$ An alternative formulation of HDG proposed recently [2] is to express the final linear system in the two variables u and $\widehat{u}$, rather than the trace only,

$$\left[\begin{pmatrix} D & E \\ G & H \end{pmatrix} - \begin{pmatrix} B \\ C \end{pmatrix} A^{-1} \begin{pmatrix} -B^{\mathrm{T}} & C^{\mathrm{T}} \end{pmatrix}\right] \begin{pmatrix} \boldsymbol{U} \\ \widehat{\boldsymbol{U}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{L} \end{pmatrix} + \begin{pmatrix} B \\ C \end{pmatrix} A^{-1} \boldsymbol{R}. \quad (7)$$

The advantage of this compact form is that only an inverse of a vector mass matrix A weighted by the diffusion coefficient appears, for which fast matrix-free inversion methods with lower complexity than the $2d(k+1)^{2d}$ arithmetic operations in the polynomial degree k of the full matrix inversion are available as detailed in the next section.

3 Sum factorization algorithms for HDG

We exemplify the structure of a matrix-free evaluation kernel on the multiplication by the cell integral contribution to the matrix B^{T} in 3D, i.e., the integral $-(\boldsymbol{\varphi}_i, \nabla\varphi_j)_{\mathcal{T}_h}$. The shape functions are assumed to be the product of three one-dimensional shape functions $\varphi_i(\boldsymbol{\xi}) = \varphi^{\mathrm{1D}}_{i_1}(\xi_1)\varphi^{\mathrm{1D}}_{i_2}(\xi_2)\varphi^{\mathrm{1D}}_{i_3}(\xi_3)$ for $i = 1, \ldots, (k+1)^3$. Integrals are evaluated with Gaussian quadrature in the points $(\xi^q_1, \xi^q_2, \xi^q_3)$, $q = 1, \ldots, n_q$. Here, we choose $n_q = (k+1)^3$ points which ensures exact integration of constant-coefficient weak forms on affine geometries. While this is usually enough also for representing integrals for curved geometries accurately, variable coefficients or nonlinearities often require more points to avoid aliasing effects. Our methods also apply to those cases, albeit at slightly reduced efficiency.

For matrix-free evaluation, we associate the input vector $\boldsymbol{U}$ with the representation $u^{(e)}_h(\boldsymbol{x}) = \sum_{j=1}^{(k+1)^3} \varphi_j(\boldsymbol{x}) U^{(e)}_j$ on a cell Ω_e. This quantity is then tested by all functions $\boldsymbol{\varphi}_i$. The integral is computed by a loop over all cells Ω_e. On each cell, the approximation reads

$$\int_{\Omega_e} \boldsymbol{\varphi}_i(\boldsymbol{x}) \cdot \nabla u_h(\boldsymbol{x}) d\boldsymbol{x} \approx \sum_{q=1}^{n_q^3} \boldsymbol{\varphi}_i(\boldsymbol{\xi}_q) \cdot \boldsymbol{J}_q^{-\mathrm{T}} \left(\sum_{j=1}^{(k+1)^3} \nabla_{\boldsymbol{\xi}} \varphi_j(\boldsymbol{\xi}_q) U^{(e)}_j \right) \det(\boldsymbol{J}_q) w_q,$$

where $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{\xi})$ is the Jacobian of the transformation from the reference to the real cell and $\nabla_{\boldsymbol{\xi}}$ denotes the gradient with respect to the reference coordinates. Moreover, w_q and $\boldsymbol{\xi}_q$ denote the quadrature weights and points, respectively. The three steps to evaluate the integrals for all test functions on an element are (a) the evaluation of the solution gradient in all quadrature points, (b) the multiplication by the inverse Jacobian $\boldsymbol{J}_q^{-\mathrm{T}} = \boldsymbol{J}^{-\mathrm{T}}(\xi_q)$ and the factor $\det(\boldsymbol{J}_q)w_q$, and (c) the multiplication by the values $\boldsymbol{\varphi}_i$ and summation of quadrature points for all i. Steps (a) and (c) can be recast as matrix-vector products of size $3n_q \times (k+1)^3$ and $3(k+1) \times n_q$, respectively. However, the potential cost of $(k+1)^6$ can be reduced to $(k+1)^4$ (or to $d(k+1)^{d+1}$

in space dimension d) by sum factorization, an algorithm that transforms the summations into one-dimensional kernels along all spatial directions by using the tensor product structure in the shape functions. Sum factorization has its origin in spectral elements [11] and has found widespread use in both continuous and discontinuous Galerkin implementations. The matrix defining the interpolation of $\nabla_{\boldsymbol{\xi}} u_h^{(e)}$ in the quadrature points reads

$$\begin{pmatrix} D_1^{\mathrm{co}} \otimes I_2 \otimes I_3 \\ I_1 \otimes D_2^{\mathrm{co}} \otimes I_3 \\ I_1 \otimes I_2 \otimes D_3^{\mathrm{co}} \end{pmatrix} \left(S_1 \otimes S_2 \otimes S_3\right) \boldsymbol{U}^{(e)}. \tag{8}$$

In this formula, the product by the first matrix $\left(S_1 \otimes S_2 \otimes S_3\right)$—with $(S_*)_{ij} = \varphi_j^{\mathrm{1D}}(\xi_i)$ the matrix of 1D shape functions evaluated in the 1D quadrature points—is a basis change [6] into the Lagrange basis defined in the points of Gaussian quadrature. This basis with collocated node and quadrature points reduces evaluation costs because it only involves $2d$ tensor product kernels to get all components of the gradient, as opposed to d^2 tensor product kernels in a naive implementation. The matrices D_*^{co} are the derivatives of the Lagrange polynomials in the Gauss points, i.e., in the collocation basis.

Matrix-vector product for general bases Using algorithms in the lines of (8) for both cell and face integrals as appropriate (e.g., using a matrix $S_1 \otimes S_2 \otimes S_3^f$ for an interpolation of values onto the face f with normal in ξ_3 direction with a $1 \times (k+1)$ matrix S_3^f), all integrals appearing in Eq. (7) can be evaluated with complexity $\mathcal{O}(k^d)$ operations for face integrals and $\mathcal{O}(k^{d+1})$ operations for the cell integrals. The computations are done by the following algorithmic steps in a loop over all cells (see also [7, Sec. 4.1] for a partly matrix-free variant):

1. Compute cell integral $\boldsymbol{P} = -(B^{\mathrm{c}})^{\mathrm{T}} \boldsymbol{U}^{(e)}$, with B^{c} the cell integral part of $B = B^{\mathrm{c}} + B^{\mathrm{f}}$ in Eq. (3).
2. Loop over all faces of cell e and add the face contribution,
   $\boldsymbol{P} = \boldsymbol{P} + \left(-(B^{\mathrm{f}})^{\mathrm{T}}\ C^{\mathrm{T}}\right) \begin{pmatrix} \boldsymbol{U}^{(e)} \\ \widehat{\boldsymbol{U}}^{(e)} \end{pmatrix}$.
3. Compute $\boldsymbol{Q}^{(e)} = A^{-1} \boldsymbol{P}$ by using the tensor product inverse matrix algorithm from [6, Sec. 4.4.1] (change to diagonal basis).
4. Compute cell integral contribution in $\boldsymbol{V}^{(e)} = D^{\mathrm{c}} \boldsymbol{U}^{(e)} - B^{\mathrm{c}} \boldsymbol{Q}^{(e)}$.
5. Loop over all faces of cell e
   (a) Add face integral $\boldsymbol{V}^{(e)} = \boldsymbol{V}^{(e)} + D^{\mathrm{f}} \boldsymbol{U}^{(e)} + E \widehat{\boldsymbol{U}}^{(e)} - (B^{\mathrm{f}}) \boldsymbol{Q}^{(e)}$.
   (b) Compute face integral $\widehat{\boldsymbol{V}}^{(e)} = G \boldsymbol{U}^{(e)} + H \widehat{\boldsymbol{U}}^{(e)} - C \boldsymbol{Q}^{(e)}$ and add into the respective position in the global result vector $\widehat{\boldsymbol{V}}$.

   When the loop over faces is completed, the cell contribution of $\boldsymbol{V}^{(e)}$ contains the result for the primal variable $\boldsymbol{V}$.

Table 1. Number of tensor product kernel calls for evaluation of HDG in $(u, \widehat{u})$ for 3D convection-diffusion operator of cost $(k+1)^4$ (cells) and $(k+1)^3$ (faces)

	cell terms			face terms	
	B^{T}	A^{-1}	B, D	$B^{\mathrm{T}}, C^{\mathrm{T}}$	second face loop
generic basis	15	18	18	6×14	6×24
collocation basis	3	—	3	6×4	6×4

Matrix-vector product for collocation basis The algorithm for generic basis contains a large number of multiplications by matrices of the form $S_1 \otimes S_2 \otimes S_3$, their transpose, and the respective face versions. In essence, these steps interpolate from the coefficients in the chosen basis to the values in the points of Gaussian quadrature. For the HDG method in $(u, \widehat{u})$ format, a basis with collocated node and quadrature points where S_1, S_2, S_3 are unit matrices is very beneficial because a large number of interpolations turn into the identity operation. In particular, the collocation basis is orthogonal with respect to the L_2 inner product and thus results in a diagonal mass matrix, considerably simplifying the multiplication by A^{-1}, in contrast to a generic tensor product basis with algorithm according to [7, Sec. 4.4.1].

Table 1 lists the number of tensor product calls for the collocation basis compared to a general basis. Both the number of cell sum factorization kernels and face sum factorization kernels are significantly reduced, which results in a significant saving also in our highly tuned implementation using advanced SIMD (vectorized) instructions [6], see Fig. 1. In particular for the face integrals, the only sum factorization calls are for the interpolation of contributions onto the faces. Note that besides the operations listed in the table, the algorithm also does $\mathcal{O}((k+1)^3)$ operations on quadrature points of cells and $\mathcal{O}((k+1)^2)$ on quadrature points of faces.

4 Numerical experiments and outlook

We run experiments of an MPI-only parallelized code on all 28 cores of a dual-socket Intel Xeon E5-2690 v4 (Broadwell) system running at 2.6 GHz to present the memory bound sparse matrix kernels in a fair way as compared to the more compute-heavy algorithms developed in this contribution. The implementation makes use of the optimized sum factorization kernels available in the `deal.II` library [5,6] and uses the Trilinos Epetra library for the sparse matrix-vector products with the trace matrix. Our test problem is equation (1) with solution $u(x, y, z) = \sin(10x)\sin(12y)\cos(10z)$, diffusion $\kappa = 1$, and convection $\boldsymbol{c}(x, y, z) = (-y, x, 0)$. The right hand side function f is set such that the given analytical solution is obtained. We solve the problem on half a spherical ball of radius 1 in the positive hemisphere $z \geq 0$ with Dirichlet conditions set on $\{z = 0\}$ and Neumann conditions on the spherical surface. We also run a 2D experiment with the z component ignored on a half circle.

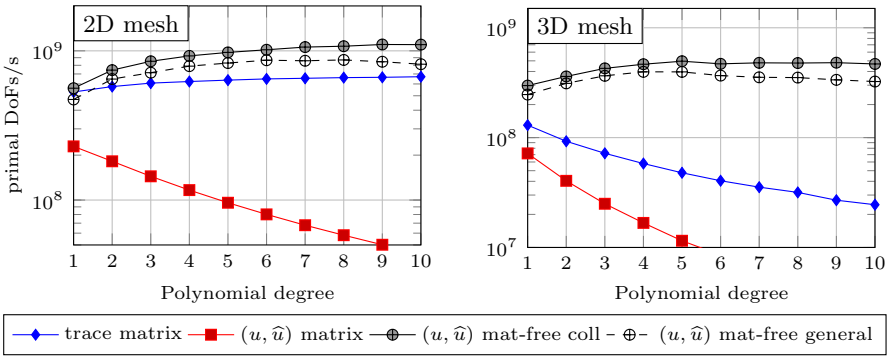

Fig. 1. Performance of one matrix-vector product: Number of degrees of freedom processed per second on 28 Broadwell cores.

Fig. 1 displays the computational throughput on experiments recorded at around 10 million unknowns with varying k. The throughput is measured as the number of primal degrees of freedom, $(k+1)^d n_{\text{elements}}$, per second (primal DoFs/s) to make the HDG matrix system (6) in the trace comparable to the larger $(u, \widehat{u})$ system. The 2D results show a similar performance in the matrix-free implementation as compared to the sparse matrix for the trace. In 3D, however, the matrix-free implementation is more than an order of magnitude faster already for $k = 5$. The gap in performance is explained by computer architectural properties: the matrix-free implementation replaces the memory-bound sparse matrix-vector product by a less memory-intensive algorithm and achieves a higher arithmetic intensity [6]. The fact that the gap widens as k is increased might seem surprising since both the trace matrix and the matrix-free implementation have a complexity of $\mathcal{O}((k+1)^4)$ per cell or $\mathcal{O}(k+1)$ per primal degree of freedom. The explanation is that the matrix-free implementation is dominated by (face) terms of cost $\mathcal{O}(k^3)$ for the considered range of polynomial degrees, see the cost estimates in Table 1, a fact also observed in matrix-free implementations for continuous and discontinuous elements [5,6]. Fig. 1 also shows that the matrix-free formulation in a generic basis is 15–40% slower than the specialized version for collocated node and quadrature points. Furthermore, our experiments illustrate that a matrix formulation for $(u, \widehat{u})$ instead of the matrix-free kernels is significantly slower than the trace matrix. In other words, the $(u, \widehat{u})$ formulation demands a matrix-free implementation. Finally, the throughput at up to $5 \cdot 10^8$ primal DoFs/s with the 3D matrix-free implementation on a general mesh is similar to the performance recorded for the symmetric interior penalty method [8].

The significant potential of matrix-free HDG implementations revealed by the experiments in this work motivate the development of iterative solver schemes that can utilize the vastly faster matrix-vector products for the $(u, \widehat{u})$

formulation. Novel developments are necessary because multigrid techniques established for other formulations [8] are not directly applicable.

Acknowledgments

This work was supported by the German Research Foundation (DFG) under the project "High-order discontinuous Galerkin for the exa-scale" (ExaDG) within the priority program "Software for Exascale Computing" (SPPEXA), grant agreement no. KO5206/1-1, KR4661/2-1 and WA1521/18-1.

References

1. D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells, *The `deal.II` library, version 8.5*, J. Numer. Math. **25** (2017), 137–145.
2. B. Cockburn, *Static condensation, hybridization, and the devising of the HDG methods*, in Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations (G. R. Barrenechea, F. Brezzi, A. Cangiani, E. H. Georgoulis, eds.), Springer, (2016), 129–177.
3. B. Cockburn, J. Gopalakrishnan, and R. Lazarov, *Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic equations*, SIAM J. Numer. Anal. **47** (2009), 1139–1365.
4. R. M. Kirby, S. J. Sherwin, and B. Cockburn, *To CG or to HDG: A comparative study*, J. Sci. Comput. **51** (2012), 183–212.
5. M. Kronbichler and K. Kormann, *A generic interface for parallel cell-based finite element operator application*, Comput. Fluids, **63** (2012), 135–147.
6. ———, *Fast matrix-free evaluation of discontinuous Galerkin finite element operators*, arXiv preprint arXiv:1711.03590 (2017).
7. M. Kronbichler, S. Schoeder, C. Müller, and W. A. Wall, *Comparison of implicit and explicit hybridizable discontinuous Galerkin methods for the acoustic wave equation*, Int. J. Numer. Meth. Eng. **106** (2016), 712–739.
8. M. Kronbichler and W. A. Wall, *A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers*, arXiv preprint arXiv:1611.03029 (2016).
9. R. E. Lynch, J. R. Rice, D. H. Thomas, *Direct solution of partial difference equations by tensor product methods*, Numer. Math. **6** (1964), 185–199.
10. N. C. Nguyen, J. Peraire, and B. Cockburn, *An implicit high-order hybridizable discontinuous Galkerin method for linear convection–diffusion equations*, J. Comput. Phys. **228** (2009), 3232–3254.
11. S. A. Orszag, *Spectral methods for problems in complex geometries*, J. Comput. Phys. **37** (1980), 70–92.
12. S. Yakovlev, D. Moxey, R. M. Kirby, and S. J. Sherwin, *To CG or to HDG: A comparative study in 3D*, J. Sci. Comput. **67** (2016), 192–220.