Efficiency of high-order elements for continuous and discontinuous Galerkin methods

Antonio Huerta[1,2,*], Aleksandar Angeloski[1], Xevi Roca[3] and Jaime Peraire[3]

[1]*Laboratori de Calcul Numeric (LaCaN). Departament de Matematica Aplicada III*
E.T.S. de Ingenieros de Caminos, Canales y Puertos,
Universitat Politecnica de Catalunya, BarcelonaTech, 08034 Barcelona, Spain.
e-mail: {antonio.huerta,aleksandar.angeloski}@upc.edu, web http://www.lacan.upc.edu
[2]*Civil & Computational Engineering Centre, College of Engineering,*
Swansea University, Swansea SA2 8PP, UK.
[3]*Department of Aeronautics and Astronautics,*
Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.
e-mail: {xeviroca,peraire}@mit.edu, web http://raphael.mit.edu

SUMMARY

To evaluate the computational performance of high-order elements, a comparison based on operation count is proposed instead of runtime comparisons. More specifically, linear versus high-order approximations are analyzed for implicit solver under a standard set of hypotheses for the mesh and the solution. Continuous as well as discontinuous Galerkin methods are considered in two-dimensional and three-dimensional domains for simplices and parallelotopes. Moreover, both element-wise and global operations arising from different Galerkin approaches are studied. The operation count estimates show, that for implicit solvers, high-order methods are more efficient than linear ones.

Received . . .

KEY WORDS: high-order elements, efficiency, work estimates, computational cost

1. INTRODUCTION

In recent years, there has been an interest and no minor discussions on the benefits and utility of high-order methods compared to low order-ones. In fact, the interest is concentrated in comparing the efficiency of linear versus high-order elements, see for instance [1–10]. Low versus high-order elements are compared taking into account the errors and the work estimates in [4]. This comparison based on asymptotic estimates clearly favors linear elements because it studies continuous Galerkin with no static condensation CG(NSC) and the number of p-order elements for a given accuracy is overestimated by a factor of $[(p+1)!]^{d/(p+1)}$. This last issue is addressed in [11]. In [5] continuous Galerkin (CG) —with static condensation, thus with a small Schur complement system element-by-element— and hybridizable discontinuous Galerkin (HDG) [12–16] are compared. This comparison is based on runtime evaluations and estimates for the number of degrees of freedom (DOF) and the minimal upper bandwidth for two-dimensional (2D) structured meshes, which are indicators of the

*Correspondence to: A. Huerta, Laboratori de Càlcul Numèric (LaCàN), E.T.S. Ingenieros de Caminos, Universitat Politècnica de Catalunya, Jordi Girona 1, E-08034 Barcelona, Spain.

Contract/grant sponsor: European Commission & Ministerio de Educación y Ciencia & Generalitat de Catalunya AGAUR; contract/grant number: ATCoMe Initial Training Network, Seventh Framework Program (FP7) & DPI2011-27778-C02-02 & 2009SGR875

computer cost. Moreover, the approximation for the number of DOF (ndof) of CG as the product of the number of edges by the interpolation order, clearly overestimates ndof because all the interior mesh vertices are counted twice for quadrilaterals and three times for triangles. This penalizes more heavily linear approximations for CG. Optimal values for h and p for the spectral element method and operation count for evaluating bi-linear forms on 2D meshes are proposed in [1–3]. Similarly, in the context of cost estimation for different approaches for computing bi-linear forms, operation counts for CG on 2D meshes has been given in [17].

Here, in order to compare high versus low-order approximations, in a manner which is implementation and hardware independent, operation count is proposed. Obviously, as noted in [18, Sc 1.2] "Operation counts alone may no longer be as important as they once were in gauging the efficiency of an algorithm... An algorithm that lends itself to parallelism may have a higher operational count but might nevertheless run faster on a parallel machine than an algorithm with a lesser operational count that cannot take advantage of parallelism." Moreover, the final runtime is also very much dependent on the memory access, which can be very different between linear and high-order approximations. Thus, complexity is not directly related to cost. Nevertheless, if any, these two issues (viz. parallelism and memory access) are prone to favor high-order elements because of the density induced by high-order approximations and the new hardware in the market. In any case, to avoid controversial arguments related to hardware or implementation here operation counts are considered and only some minor remarks are made when parallelization is obvious.

Moreover, to further justify the operation count, it is important to note that for a given algorithm the number of memory operations and floating point operations is determined by the dimensions of the input and output data. Thus, the ratio of required memory operations over the number of floating point operations, $\mathtt{R}_{\text{m/f}}$, is determined by the algorithm and the dimensions of the input and output data.

Note that, on one hand, computing bound algorithms, i.e. $\mathtt{R}_{\text{m/f}} \ll 1$, should be evaluated in terms of floating point operations (FLOPS). The operation counts proposed here are thus pertinent for these algorithms. Whereas, on the other hand, memory bound algorithms, i.e. $\mathtt{R}_{\text{m/f}} \gg 1$, should be measured in terms of memory operations per second (MOPS). Nevertheless, for a fixed computing platform there is a peak performance for memory operations (peak MOPS) and floating point operations (peak FLOPS). Therefore, the ratio of peak MOPS over peak FLOPS, $\mathtt{R}_{\text{M/F}}$, is a constant determined by the hardware. This constant, which is $\mathtt{R}_{\text{M/F}} < 1$, is decreasing because today's hardware is more efficient with floating point operations than with memory operations. In summary, it is important to point out that once the performance of a memory bound algorithm is measured in FLOPS one can convert it easily to MOPS, since for a given input data and algorithm $\text{MOPS} = \min(\mathtt{R}_{\text{m/f}}, \mathtt{R}_{\text{M/F}})\,\text{FLOPS}$.

The comparison is performed for an implicit solver (iterative and direct) typically associated to a second order differential operator and standard hypotheses of smoothness and structured uniform meshes. To have a fair comparison both local (element-by-element) and global operations are considered. This enables considering all major cost contributors: solving the global system, generating the element matrices, removing the inner degrees of freedom (DOF), viz. static condensation, and, of course, their recovery. Besides, the mesh type encompasses simplices and parallelotopes in two-dimensions (2D) and three-dimensions (3D). And finally, it is also important to note that different Galerkin approaches are considered: CG —with static condensation—, compact discontinuous Galerkin (CDG) as proposed in [19] because it presents the smallest stencils, and HDG. Note that the operation count also takes into account element-by-element operations required in CG and HDG. Results for CG(NSC) are also presented for completeness. However, this method is not recommended to implement high-order methods because its unnecessary overhead induced by the interior nodes. Comparisons are plotted up to degree ten because performance and behavior are already clear.

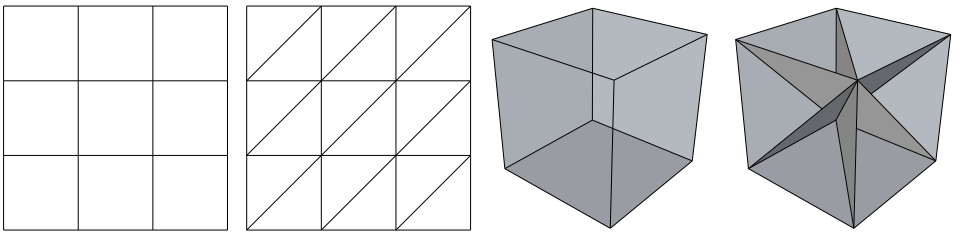

Figure 1. Splitting parallelotopes (quadrilaterals and hexahedra) in simplices (triangles and tetrahedra).

2. COMPUTATIONAL COSTS

Cost estimates are not trivial, they depend on a large number of factors. Here however, only three factors are retained, namely:

1. the Galerkin method employed: CG, CG(NSC), CDG, and HDG;
2. the mesh type: simplices or parallelotopes in 2D and 3D; and,
3. the degree p of the polynomial approximation.

Basic indicators (number of DOF, number of non-zeros per row, and number of non-zero entries in the matrix) are determined as functions of the three previous factors. Given these indicators, cost estimates of element-wise and global operations are presented in this section. These estimates evaluate all the number of floating point operations (FLOPS) including every constant in contrast with standard asymptotic estimates which neglect them. This is crucial because there is no need to be in the asymptotic range and it allows comparing costs for small values of the different parameters, for instance p.

2.1. Hypotheses and auxiliary quantities

A key issue in comparing high versus low-order elements is the estimate of the number of elements for a given tolerance, which obviously depends on the interpolation order p. Here, expressions for the number of elements are determined for quadrilateral, triangular, hexahedral and tetrahedral meshes.

Standard hypotheses for this kind of analysis are employed. The mesh is assumed structured, uniform, of dimension, d, two or three, and large (having a number of boundary faces negligible compared with the number of interior ones, i.e. boundary influence is negligible). Given a structured quadrilateral (hexahedral) mesh, the corresponding triangular (tetrahedral) mesh is obtained by dividing each quadrilateral (hexahedron) in two triangles (six tetrahedra), see Figure 1. In what follows, the number of elements and faces is denoted by n_e and n_f, respectively. In addition, the number of DOF per element and face are identified respectively by $\texttt{ndof}_e$ and $\texttt{ndof}_f$. Note that, $\texttt{ndof}_e$ and $\texttt{ndof}_f$ do not take into account if the degrees of freedom are shared (continuous Galerkin) or not (discontinuous Galerkin) between different mesh entities. That is, herein $\texttt{ndof}_e$ and $\texttt{ndof}_f$ refer to the number of nodes that appear independently on each element and each face of the mesh, respectively. Moreover, bear in mind that throughout this paper only scalar solutions are considered and, consequently, the $\texttt{ndof}$ coincides with the number of nodes. Finally, the solution is supposed smooth (i.e. bounded solution with bounded derivatives) and such that the approximation error is controlled by the interpolation one. Note however, that such an assumption can be relaxed strengthening the case for high-order approximations, see Remark 1.

To study a cartesian d-dimensional domain, $[-\ell/2, \ell/2]^d \subset \mathbb{R}^d$, a 1D problem is first considered. The bound of the interpolation error, $R_p(x)$, for an element of order p is prescribed to a given precision ϵ, namely

$$R_p(x) \leq \frac{A_p}{(p+1)!}\Big(\frac{h_p}{2}\Big)^{p+1} = \epsilon,$$

where the hypothesis of bounded solution with bounded derivatives is used, i.e. $|f^{p+1)}(\xi)| \leq A_p \leq A$. Since the inverse of the mesh size field is the element density, the number of elements for a quadrilateral/hexahedral mesh is obtained integrating the element density for each dimension of length ℓ and then consider the d-dimensional Cartesian domain :

$$n_{e,p} = \ell^d h_p^{-d}(\epsilon) = \ell^d 2^{-d} A_p^{d/(p+1)} \big[\epsilon(p+1)!\big]^{-d/(p+1)}. \tag{1}$$

For the particular case of function $f(x_1, \dots, x_d) = \sin(2\pi x_1/\lambda) \cdots \sin(2\pi x_d/\lambda)$ as used in [1–4], the bound for each dimension is simply $A_p = (2\pi/\lambda)^{p+1}$ and (1) becomes:

$$n_{e,p} = \pi^d k^d \big[\epsilon(p+1)!\big]^{-d/(p+1)}, \tag{2}$$

where $k = \ell/\lambda$ is the number of wave-lengths along a domain dimension. Given the number of parallelotopes, and assuming the same precision for parallelotopes and simplices, the number of simplices is calculated multiplying by 2 and 6 in 2D and 3D, respectively, see Figure 1. Moreover, the number of faces of the mesh can be expressed in terms of the number of elements. Namely,

$$n_{f,p} = \begin{cases} d\, n_{e,p} & \text{for paralletopes,} \\ (d+1) n_{e,p}/2 & \text{for simplices.} \end{cases} \tag{3}$$

Information on the number of elements and/or faces for different approximations p is important and shows, for instance from (2) that the ratio of number of elements,

$$n_{e,1}/n_{e,p} = 2^{-d/2} \epsilon^{-(d/2)(p-1)/(p+1)} \big((p+1)!\big)^{d/(p+1)} \geq 1, \tag{4}$$

is a strict monotonically increasing function in p, which grows faster as the number of dimensions, d, increases and as the error tolerance, ϵ, decreases. Nevertheless, the most characteristic cost indicators for the global linear system arising from a specific Galerkin method are `ndof` and the number of non-zero entries (`nnz`). An average number of non-zero entries per row can be computed as `nnzpr` := `nnz`/`ndof`. This average is quite precise for structured/compact schemes as discontinuous Galerkin, both CDG and HDG, but it is only and estimate for CG because of the different connectivity of vertices and edge/face nodes. Appendix A determines the values of `ndof`, `nnz` and, consequently, `nnzpr` for meshes of simplices and parallelotopes for CG, CG(NSC), HDG and CDG. The results are summarized in Tables I and II. It is important to point out that the theoretical values obtained with the systematic derivation detailed in Appendix A have been verified. Large structured meshes in 2D and 3D for simplices and parallelotopes were generated. Then, the `ndof` and `nnz` for the different methods were explicitly counted on that mesh for several interpolation degrees. And finally, the general theoretical formulae derived in Appendix A were validated with these numerical results.

Note that apart from the subindices e and f to characterize elements and faces, the subindex g is introduced for the 3D meshes to characterize *edges*. A more systematic notation (in terms of the dimension of the entities) is employed in Appendix A, but for clarity in the main text edges/faces/elements ($g/f/e$) are used in the discussions and formulae.

In any case, `ndof`, `nnz` and `nnzpr` are just indicators of the computational cost. To obtain an accurate estimation of the cost, it is required to compute the number of floating operations. Accordingly, operation counts for the different stages involved in every Galerkin method are derived in the following sections.

Remark 1

The assumption that the approximation error is controlled by the interpolation error is not always satisfied. For example, for oscillatory solutions, such as the one chosen here and typical of the Helmholtz equation, the pre-asymptotic range is controlled by the so-called pollution error and not the interpolation one. Nevertheless, even in this case high order elements are more competitive than low order ones as shown numerically in [9, 10] and also in [20] where a hybrid discontinuous enrichment method [21] is compared with an ultra-weak variational formulation [22] and a partition of unity method [23], the three of them enriched with plane waves.

Table I. Expressions for `ndof` for different methods.

		`ndof`
Triangles	CG	$n_{e,p}\big(3\texttt{ndof}_{f,p}-5\big)/2$
	HDG	$3n_{e,p}\,\texttt{ndof}_{f,p}/2$
	CG(NSC)	$n_{e,p}\big(2\texttt{ndof}_{e,p}-3\texttt{ndof}_{f,p}+1\big)/2$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}$
Quads	CG	$n_{e,p}\big(2\texttt{ndof}_{f,p}-3\big)$
	HDG	$2n_{e,p}\,\texttt{ndof}_{f,p}$
	CG(NSC)	$n_{e,p}(\texttt{ndof}_{e,p}-2\texttt{ndof}_{f,p}+1)$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}$
Tets	CG	$n_{e,p}\big(12\texttt{ndof}_{f,p}-29\texttt{ndof}_{g,p}+23\big)/6$
	HDG	$2n_{e,p}\,\texttt{ndof}_{f,p}$
	CG(NSC)	$n_{e,p}\big(6\texttt{ndof}_{e,p}-12\texttt{ndof}_{f,p}+7\texttt{ndof}_{g,p}-1\big)/6$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}$
Hexes	CG	$n_{e,p}\big(3\texttt{ndof}_{f,p}-9\texttt{ndof}_{g,p}+7\big)$
	HDG	$3n_{e,p}\,\texttt{ndof}_{f,p}$
	CG(NSC)	$n_{e,p}(\texttt{ndof}_{e,p}-3\texttt{ndof}_{f,p}+3\texttt{ndof}_{g,p}-1)$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}$

Table II. Expressions for `nnz` for different methods.

		`nnz`
Triangles	CG	$n_{e,p}\big(15\texttt{ndof}_{f,p}^2-36\texttt{ndof}_{f,p}+19\big)/2$
	HDG	$15n_{e,p}\,\texttt{ndof}_{f,p}^2/2$
	CG(NSC)	$n_{e,p}\big(2\texttt{ndof}_{e,p}^2-3\texttt{ndof}_{f,p}^2+1\big)/2$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}(\texttt{ndof}_{e,p}+3\texttt{ndof}_{f,p})$
Quads	CG	$n_{e,p}\big(14\texttt{ndof}_{f,p}^2-32\texttt{ndof}_{f,p}+17\big)$
	HDG	$14n_{e,p}\,\texttt{ndof}_{f,p}^2$
	CG(NSC)	$n_{e,p}(\texttt{ndof}_{e,p}^2-2\texttt{ndof}_{f,p}^2+1)$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}(\texttt{ndof}_{e,p}+4\texttt{ndof}_{f,p})$
Tets	CG	$n_{e,p}\big(84\texttt{ndof}_{f,p}^2-288\texttt{ndof}_{f,p}\,\texttt{ndof}_{g,p}+$ $+223\texttt{ndof}_{g,p}^2+192\texttt{ndof}_{f,p}-288\texttt{ndof}_{g,p}+95\big)/6$
	HDG	$14n_{e,p}\,\texttt{ndof}_{f,p}^2$
	CG(NSC)	$n_{e,p}\big(6\texttt{ndof}_{e,p}^2-12\texttt{ndof}_{f,p}^2+7\texttt{ndof}_{g,p}^2-1\big)/6$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}(\texttt{ndof}_{e,p}+4\texttt{ndof}_{f,p})$
Hexes	CG	$3n_{e,p}\big(11\texttt{ndof}_{f,p}^2-48\texttt{ndof}_{f,p}\,\texttt{ndof}_{g,p}+$ $+49\texttt{ndof}_{g,p}^2+32\texttt{ndof}_{f,p}-64\texttt{ndof}_{g,p}+21\big)$
	HDG	$33n_{e,p}\,\texttt{ndof}_{f,p}^2$
	CG(NSC)	$n_{e,p}\big(\texttt{ndof}_{e,p}^2-3\texttt{ndof}_{f,p}^2+3\texttt{ndof}_{g,p}^2-1\big)$
	CDG	$n_{e,p}\,\texttt{ndof}_{e,p}\big(\texttt{ndof}_{e,p}+6\texttt{ndof}_{f,p}\big)$

2.2. Cost of element-wise operations

To perform a proper cost comparison it is important to evaluate operations performed at element level. For instance, the cost of generating the matrices and, if the method requires it, the cost of removing the interior DOF.

On the one hand, the cost of generating the elemental matrices is determined by the number of elements times the cost of generating one of them. Note that each one of the matrix entries corresponds to an integral that is performed by means of a numerical quadrature. Therefore, the computational cost of computing one elemental matrix is determined by the number of matrix entries (number of test functions times number of trial functions) and the number of quadrature points. The number of quadrature points must be in accordance with the integrand. Note that a general integrand for an elemental matrix entry is composed by four terms:

1. the test function, degree p, or its derivative, degree $p-1$,
2. the trial function, degree p, or its derivative, degree $p-1$,
3. the Jacobian of the isoparametric mapping for curved, degree $d(p-1)$, or straight-sided, degree 0, elements, and
4. an optional accompanying function, approximated with a polynomial of degree p, typical in nonlinear problems such as the flux of a conservation law.

Thus, it is required that the numerical quadrature integrates exactly polynomials of degree equal to the sum of the degrees of the four terms. Thus, the cost of generating the elemental matrix is determined by the computation technique that depends on the element type: curved, straight-sided, simplex or parallelotope. In the following two sections, the computational cost for elemental matrices with and without a flux function is derived according to the computation technique and the number of quadrature points.

On the other hand, the cost of removing the interior DOF corresponds to first parameterize the inner DOF in terms of those on the element boundaries and then, recover the inner DOF from the computed values of the DOF on the element boundaries.

2.2.1. Elemental matrices: without flux function. In this case, representative of linear problems, mass matrices are typically among the most expensive to generate, in particular, those associated to curved elements. Three terms are in the integrand: the test function, maximum degree p, the trial function, maximum degree p, and the non-constant Jacobian, degree $d(p-1)$. Exact evaluation of mass matrices for curved element requires, consequently, to integrate exactly polynomials of degree $(d+2)p-d$. Thus, the number of quadrature points should be $\texttt{ndof}_{e,q}$ (number of DOF per element of order q), where

$$q := \left\lceil \frac{(d+2)p-d-1}{2} \right\rceil, \tag{5}$$

where $\lceil\cdot\rceil$ denotes the ceiling function (smallest integer not less than it). This guarantees exact integration of polynomials up to degree $(d+2)p-d$. Therefore, given the $\texttt{ndof}_{e,p}$ trial and $\texttt{ndof}_{e,p}$ test functions, the cost of creating an element matrix can be estimated in $\texttt{ndof}_{e,p}^2\texttt{ndof}_{e,q}$ operations. Consequently, the cost of creating all the element matrices is

$$E_p = n_{e,p}\,\texttt{ndof}_{e,p}^2\texttt{ndof}_{e,q}. \tag{6}$$

It is important to note that this cost can be reduced for straight-sided elements. In that case, the Jacobian is constant and therefore, the mass matrix can be pre-computed once. Consequently, the cost of creating all the element matrices is not (6) but

$$E_p^J = n_{e,p}\,\texttt{ndof}_{e,p}^2. \tag{7}$$

Another alternative for quadrilateral and hexahedral elements to reduce the cost described by (6) for curved elements is the sum factorization technique. This approach exploits the tensor product structure of the basis, see [1–3]. In fact, the reduction induced by the sum factorization is almost as efficient as the one corresponding to straight-sided elements. That is, considering again $\texttt{ndof}_{e,q}$ quadrature points, recall (5), but this time organized with a tensor product structure, $(q+1)^d$, the cost with sum factorization is

$$E_p^{\text{sf}} = n_{e,p}\sum_{i=0}^{d-1}(p+1)^{2(d-i)}(q+1)^{i+1}. \tag{8}$$

Note that this latter cost varies as $n_{e,p}\mathcal{O}(\texttt{ndof}_{e,p}^2(q+1))$.

2.2.2. Elemental matrices: with flux function. This, which is representative of nonlinear situations, evaluates the cost of generating matrices whose entries are composed by four terms: the one associated to test functions, maximum degree p, the trial term, maximum degree p, the non-constant Jacobian, degree $d(p-1)$, and the accompanying flux function assumed to be described by a

polynomial of degree p. Thus, polynomials of degree $(d+3)p-d$ must be exactly integrated. The number of quadrature points must be $\texttt{ndof}_{e,\hat{q}}$ (number of DOF per element of order $\hat{q}$), where

$$\hat{q} := \left\lceil \frac{(d+3)p-d-1}{2} \right\rceil.$$

This guarantees exact integration of polynomials up to degree $(d+3)p-d$. Consequently, the cost of creating all the element matrices is

$$E_p^{\text{fl}} = n_{e,p}\,\texttt{ndof}_{e,p}^2 \texttt{ndof}_{e,\hat{q}}. \tag{9}$$

It is important to note that this cost can be reduced for straight-sided elements. In that case, the Jacobian is constant and therefore, it requires degree 0. That is, the total degree of the integrand is $3p$. Then, the cost of creating all the element matrices is not (9) but

$$E_p^{J,\text{fl}} = n_{e,p}\,\texttt{ndof}_{e,p}^2 \texttt{ndof}_{e,\hat{q}^J}, \tag{10}$$

where $\hat{q}^J := (3p-1)/2$ to ensure integrating exactly polynomials of degree $3p$.

For elemental matrices with an accompanying function the sum factorization technique can also be applied. Considering again $\texttt{ndof}_{e,\hat{q}}$ quadrature points, but this time organized with a tensor product structure, $(\hat{q}+1)^d$, the cost with sum factorization is

$$E_p^{\text{sf,fl}} = n_{e,p} \sum_{i=0}^{d-1} (p+1)^{2(d-i)} (\hat{q}+1)^{i+1}. \tag{11}$$

Note that this latter cost varies as $n_{e,p}\mathcal{O}(\texttt{ndof}_{e,p}^2(\hat{q}+1))$.

2.2.3. Removing and recovering inner DOF. The ratio between interior and boundary nodes increases as the order increases. If possible, high-order methods parameterize the DOF in the interior of the elements in terms of the DOF on the faces. This results in a reduced global system that only depends on the DOF on the faces of the elements. This technique corresponds to the well-known static condensation in CG and the solution of the local problem in HDG. For an element, the cost of this parameterization is dominated by the cost of inverting a dense elemental matrix of dimension $\texttt{ndof}_{e,p}$. Using a Gauss-Jordan method to compute the inverse of a dense matrix takes $\texttt{ndof}_{e,p}^3$ multiplications/divisions and $\texttt{ndof}_{e,p}^3 - 2\texttt{ndof}_{e,p}^2 + \texttt{ndof}_{e,p}$ additions/subtractions, see [18, Sc 3.7]. Note that in this case matrix inversion is justified because of the system dimension, condition and, more important, the fact that the inverse operator is going to be applied to several matrices and vectors. In addition, the application of many inverse operators (one per element) can be casted to a generalized tensor contraction that can obtain a significant efficiency in multi-core CPU and in vector processors (GPU), see [24, 25]. It is important to point out that the computation of an LU factorization is three times less expensive than the inversion of a matrix. However, its application to several matrices and vectors does not feature the same level of parallelism of the explicit inverse. That is, applying an inverse operator through an LU factorization requires a forward and a backward substitution, both with a sequential nature, and therefore, does not allow a fine grain level of parallelism. Nevertheless, all the conclusions of this work do not change if an LU factorization is used, since both implementations require a similar computational cost. Thus, the cost of the static condensation for all elements is

$$L_p = n_{e,p}(2\texttt{ndof}_{e,p}^3 - 2\texttt{ndof}_{e,p}^2 + \texttt{ndof}_{e,p}). \tag{12}$$

Note also that, in general, the approximation inside the element is also determined and, consequently, once the solution is known at the element boundary a recovery process is done to evaluate the inner DOF. The parametrization computed previously is used and the cost is dominated by the multiplication of dense matrices with vectors of face DOF. These dense matrices are of order $(d+1)\texttt{ndof}_{f,p}\,\texttt{ndof}_{e,p}^2$ for simplices and $2d\,\texttt{ndof}_{f,p}\,\texttt{ndof}_{e,p}^2$ for parallelotopes. Thus, the

recovery cost is

$$R_p = \begin{cases} n_{e,p}\,\mathtt{ndof}_{e,p}^2\big(2(d+1)\mathtt{ndof}_{f,p}-1\big) & \text{for simplices,} \\ n_{e,p}\,\mathtt{ndof}_{e,p}^2\big(4d\,\mathtt{ndof}_{f,p}-1\big) & \text{for parallelotopes.} \end{cases} \tag{13}$$

2.3. Solving the global problem

The cost of generating the matrices and solving the local (elemental) problems is important. But it is crucial to estimate the operation count for the global solve. The global system of linear equations can be solved by a direct or iterative method. Note that the expected computational cost of a real implementation should be between the theoretical cost of a direct and the one of an optimal iterative solver. Given the number of DOF and the number of non-zero entries in the matrix the cost per iteration of an iterative method can be estimated because it is dominated by the sparse matrix-vector product, and by applying the pre-conditioning operator. A typical pre-conditioner is the incomplete LU factorization with zero fill-in (ILU0); this method is chosen because its cost per iteration can be estimated. Note that the total number of iterations is not estimated since it is problem and mesh dependent. Nevertheless, the ratio of the total cost of the implicit solver could be estimated under the assumption that the ILU preconditioner leads to a constant number of iterations regardless of p, which is reasonable for convection dominated problems. In Appendix B these estimates are determined, namely

$$G_p^{iter} = \mathtt{SpMV} + \mathtt{SpFS} + \mathtt{SpBS} = 4\mathtt{nnz} + \mathtt{ndof}. \tag{14}$$

Estimating the cost of a direct method is largely dependent on the structure of the sparse matrix, which is in general difficult to determine. However, it is important to note that HDG induces for any mesh a sparse matrix with uniform pattern (North, South, East and West for 2D and the corresponding in 3D) of dense square blocks (of dimension the $\mathtt{ndof}_{f,p}$ in each face). The HDG linear system has this structure because global DOF are only on the faces, DOF of two different faces are connected only if they belong to the same element, and each mesh face contains several nodes. Consequently, each face is connected to a constant number of faces per element for uniform meshes of simplices or parallelotopes, and when two faces are connected all nodes in these faces are interconnected (dense block matrices). This structure in the connections between faces is equivalent to the connections between nodes of a structured linear CG mesh. Specifically, the equivalent structured CG mesh is obtained by substituting each mesh face by a node, and each connection between faces by an edge. This equivalence between a HDG and a structured CG mesh allows estimating the cost of solving the HDG linear system. That is, the HDG linear system is equivalent to solve a structured CG linear system, where the scalar entries for the nodes have been substituted by dense matrices of size $\mathtt{ndof}_{f,p} \times \mathtt{ndof}_{f,p}$. This equivalent linear system can be solved in two steps by means of a sparse direct solver. First, the faces have to be reordered (instead of the mesh nodes). Second, the dense blocks (instead of scalar entries) are added, subtracted, multiplied and inverted to perform the operations induced by the direct solver. Note that the most expensive operation is the inversion of the dense blocks.

In [26, 27] the number of FLOPS required to solve a linear system arising from a structured CG mesh is determined assuming that nested dissection is used to renumber nodes. The resulting cost is $\mathtt{ndof}^{(d+1)/2}$ where $\mathtt{ndof}$ is the number of DOF of the scalar equation. Taking into account this cost and the equivalence between a HDG and a structured CG linear system, it is possible to estimate the cost of solving the HDG system. To this end, it is required to assume that the mesh faces (instead of nodes for the CG mesh case) are renumbered using nested dissection; then use the relation between the number of faces and number of elements given in (3) to estimate the cost in terms of the number of elements (instead of the number of nodes for the CG mesh case); and finally recall the cost of inverting a dense matrix, see (12) and [18, Sc 3.7], of size $\mathtt{ndof}_{f,p} \times \mathtt{ndof}_{f,p}$ (instead of one floating point operation for the CG mesh case). In this case, the number of FLOPS

required to solve with a direct solver the HDG linear system is:

$$G_p = \begin{cases} d^{(d+1)/2}\, n_{e,p}^{(d+1)/2}(2\texttt{ndof}_{f,p}^3 - 2\texttt{ndof}_{f,p}^2 + \texttt{ndof}_{f,p}) & \text{for paralletopes,} \\ \big((d+1)/2\big)^{(d+1)/2} n_{e,p}^{(d+1)/2}(2\texttt{ndof}_{f,p}^3 - 2\texttt{ndof}_{f,p}^2 + \texttt{ndof}_{f,p}) & \text{for simplices.} \end{cases} \tag{15}$$

3. COST COMPARISON: LINEAR VERSUS HIGH-ORDER

Once the number of FLOPS is determined for each approximation order p as a function of the number of elements, $n_{e,p}$, and number of DOF per edge/face/element, $\texttt{ndof}_{g,p}/\texttt{ndof}_{f,p}/\texttt{ndof}_{e,p}$, a comparison is possible. Note that these comparisons also depend on the number of spatial dimensions, d, the error tolerance, ϵ, the element type, simplices/parellotopes, and, obviously on the method employed.

3.1. Element-wise operations

A common characteristic for all element-wise operations is the linear dependency on the number of elements $n_{e,p}$ and a polynomial dependency on the number of DOF per edge/face/element, $\texttt{ndof}_{g,p}/\texttt{ndof}_{f,p}/\texttt{ndof}_{e,p}$ depending on the operation. Thus, the ratio of the number of FLOPS between order 1 and order p is always the product of the element ratio, see (4), and a ratio of the polynomial dependence on the $\texttt{ndof}$. The first term, as noted previously is a strict monotonically increasing function in p whereas the second is, in general, decreasing in p. The combination of these two terms indicates when the number of FLOPS is larger or smaller for linear elements compared to high-order elements.

Thus, the benefit of having less elements in high-order methods is undermined by the cost of the element-wise operation. This is alleviated when high accuracy is required, since many elements are needed in the linear mesh to achieve such accuracy. It is important to note that this element-wise work can easily be parallelized, in which case its effect to the total cost to solve the problem is significantly reduced.

Note also that element-wise FLOPS grow linearly with the number of elements, and non-linearly with the number of DOF. Thus, simplices are more efficient for these operations because they have fewer DOF.

3.1.1. Elemental matrices: without flux function. Figures 2 and 3 depict the ratios of E_1/E_p and E_1^J/E_p^J in the absence of a flux function, for the cases of curved and straight-sided elements, respectively, obtained from (6) and (7). Figure 4 shows the ratio $E_1^{\text{sf}}/E_p^{\text{sf}}$ induced by (8), which corresponds to parallelotopes implemented with sum factorization. These figures indicate that when tensorial basis or straight-sided elements are employed first-order is not always the best choice for accuracies of two or more significant digits. Note that for curved elements (or straight-sided ones but implemented without exploiting its advantages) the results are just the opposite, namely, up to engineering accuracy it is difficult to improve on linear elements.

3.1.2. Elemental matrices: with flux function. In this case, which is representative of nonlinear problems, Figures 5 and 6 depict the ratios of $E_1^{\text{fl}}/E_p^{\text{fl}}$ and $E_1^{J,\text{fl}}/E_p^{J,\text{fl}}$ in the presence of a flux function, for the cases of curved and straight-sided elements, respectively, obtained from (9) and (10). Figure 7 shows the ratio $E_1^{\text{sf,fl}}/E_p^{\text{sf,fl}}$ induced by (11), which corresponds to parallelotopes implemented with sum factorization. It is clear from these figures that the introduction of a flux function decreases the performance of high-order elements. Nevertheless, when tensorial basis or straight-sided 3D elements are used, first-order is not always the best choice for accuracies of two or more significant digits. On the contrary, for curved elements and straight-sided 2D elements up to engineering accuracy it is difficult to improve on linear elements.

3.1.3. Removing and recovering inner DOF. CG and HDG allow a parametrization of the interior nodes in terms of the ones on the faces (static condensation), which reduces considerably the number

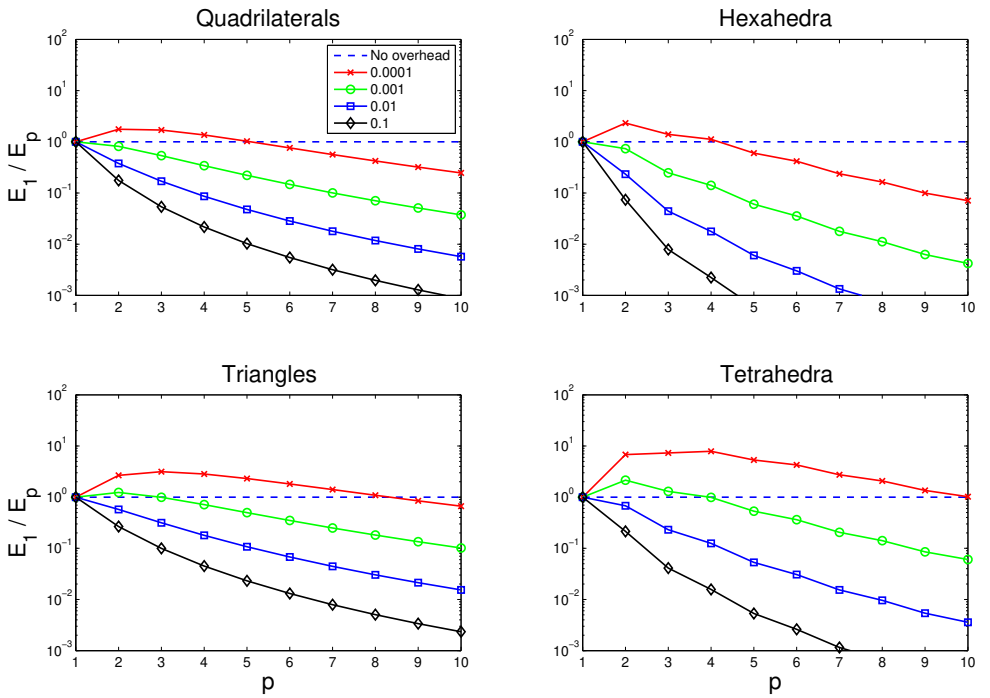

Figure 2. FLOPS ratio (linear/order p) for creating element matrices of curved elements (no tensorial basis exploitation for the parallelotopes) without flux function.

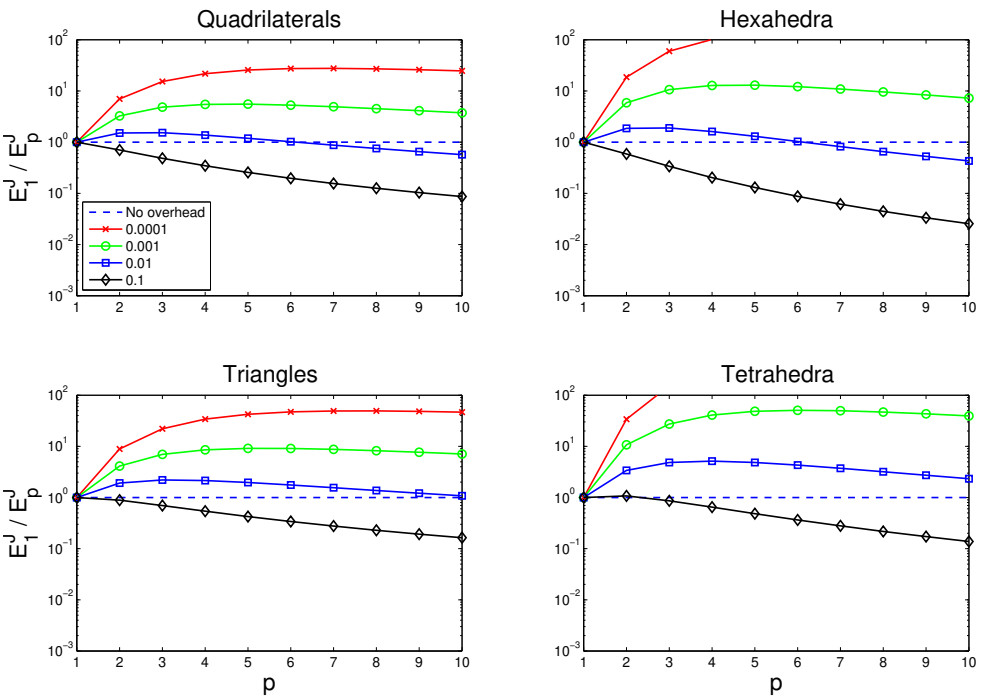

Figure 3. FLOPS ratio (linear/order p) for creating element matrices of straight-sided elements without flux function.

of DOF. Its cost, in terms of FLOPS, is indicated by (12). Moreover, these methods also require the recovery of those inner DOF once the global problem is solved, see (13). The corresponding ratios L_1/L_p and R_1/R_p are depicted in Figures 8 and 9. This static condensation, which implies a

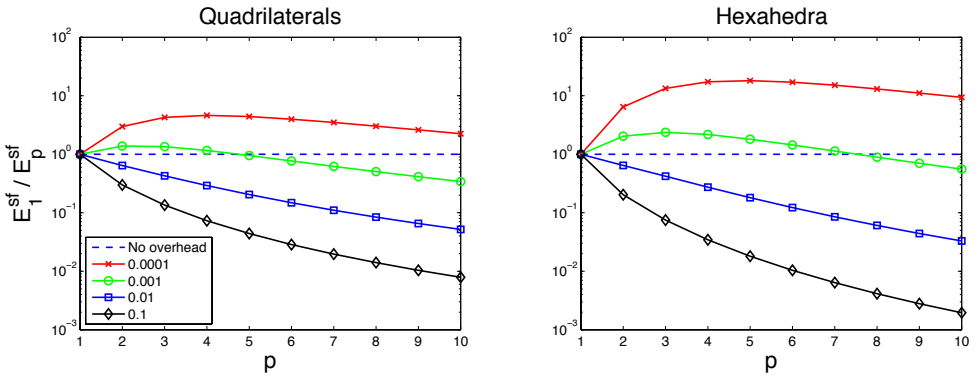

Figure 4. FLOPS ratio (linear/order p) for creating element matrices with sum factorization without flux function.

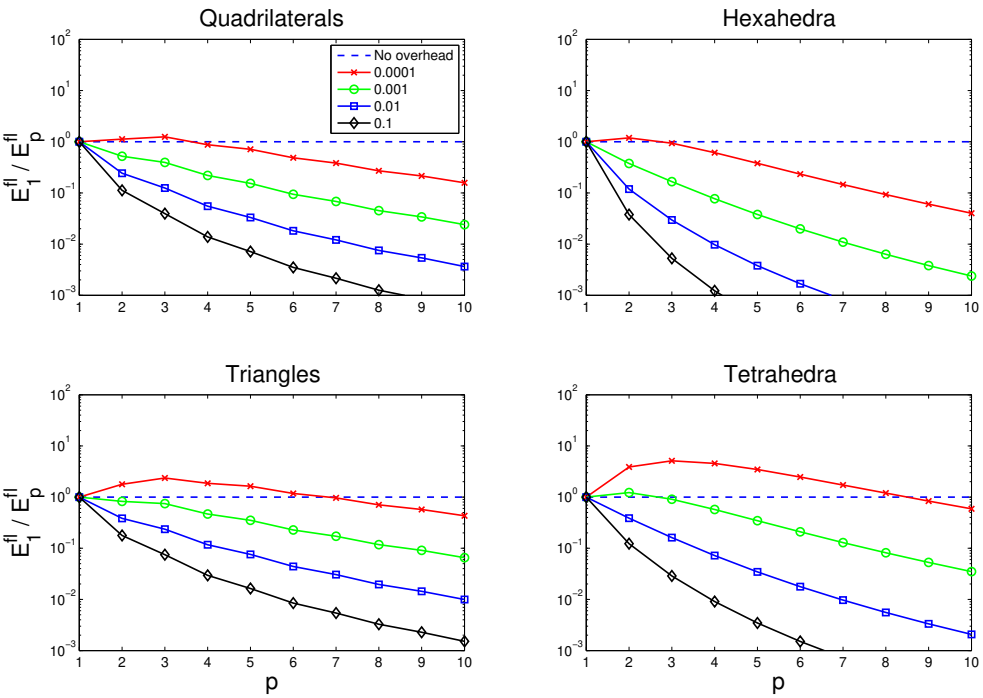

Figure 5. FLOPS ratio (linear/order p) for creating element matrices of curved elements (no tensorial basis exploitation for the parallelotopes) with a flux function.

reduction in DOF, introduces an overhead for elements of order $p > 1$, which, in general, is not compensated by the reduction in number of elements (except for accuracies higher than engineering ones). Thus, if methods exploiting static condensation are to be more competitive in higher orders they must compensate the overhead element-wise operations (easily parallelizable) of removing and recovering the inner DOF in the global solve.

3.2. Global problem operations

If an iterative solver is used with an ILU0 pre-conditioner, equation (14) indicates the number of FLOPS required in each iteration. This operation count depends on `ndof` and `nnz`, which are different for each method and mesh type, see Tables I and II. Thus a comparison for all these methods and meshes is possible. Figures 10 and 11 show the ratio of FLOPS for the four methods compared (CG, HDG, CG(NSC) and CDG) and two mesh typologies (quadrilaterals and hexahedra)

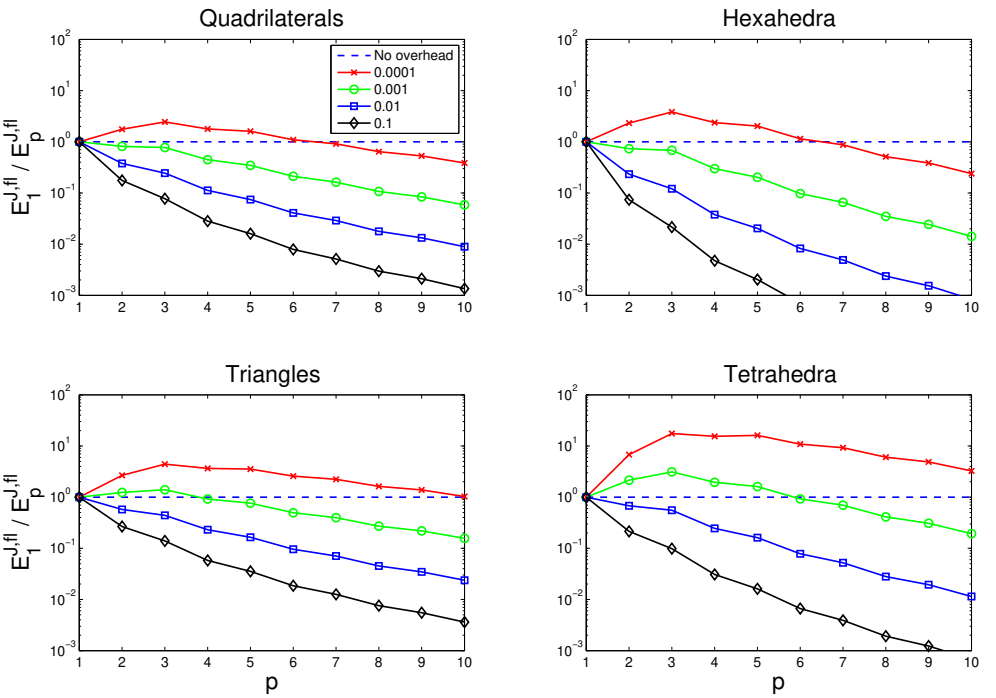

Figure 6. FLOPS ratio (linear/order p) for creating element matrices of straight-sided elements with a flux function.

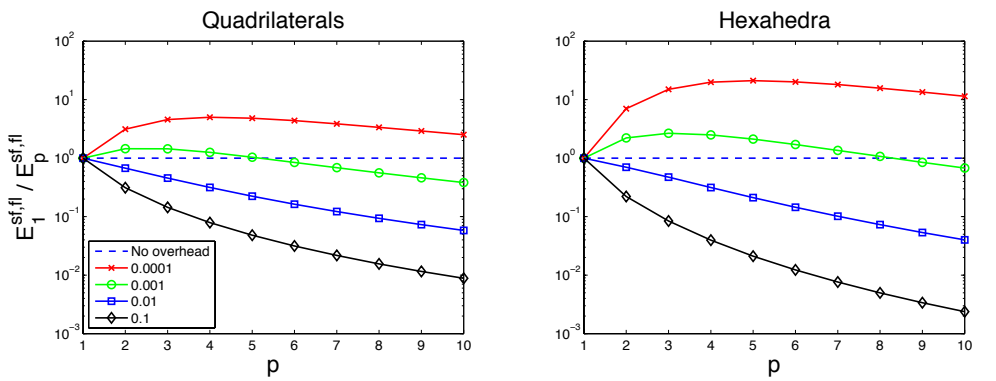

Figure 7. FLOPS ratio (linear/order p) for creating element matrices with sum factorization with a flux function.

because, as noted earlier, parallelotopes are the worst case scenario for high-order elements, see [28].

Note that CG and HDG have a better performance with high-order methods compared with CG(NSC) and CDG. This is obvious, because the number of inner nodes increases drastically with p and *static condensation* implies that `ndof` and `nnz` grow with the number of DOF on the faces, $\texttt{ndof}_{f,p}$, instead of the complete element, $\texttt{ndof}_{e,p}$, as shown in Tables I and II.

There is, however, in this comparison a major conclusion: for engineering accuracy high-order methods are more efficient in a wide range of p. Recall that CG(NSC) is only put here for comparison purposes.

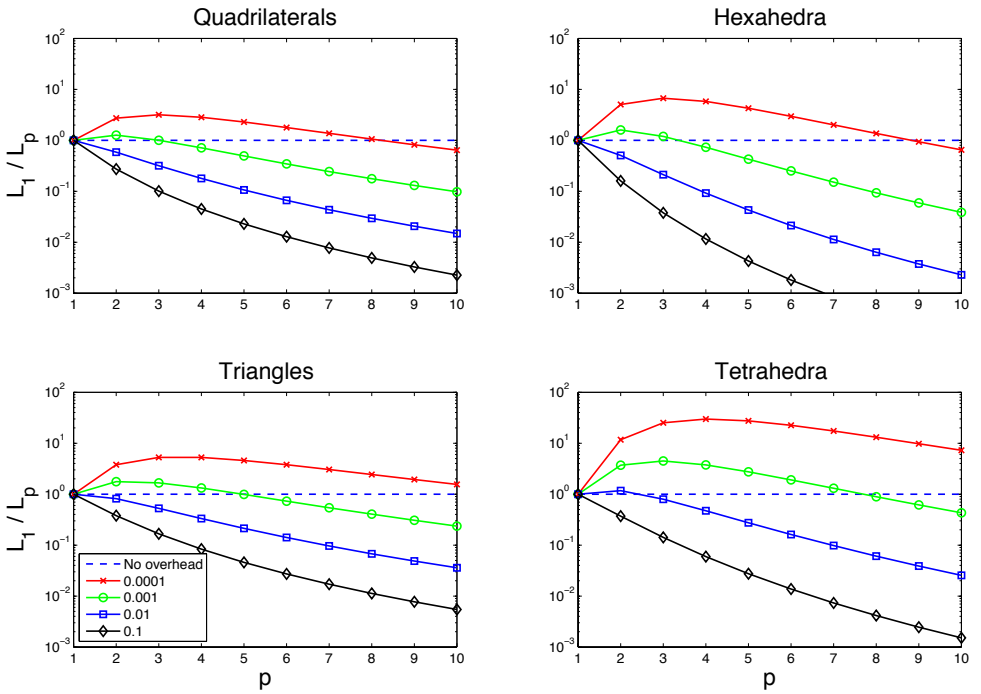

Figure 8. FLOPS ratio (linear/order p) for removing inner DOF.

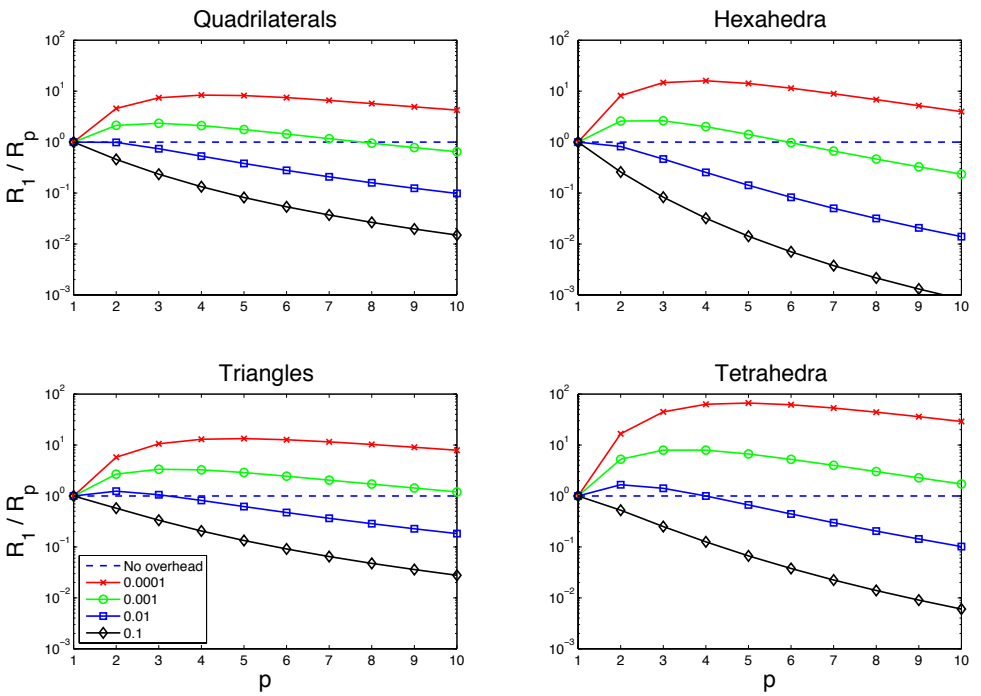

Figure 9. FLOPS ratio (linear/order p) for recovering inner DOF.

For a sparse direct solver the operation counts are only available for HDG, see (15). Thus the ratio between linear and order p becomes, for simplices and parallelotopes:

$$\frac{G_1}{G_p} = (n_{e,1}/n_{e,p})^{(d+1)/2}(2\mathtt{ndof}_{f,1}^3 - 2\mathtt{ndof}_{f,1}^2 + \mathtt{ndof}_{f,1})/(2\mathtt{ndof}_{f,p}^3 - 2\mathtt{ndof}_{f,p}^2 + \mathtt{ndof}_{f,p}).$$

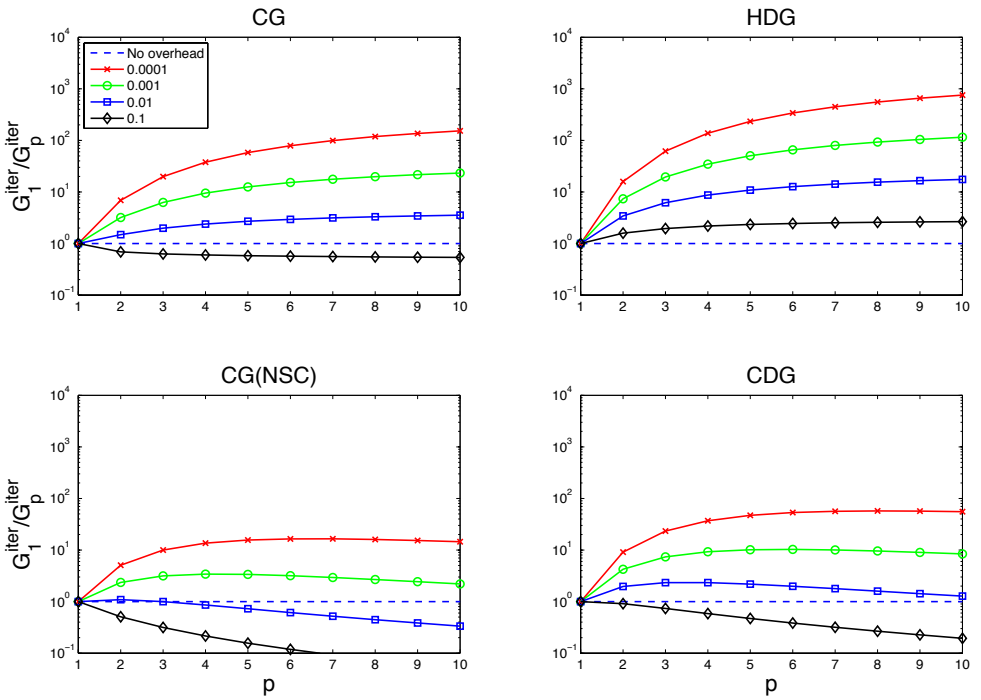

Figure 10. FLOPS ratio (linear/order p) per iteration in a global solve a quadrilateral mesh.

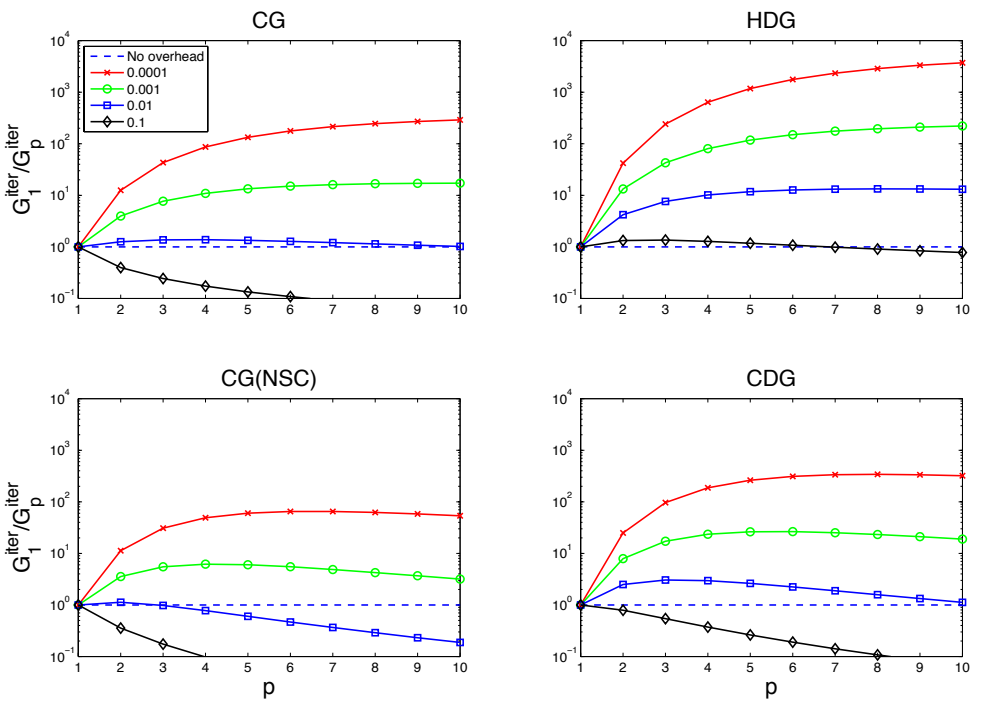

Figure 11. FLOPS ratio (linear/order p) per iteration in a global solve a hexahedral mesh.

Figure 12 is clear (and even more conclusive in 3D) high-order methods have fewer FLOPS in the global solve with HDG for engineering accuracy. The power $(d+1)/2 \leq 1$ for the strict monotonically increasing ratio of elements (in p) dominates the ratio of the inverse of a dense

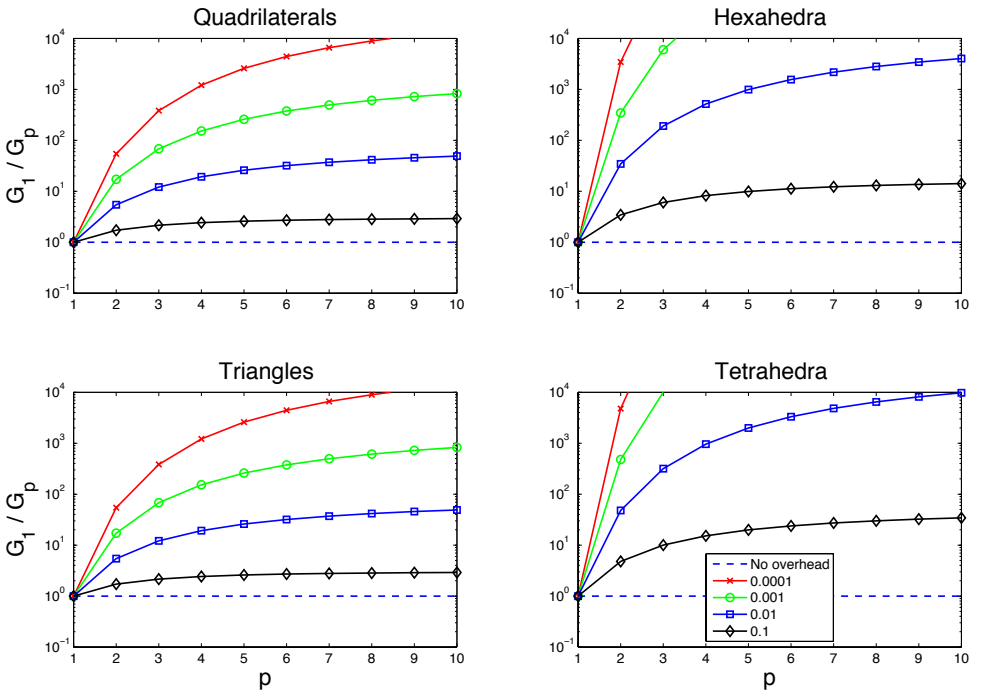

Figure 12. FLOPS ratio (linear/order p) for the global solve in HDG with a sparse direct solver after nested dissection renumbering.

matrix with DOF only in the faces. Recall that as p increases, the number of interior nodes increases faster than the number of boundary nodes in each element.

3.3. Total cost comparison

The total cost comparison requires accounting for element and global operations. A naive sum for HDG with a direct solve, see (15), under the "worst" case scenario, that is, no parallelization of element-wise operations, curved elements with no tensorial basis, in the presence of a flux function, see (9), and removing and recovering the inner DOF, equations (12) and (13), will induce the following FLOPS ratio

$$\frac{T_1}{T_p} = \frac{E_1 + L_1 + R_1 + G_1}{E_p + L_p + R_p + G_p} = \frac{n_{e,1}}{n_{e,p}} \frac{g(1,d)}{g(p,d)},$$

where $g(p,d)$ is

$$\begin{aligned} g(p,d) =& \mathtt{ndof}_{e,p}^2 \mathtt{ndof}_{e,[(d+3)p-d-1]/2} + 2\mathtt{ndof}_{e,p}^3 + \alpha_d \mathtt{ndof}_{e,p}^2 + \mathtt{ndof}_{e,p} + \\ & \beta_d n_{e,p}^{(d-1)/2} (2\mathtt{ndof}_{f,p}^3 - 2\mathtt{ndof}_{f,p}^2 + \mathtt{ndof}_{f,p}), \end{aligned}$$

and α_d and β_d are two parameters dependent on the mesh type, namely

$$\alpha_d = \begin{cases} 2(d+1)\,\mathtt{ndof}_{f,p} - 3 \\ 4d\,\mathtt{ndof}_{f,p} - 3 \end{cases} \qquad \beta_d = \begin{cases} ((d+1)/2)^{(d+1)/2} & \text{for simplices,} \\ d^{(d+1)/2} & \text{for parallelotopes.} \end{cases}$$

Figures 13 and 14 depict these comparisons for two cases: one and a hundred wavelengths per domain, $k = 1$ and $k = 100$, see (2). Note that as expected, as k grows, the global FLOPS, G_p, are dominant and high-order elements are more competitive. Nevertheless, even for the extreme case of $k = 1$ these results clearly indicate that high-order methods are more competitive, for engineering accuracies, than linear elements. This is even more evident in 3D problems. Note that any simple

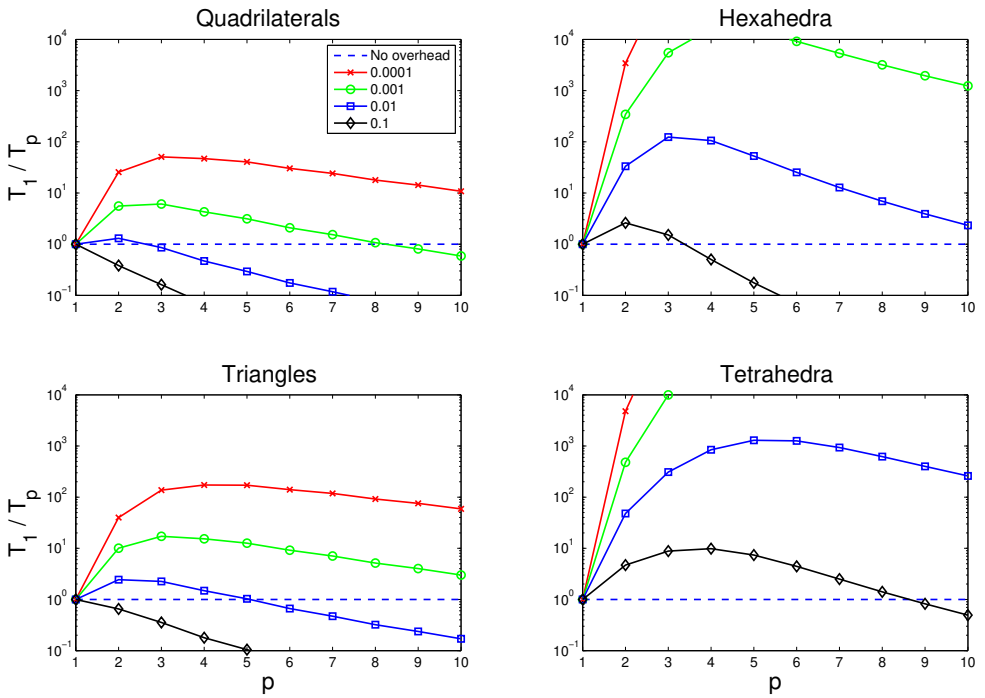

Figure 13. Total FLOPS ratio (linear/order p) for HDG with a sparse direct solver and worse case scenario for a problem with 1 wavelength per domain, $k = 1$.

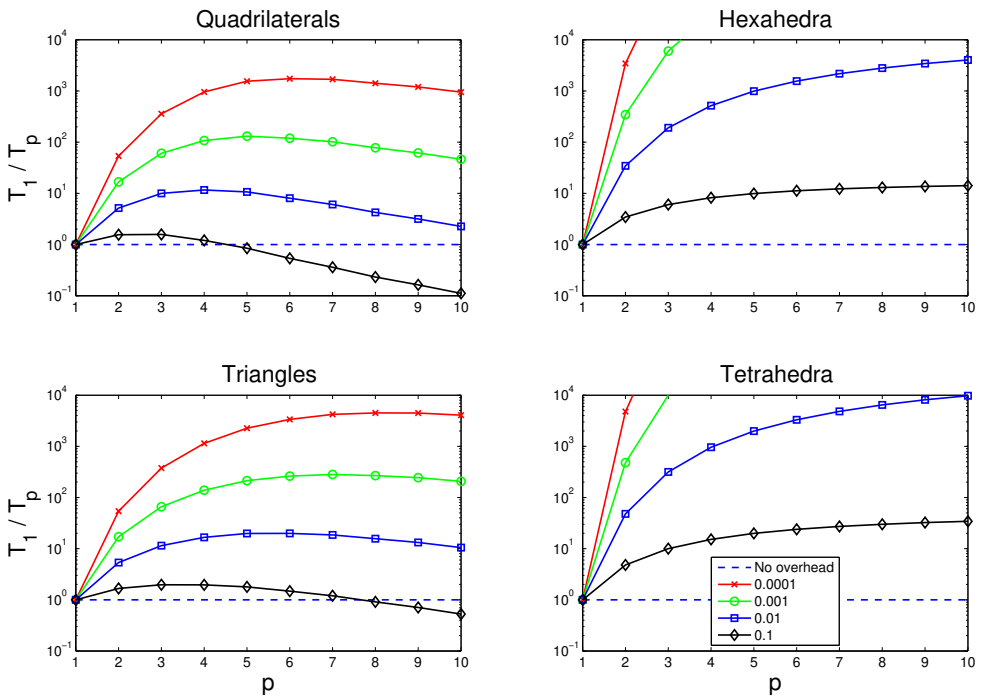

Figure 14. Total FLOPS ratio (linear/order p) for HDG with a sparse direct solver and worse case scenario for a problem with 100 wavelengths per domain, $k = 100$.

parallelization of element-wise operations will improve the performance of high-order methods. Moreover, any improvement in creating element matrices using tensorial basis or exploiting constant Jacobians, will also improve the performance of high-order elements. These results also extend for

Table III. Relative increase of `ndof` expressed in % for HDG compared to CG due to multi-valued nodes.

p	1	2	3	4	5	6	7	8	9	10
Quadrilaterals	300	100	60	43	33	27	23	20	18	16
Hexahedra	1100	286	153	103	77	62	51	44	38	34

iterative methods as soon as the number of iterations is typical of engineering problems and the global solve cost compensates the element-wise operations.

4. DISCONTINUOUS VERSUS CONTINUOUS GALERKIN METHODS

At this point where the number of DOF, `ndof`, as well as the number of non-zeros in the matrix, `nnz`, have been evaluated for different mesh typologies, it is worth comparing continuous and discontinuous methods. See [5] for another comparison.

Obviously the mesh hypotheses presented in Section 2.1 are retained. Moreover, in contrast with the previous comparison between low and high-order methods where the number of elements in the domain was changing with the order p. Now the comparison is performed for the same characteristic size h (i.e. with a constant number of elements) because the different methods have a priori error estimates of the same order.

The methods to be compared are CG, CG(NSC), CDG, and HDG. Additionally, results are also shown for the post-processed solution of HDG (pHDG). Recall, see [13, 14], that HDG presents, for elliptic operators, a super-convergent property that produces a solution with an extra order of approximation after an inexpensive element-by-element post-process. Thus, an HDG solution with order $p-1$ can be post-processed into an order p approximation.

This comparison can be done directly in terms of, for instance, `ndof`. See Table III for a comparison between HDG and CG, where the relative increase in `ndof` expressed in percentage is evaluated. This is typical of discontinuous Galerkin methods since the corresponding finite element spaces are multi-valued at element boundaries: a CDG (HDG) point inside an entity on the boundary of an element corresponds to as many nodes as elements (faces) surround the container entity, see Figure 22 (Figure 23). Specifically, CDG penalizes the inner nodes on the faces since they are double-valued (duplicated). On the contrary, HDG penalizes the nodes on the vertices since they are multi-valued (as many values as faces surround the vertex). Note that the overhead of multi-valued nodes in CDG (HDG) is more severe for higher dimensions, due to the fact that more elements (faces) are adjacent to each mesh entity, see Table V. As expected, the percentage of multi-valued nodes in the discontinuous Galerkin methods decreases as p increases, due to the decrease in the ratio between external and internal nodes in the elements as p increases. This makes DG methods less competitive for low p.

Nevertheless, here instead of comparing directly `ndof` and/or `nnz`, the comparison is centered in the the number of FLOPS required in each iteration of a global solve. The cost per iteration of a pre-conditioned (ILU0) iterative solver is given in equation (14). It is a linear combination of the `ndof` and `nnz` of the global matrix. Recall that the exact values of `ndof` and `nnz` for each method and mesh type are given in Tables I and II, respectively. Figure 15 shows these comparisons normalized by the CG results. First of all, it is important to note that CG always requires fewer FLOPS than any other method. Second, note the differences in the asymptotic behavior as p increases of the methods requiring interior nodes, CG(NSC) and CDG, and those not using them, CG, HDG and pHDG. Moreover, "static condensation" is more effective as p increases. Finally, although DG methods are always above their continuous counterpart the post-processed HDG solution, pHDG, has the smallest overhead over CG, because it obtains with a $p-1$ approximation (and computational effort) similar results (same order of the a priori error estimate) as all the others with order p, including CG. This super-convergence, however, does not reduce the number of FLOPS below CG for the mesh hypotheses used here (uniform structured with a negligible boundary influence).

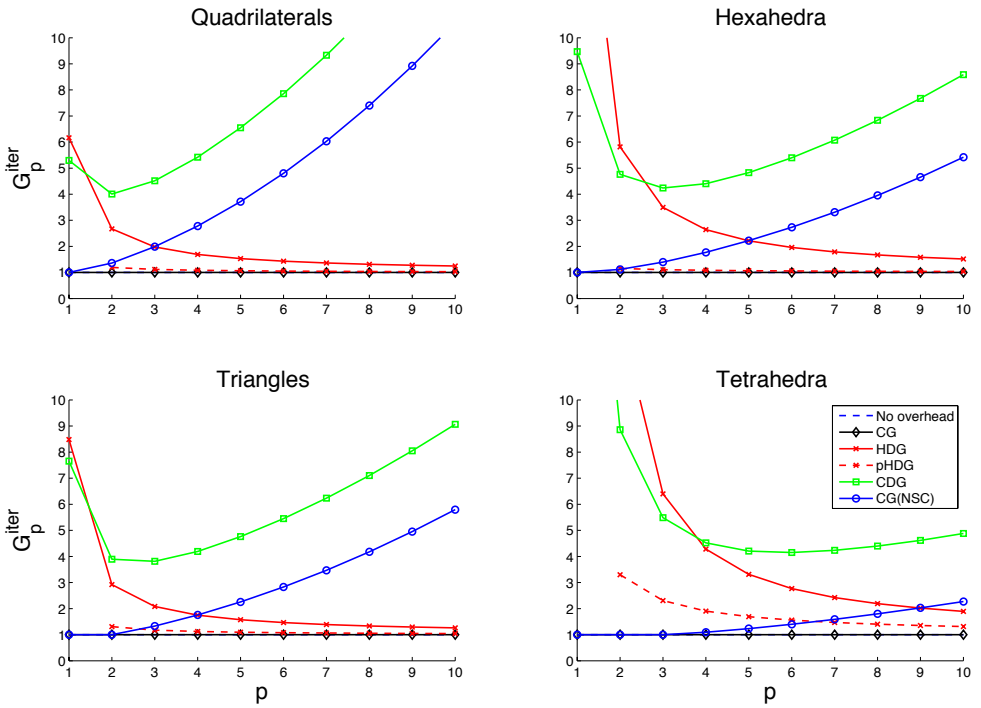

Figure 15. FLOPS ratio, normalized with respect to CG, per iteration.

5. CONCLUDING REMARKS

The comparison between linear and high-order elements presented here in terms of operation counts indicates that, under the hypotheses of the study (i.e. smooth solutions, interpolation error controlling the approximation one and large structured meshes) there is an optimal approximation order, which minimizes the computational cost. This optimal order is linear only for the particular case of element-by-element operations with a cubic dependence in the number of DOF per element (i.e. curved elements and with no tensorial basis exploitation for parallelotopes). Note that this can be the case for a large number of explicit solvers. In every other case studied the optimal $p > 1$. Nevertheless, estimating exactly the optimal value is by no means an easy task. For element-wise operations which have up to quadratic dependence on the number of DOF per element, high-order elements are more efficient for a wide range of orders and at least engineering accuracy (one or two significant digits). This better performance of high-order methods in creating the required element matrices is more evident in 3D and for simplices.

At global level, high-order elements are consistently more efficient than linear ones when confronted with the cost for solving with an iterative solver the global system arising from second order differential operators in continuous Galerkin (CG) —with static condensation yielding an element-wise and small Schur complement system—, compact discontinuous Galerkin (CDG), and hybridizable discontinuous Galerkin (HDG). Moreover, if a direct solver for HDG is employed, again high-order elements are more efficient. If both local and global operations are taken into account, high-order methods still prove to be more efficient even in the worse case scenario of no parallelization of element-wise operations.

The comparison has been also extended to discontinuous versus continuous approaches for a given accuracy (i.e. mesh discretization) where it is also shown that the only method with a cost comparable to CG is HDG, provided its super-convergent property is utilized. Of course this comparison is only made in terms of operation count and not brings to the discussion other advantages each Galerkin method may present in terms of the application, stability, error estimation and adaptivity.

Table IV. Notation related with the different types of mesh entities.

Entity	Dim	# in mesh	# nodes	# interior nodes	Connections
vertex	0	n_0	$\texttt{ndof}_0$	$\texttt{ndof}_0^{int}$	c_0
edge	1	n_1	$\texttt{ndof}_1$	$\texttt{ndof}_1^{int}$	c_1
polygon	2	n_2	$\texttt{ndof}_2$	$\texttt{ndof}_2^{int}$	c_2
cell	3	n_3	$\texttt{ndof}_3$	$\texttt{ndof}_3^{int}$	c_3

A. INDICATORS OF THE COST OF SOLVING THE GLOBAL LINEAR SYSTEM

In this section, the expressions of three indicators of the cost of solving the global linear system are obtained: the number of global degrees of freedom (ndof), the number of non-zero entries in the global matrix (nnz), and the average number of non-zero entries per row (nnzpr). Specifically, these cost indicators are obtained for different: Galerkin methods (CG, CG(NSC), HDG, and CDG), element types (simplices and parallelotopes), number of spatial dimensions ($d = 2, 3$), and interpolation degrees (p). To this end, it is assumed that the domain is discretized with a structured mesh composed by equally sized elements and that the number of boundary entities is negligible compared with the total number of elements. The final values are expressed in terms of the number of mesh elements (n_d) and the number of degrees of freedom on an i-dimensional mesh entity ($\texttt{ndof}_i$), see Section A.4. Herein, the i-dimensional mesh entities are referred as: vertices (0D), edges (1D), polygons (2D), and cells (3D).

A.1. Outline of the counting technique and notation

The goal is to obtain the expressions of ndof, nnz, and nnzpr in terms of n_d and $\texttt{ndof}_i$. To this end, a set of intermediate values that can be expressed in terms of n_d and $\texttt{ndof}_i$ are computed first, Sections A.2 and A.3. Then, these intermediate values are used to obtain the expressions of ndof, nnz, Section A.4. Finally, nnzpr is obtained as $\texttt{nnz}/\texttt{ndof}$.

The first set of intermediate values, Section A.2, corresponds to the structure of the mesh entities:

$a_{i,j}$: The average number of j-dimensional entities that are adjacent to an i-dimensional entity, see Section A.2.1. For a structured mesh with a negligible number of boundary elements, these numbers are just constants.

n_i: The number of i-dimensional entities on the mesh. These numbers can be expressed in terms of the number of elements (n_d), see Section A.2.2.

$\texttt{ndof}_i^{int}$: the number of nodes that are on the interior of an i-dimensional entity. These numbers can be expressed in terms of $\texttt{ndof}_i$, see Section A.2.3. Figure 16 shows the difference between $\texttt{ndof}_i$ and $\texttt{ndof}_i^{int}$ values: the former is the total number of elements for the i-dimensional entity, while the later is the number of interior nodes in the i-dimensional entity.

The second set of intermediate values, Section A.3, corresponds to the structure of the connections between the mesh nodes for different Galerkin methods:

c_i: The number of nodes that are connected to a global node that is on the interior of an i-dimensional entity for a continuous Galerkin method. These numbers can be expressed in terms of $a_{i,j}$ and $\texttt{ndof}_i^{int}$. Hence, they can also be expressed in terms of $\texttt{ndof}_i$, see Section A.3.1.

c: The number of nodes that are connected to a global node for a discontinuous Galerkin method. These numbers are expressed directly in terms of $a_{i,j}$ and $\texttt{ndof}_i$, see Section A.3.2.

A summary of the notation related with the different types of i-dimensional entities is presented in Table IV.

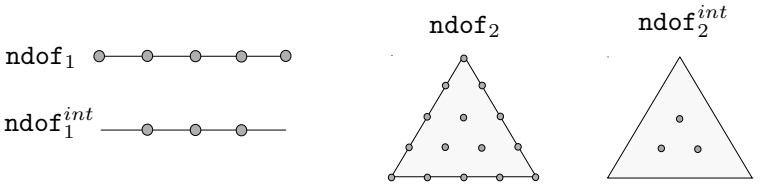

Figure 16. Difference between $\texttt{ndof}_i$ and $\texttt{ndof}_i^{int}$ values on a triangular mesh of interpolation degree $p = 4$ for: (left) edges ($i = 1$) and (right) polygons ($i = 2$).

Table V. Average number of adjacent entities for different mesh types.

Mesh	Triangular			Quadrilateral			Tetrahedral				Hexahedral			
A	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**
0	1	6	6	1	4	4	1	14	36	24	1	6	12	8
1	2	1	2	2	1	2	2	1	36/7	36/7	2	1	4	4
2	3	3	1	4	4	1	3	3	1	2	4	4	1	2
3							4	6	4	1	8	12	6	1

A.2. Quantities related with the mesh entities

A.2.1. Average number of adjacent mesh entities. In order to compute the required cost indicators, the average number of j-dimensional entities that contain or are contained into an i-dimensional entity (*adjacent*) has to be computed. To this end, a structured and uniform mesh with a negligible number of boundary entities is considered both for parallelotopes and simplices. Specifically, the parallelotope mesh is obtained as a periodical 2D (3D) Cartesian grid composed by quadrilateral (hexahedral) elements. The simplicial mesh, is obtained by splitting each quadrilateral (hexahedron) of the 2D (3D) Cartesian grid into 2 triangular (6 tetrahedral) elements, see Figure 1.

To compute the average number of adjacent entities, the symmetry of the initial Cartesian grid can be exploited. That is, any inner quadrilateral (hexahedron) of the 2D (3D) grid is surrounded by 8 (26) elements. Therefore, all the possible topological configurations of inner mesh entities are represented in the central tile of an initial 2D (3D) Cartesian grid composed by 3×3 ($3 \times 3 \times 3$) elements. Figure 17 shows the mesh entities on the central tile of a triangular and a quadrilateral mesh. By symmetry, all the inner vertices are equivalent to the bottom-left central vertex; all the inner edges are equivalent to either the diagonal, the left, or the bottom edge; and all the inner polygons are equivalent to one of the central polygons. Similarly, in the 3D case all the mesh inner entities are equivalent to an entity that is either inside or on the left, the bottom, or the frontal side of the central tile.

Exploiting the symmetry of the initial Cartesian grid, the average number of j-dimensional entities adjacent to an i-dimensional entity is obtained by: creating a 2D (3D) grid composed by 3×3 ($3 \times 3 \times 3$) elements; splitting all the initial elements into either simplices or parallelotopes; counting the number of adjacent j-dimensional entities around all the representative topological configurations of i-entities inside the central tile; and finally, summing the number of adjacent j-dimensional entities and dividing it by the number of representative i-entities contained in the central tiling. The results obtained with this procedure for a structured triangular, quadrilateral, tetrahedral and hexahedral mesh are presented in Table V. In this table, a number on the i-th row and the j-th column determines the average number of j-dimensional entities that are adjacent to an i-entity. For instance, in a structured tetrahedral mesh there are on average $36/7$ polygons (second column) adjacent to an edge (first row); and in a structured hexahedral mesh there are 4 edges (first column) adjacent to a polygon (second row). It is important to point out that these results agree with the average number of adjacent entities presented in [29], except for the tetrahedral case. Nevertheless, the differences on the averages are not significant since in [29]: $a_{0,2} = 35$ (≈ 36), $a_{0,3} = 23$ (≈ 24), $a_{1,2} = 5$ ($\approx 36/7$), and $a_{1,3} = 5$ ($\approx 36/7$).

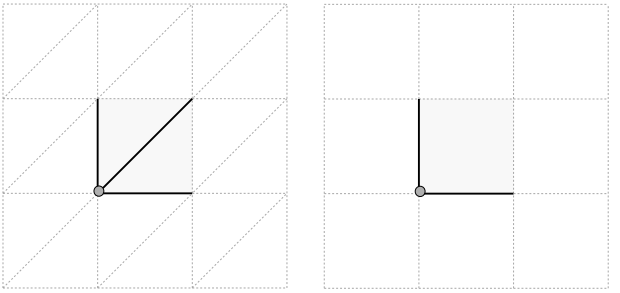

Figure 17. Representative entities on the central tile of a structured triangular and quadrilateral mesh.

Table VI. Number of mesh entities expressed in terms of the number of elements n_d.

Mesh type	$n_0(n_d)$	$n_1(n_d)$	$n_2(n_d)$
triangular	$n_d/2$	$3n_d/2$	-
quadrilateral	n_d	$2n_d$	-
tetrahedral	$n_d/6$	$7n_d/6$	$2n_d$
hexahedral	n_d	$3n_d$	$3n_d$

A.2.2. Number of mesh entities. The number of different types of mesh entities can be expressed in terms of the number of mesh elements. That is, each element (d-dimensional entity) contains $a_{d,i}$ different i-dimensional entities. In addition, each i-dimensional entity is shared on average by $a_{i,d}$ elements. Therefore, the total number of i-dimensional entities is

$$n_i(n_d) = \frac{a_{d,i}}{a_{i,d}} n_d.$$

To illustrate this general expression, the computation of the number of vertices and edges expressed in terms of the number of elements of a triangular mesh is presented:

Number of vertices. Each element (triangle) contains $a_{2,0} = 3$ vertices, and each vertex is shared on average by $a_{0,2} = 6$ elements (triangles). Therefore, the total number of vertices is

$$n_0(n_2) = \frac{a_{2,0}}{a_{0,2}} n_2 = \frac{3}{6} n_2 = \frac{1}{2} n_2.$$

Number of edges. Each element (triangle) contains $a_{2,1} = 3$ edges, and each edge is shared on average by $a_{1,2} = 2$ elements (triangles). Therefore, the total number of edges is

$$n_1(n_2) = \frac{a_{2,1}}{a_{1,2}} n_2 = \frac{3}{2} n_2.$$

According to the values in Table V and the proposed procedure, Table VI presents the number of mesh entities expressed in terms of the number of elements for triangular, quadrilateral, tetrahedral, and hexahedral structured meshes.

A.2.3. Number of nodes per mesh entity. The number of nodes and the number of interior nodes on a i-dimensional mesh entity depends on: the dimension of the mesh entity (i), the type of elements (simplices and parallelotopes), and the interpolation degree (p). Table VII presents, in terms of these parameters, the total number of nodes ($\texttt{ndof}_i$) and the number of internal nodes ($\texttt{ndof}_i^{int}$). Note that $\texttt{ndof}_i$ is obtained as: the dimension of the space of polynomials of degree p in i variables on a simplex; and the dimension of the Cartesian product of i different spaces of polynomials of degree p in one variable on a hexahedron. Then, the number of internal nodes on a i-dimensional entity

Table VII. The total number of nodes and the number of internal nodes on an i-dimensional entity for simplices and parallelotopes.

	simplices		parallelotopes	
Dim	$\texttt{ndof}_i$	$\texttt{ndof}_i^{int}$	$\texttt{ndof}_i$	$\texttt{ndof}_i^{int}$
0	1	1	1	1
1	$p+1$	$p-1$	$p+1$	$p-1$
2	$(p+1)(p+2)/2$	$(p-1)(p-2)/2$	$(p+1)^2$	$(p-1)^2$
3	$(p+1)(p+2)(p+3)/6$	$(p-1)(p-2)(p-3)/6$	$(p+1)^3$	$(p-1)^3$

Table VIII. Number of internal nodes in terms of the total number of nodes on an i-dimensional entity.

	simplices	parallelotopes
$\texttt{ndof}_0^{int}$	1	1
$\texttt{ndof}_1^{int}$	$\texttt{ndof}_1 - 2$	$\texttt{ndof}_1 - 2$
$\texttt{ndof}_2^{int}$	$\texttt{ndof}_2 - 3\texttt{ndof}_1 + 3$	$\texttt{ndof}_2 - 4\texttt{ndof}_1 + 4$
$\texttt{ndof}_3^{int}$	$\texttt{ndof}_3 - 4\texttt{ndof}_2 + 6\texttt{ndof}_1 - 4$	$\texttt{ndof}_3 - 6\texttt{ndof}_2 + 12\texttt{ndof}_1 - 8$

($\texttt{ndof}_i^{int}$) can be computed from $\texttt{ndof}_i$. Specifically, $\texttt{ndof}_i^{int}$ is computed as the total number of nodes on a i-dimensional entity for the interpolation degree $p-2$.

The number of internal nodes can also be computed as the total number of nodes minus the number of nodes on the entity boundary. The number of nodes on the boundary, is the summation of the internal degrees of freedom for all the entities on the boundary. Therefore, $\texttt{ndof}_i^{int}$ can be obtained by means of the recursion

$$\texttt{ndof}_0^{int} := 1,$$
$$\texttt{ndof}_i^{int} := \texttt{ndof}_i - \sum_{j=0}^{i-1} a_{i,j}\texttt{ndof}_j^{int}, \qquad \text{for } i = 1, ..., d.$$

This recursion is used in Table VIII to express $\texttt{ndof}_i^{int}$ in terms of $\texttt{ndof}_i$ for triangular, quadrilateral, tetrahedral, and hexahedral meshes.

A.3. Connections between the mesh nodes for different Galerkin methods

In this section, the structure of the connections between the mesh nodes for different Galerkin methods (CG(NSC),CG,CDG,HDG) and element types (simplices and parallelotopes) is described. Since each degree of freedom is associated with a mesh node, the connections of the mesh nodes allow the estimation of the number of non-zero entries of the global linear system. That is, two degrees of freedom are connected if they both appear with a non-zero coefficient in at least one equation of the linear system. It is important to point out that each mesh node is associated with one mesh point. However, a mesh point can be associated with one (continuous Galerkin) or more (discontinous Galerkin) mesh nodes. Thus, to describe the connections of the nodes it is required to differentiate between the continuous (CG(NSC),CG) and the discontinous (CDG,HDG) Galerkin methods. Finally, several high-order methods (CG,HDG) remove the internal degrees of freedom of the elements from the global linear system. Specifically, the internal degrees of freedom are expressed in terms of the degrees of freedom on the faces by means of an element-by-element procedure (condensation). Therefore, the number of connections of the internal nodes of the polygons (cells) is zero in 2D (3D) for the condensed methods.

A.3.1. Continuous Galerkin methods. The structure of the connections of a continuous Galerkin discretization is straightforward to characterize. That is, a node is connected with all the nodes of the adjacent elements. However, the computation of the number of nodes connected to a node presents two issues. First, the nodes on the boundary of the elements are shared. Therefore, one has to pay

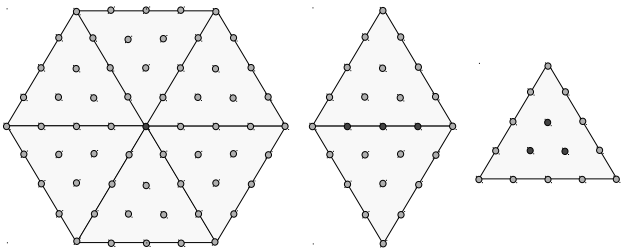

Figure 18. Node connections for the CG(NSC) method on a triangular mesh of interpolation degree $p = 4$: (left) vertex node; (center) interior edge nodes; and (right) interior face nodes.

special attention to count only one time the connection between two nodes. Second, the number of adjacent elements for a node depends on the type of the mesh entities. To overcome these two issues, the structure of the connections is described between the interior nodes of the different types of mesh entities: vertices, edges, polygons, and cells. Specifically, the number of nodes connected to an interior node of a i-dimensional entity (c_i) is obtained as

$$c_i = \sum_{j=0}^{d} c_{i,j}\, \texttt{ndof}_j^{int} \quad \text{for } i = 0, \ldots, d, \tag{16}$$

where $c_{i,j}$ is the total number of j-dimensional entities that are connected to a node on a i-dimensional entity. The following examples illustrate this notation: c_2 is the total number of nodes connected to an interior node of a polygon; $c_{1,2}$ is the number of polygons connected to an internal edge node; and $c_{0,3}$ is the number of cells connected to a vertex node.

Non-condensed continuous Galerkin (CG(NSC)). Since it is not a condensed method, the structure of the connections is described for all the types of mesh nodes. The results and the computation of the number of connected nodes are presented separately for 2D and 3D meshes.

Triangular and quadrilateral (2D). The number of connections between the different types of interior nodes (rows) with the different types of mesh entities (columns) are presented in Table IX. To illustrate the computation of this table, the derivation of the connections of a vertex node of triangular mesh with the surrounding entities is detailed. In addition, Figures 18 (triangles) and 19 (quadrilaterals) are included to show the three possible types of connections for a 2D mesh of interpolation degree $p = 4$.

A vertex node on a triangular mesh is connected with the interior nodes on the vertices, edges, and polygons of the adjacent elements. First, a vertex node is connected to $c_{0,0} = a_{0,0} + a_{0,1}$ vertices, since it shares an element with: itself ($a_{0,0}$); and the opposite vertex of each adjacent edge ($a_{0,1}$). Second, a vertex node is connected to $c_{0,1} = a_{0,1} + a_{0,2}$ edges, since it shares an element with: the adjacent edges ($a_{0,1}$); and the opposite edge of each adjacent triangle ($a_{0,2}$). Finally, it is connected to $c_{0,2} = a_{0,2}$ adjacent triangles. These results correspond to second row of Table IX. A similar derivation is used to obtain the connections of a vertex node on a quadrilateral mesh.

The connections for the interior nodes of an edge and a polygon on a 2D mesh (triangular and quadrilateral) are obtained with the same approach. The corresponding values for triangles appear in the third and fourth rows, and for quadrilaterals in the seventh and eight rows of Table IX, respectively.

Tetrahedral and hexahedral mesh (3D). The number of connections between the different types of interior nodes (rows) with the different types of mesh entities (columns) are presented in Table X for tetrahedra and hexahedra. Since the 3D case is more complex to illustrate than the 2D case, the computation of the connections of a vertex and an edge node are both

Table IX. Connections between nodes and entities for the CG(NSC) method on a triangular and a quadrilateral mesh.

	$\mathbf{C}^{\mathrm{t}}$	**0**	**1**	**2**
Triangles	**0**	$a_{0,0}+a_{0,1}=7$	$a_{0,1}+a_{0,2}=12$	$a_{0,2}=6$
	1	$a_{1,0}+a_{1,2}=4$	$a_{1,1}+2a_{1,2}=5$	$a_{1,2}=2$
	2	$a_{2,0}=3$	$a_{2,1}=3$	$a_{2,2}=1$
	$\mathbf{C}^{\mathrm{q}}$	**0**	**1**	**2**
Quads	**0**	$a_{0,0}+a_{0,1}+a_{0,2}=9$	$a_{0,1}+2a_{0,2}=12$	$a_{0,2}=4$
	1	$a_{1,0}+2a_{1,2}=6$	$a_{1,1}+3a_{1,2}=7$	$a_{1,2}=2$
	2	$a_{2,0}=4$	$a_{2,1}=4$	$a_{2,2}=1$

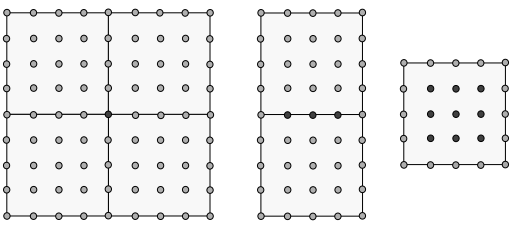

Figure 19. Connections between nodes and entities for the CG(NSC) method on a quadrilateral mesh of interpolation degree $p=4$: (left) vertex node; (center) interior edge nodes; and (right) interior face nodes.

Table X. Connections between nodes and entities for the CG(NSC) method for a tetrahedral and hexahedral mesh.

	$\mathbf{C}^{\mathrm{T}}$	**0**	**1**	**2**	**3**
Tets	**0**	$a_{0,0}+a_{0,1}=15$	$a_{0,1}+a_{0,2}=50$	$a_{0,2}+a_{0,3}=60$	$a_{0,3}=24$
	1	$a_{1,0}+a_{1,2}=50/7$	$a_{1,1}+2a_{1,2}+a_{1,3}=115/7$	$a_{1,2}+2a_{1,3}=108/7$	$a_{1,3}=36/7$
	2	$a_{2,0}+a_{2,3}=5$	$a_{2,1}+3a_{2,3}=9$	$a_{2,2}+3a_{2,3}=7$	$a_{2,3}=2$
	3	$a_{3,0}=4$	$a_{3,1}=6$	$a_{3,2}=4$	$a_{3,3}=1$
	$\mathbf{C}^{\mathrm{H}}$	**0**	**1**	**2**	**3**
Hexes	**0**	$a_{0,0}+a_{0,1}+a_{0,2}+a_{0,3}=27$	$a_{0,1}+2a_{0,2}+3a_{0,3}=54$	$a_{0,2}+3a_{0,3}=36$	$a_{0,3}=8$
	1	$a_{1,0}+2a_{1,2}+2a_{1,3}=18$	$a_{1,1}+3a_{1,2}+5a_{1,3}=33$	$a_{1,2}+4a_{1,3}=20$	$a_{1,3}=4$
	2	$a_{2,0}+4a_{2,3}=12$	$a_{2,1}+8a_{2,3}=20$	$a_{2,2}+5a_{2,3}=11$	$a_{2,3}=2$
	3	$a_{3,0}=8$	$a_{3,1}=12$	$a_{3,2}=6$	$a_{3,3}=1$

detailed. First, a vertex node is connected to $c_{0,0}=a_{0,0}+a_{0,1}$ vertices, since it shares an element with: itself ($a_{0,0}$); and the opposite vertex of each adjacent edge ($a_{0,1}$). A vertex node is connected to $c_{0,1}=a_{0,1}+a_{0,2}$ edges, since it shares an element with: each adjacent edge ($a_{0,1}$); and the opposite edge of each adjacent polygon ($a_{0,2}$). A vertex node is connected to $c_{0,2}=a_{0,2}+a_{0,3}$ polygons, since it shares an element with: each adjacent face ($a_{0,2}$); and the opposite polygon of each adjacent tetrahedra ($a_{0,3}$). Finally, a vertex node is connected to $c_{0,3}=a_{0,3}$ adjacent tetrahedra.

Second, an edge node is connected to $c_{1,0}=a_{1,0}+a_{1,2}$ vertices, since it shares an element with: the end vertices of its edge ($a_{1,0}$); and the opposite vertex of each adjacent polygon ($a_{1,2}$). An edge node is connected to $c_{1,1}=a_{1,1}+2a_{1,2}+a_{1,3}$ edges, since it shares an element with: its edge ($a_{1,1}$); two additional edges for each adjacent polygon ($2a_{1,2}$); and the opposite edge of each adjacent tetrahedra ($a_{1,3}$). An edge node is connected to $c_{1,2}=a_{1,2}+2a_{1,3}$ polygons, since it shares an element with: each adjacent polygon ($a_{1,2}$); and two additional polygons for each adjacent tetrahedron ($2a_{1,3}$). Finally, an edge node is connected to $c_{0,3}=a_{0,3}$ adjacent tetrahedra.

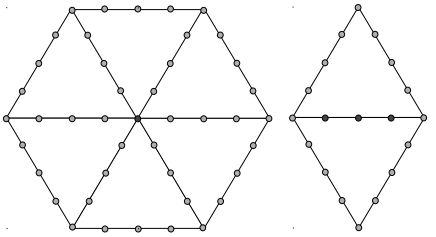

Figure 20. Connections between nodes and entities for the CG method on a triangular mesh of interpolation degree $p = 4$: (left) vertex node; and (right) interior edge nodes.

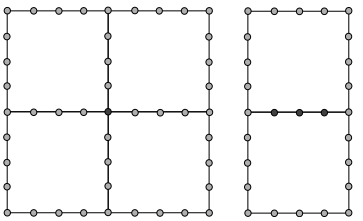

Figure 21. Connections between nodes and entities for the CG method on a quadrilateral mesh of interpolation degree $p = 4$: (left) vertex node; and (right) interior edge nodes.

Table XI. Node connections for the CG method on a triangular and a quadrilateral mesh.

	$\mathbf{C}^{\mathrm{t}}$	**0**	**1**	**2**
Triangles	**0**	$a_{0,0} + a_{0,1} = 7$	$a_{0,1} + a_{0,2} = 12$	0
	1	$a_{1,0} + a_{1,2} = 4$	$a_{1,1} + 2a_{1,2} = 5$	0
	2	0	0	0
	$\mathbf{C}^{\mathrm{q}}$	**0**	**1**	**2**
Quads	**0**	$a_{0,0} + a_{0,1} + a_{0,2} = 9$	$a_{0,1} + 2a_{0,2} = 12$	0
	1	$a_{1,0} + 2a_{1,2} = 6$	$a_{1,1} + 3a_{1,2} = 7$	0
	2	0	0	0

The connections for interior nodes on the polygons and the elements are computed with a similar derivation. The same approach is used to obtain the connections of a hexahedral mesh. The obtained connections for a tetrahedral and hexahedral mesh are presented in Table X.

Continuous Galerkin method (CG). The structure of the connections of the degrees of freedom for the CG method is obtained from the connections of the CG(NSC) method. Specifically, all the connections are obtained from the CG(NSC) table, except the values on the row and column associated with the polygons (cells) in 2D (3D). These values are zero, since the static condensation removes the inner degrees of freedom on the elements from the global system.

Triangular and quadrilateral mesh. The structure of the connections for the interior nodes for a triangular and a quadrilateral mesh is illustrated in Figures 20 and 21, respectively. Furthermore, the corresponding number of connected entities is presented in Table XI.

Tetrahedral and hexahedral mesh. The number of connected entities for a tetrahedral and hexahedral mesh are presented in Table XII.

Given the connections for different mesh types on Tables IX, X for CG(NSC), and XI, XII for CG, the total connections for a node on a i-dimensional entity can be computed by using equation (16). The resulting expressions for the values of c_i in terms of the $\texttt{ndof}_i$ on the entities are given in Table XIII. The values c_i for $i = 0, .., d$ are used for computing the number of non-zero entries ($\texttt{nnz}$) of

Table XII. Connections between nodes and entities for the CG method for a tetrahedral and hexahedral mesh.

	$\mathbf{C}^{\mathrm{T}}$	**0**	**1**	**2**	**3**
Tets	**0**	$a_{0,0}+a_{0,1}=15$	$a_{0,1}+a_{0,2}=50$	$a_{0,2}+a_{0,3}=60$	0
	1	$a_{1,0}+a_{1,2}=50/7$	$a_{1,1}+2a_{1,2}+a_{1,3}=115/7$	$a_{1,2}+2a_{1,3}=108/7$	0
	2	$a_{2,0}+a_{2,3}=5$	$a_{2,1}+3a_{2,3}=9$	$a_{2,2}+3a_{2,3}=7$	0
	3	0	0	0	0
	$\mathbf{C}^{\mathrm{H}}$	**0**	**1**	**2**	**3**
Hexes	**0**	$a_{0,0}+a_{0,1}+a_{0,2}+a_{0,3}=27$	$a_{0,1}+2a_{0,2}+3a_{0,3}=54$	$a_{0,2}+3a_{0,3}=36$	0
	1	$a_{1,0}+2a_{1,2}+2a_{1,3}=18$	$a_{1,1}+3a_{1,2}+5a_{1,3}=33$	$a_{1,2}+4a_{1,3}=20$	0
	2	$a_{2,0}+4a_{2,3}=12$	$a_{2,1}+8a_{2,3}=20$	$a_{2,2}+5a_{2,3}=11$	0
	3	0	0	0	0

Table XIII. Expressions for the number of connected nodes to an interior node of a i-dimensional entity in terms of $\texttt{ndof}_i$ for the continuous Galerkin methods.

	c_i	Expression
Triangles	c_0	$6\texttt{ndof}_2-6\texttt{ndof}_1+1$
	c_1	$2\texttt{ndof}_2-\texttt{ndof}_1$
	c_2	$\texttt{ndof}_2$
Quads	c_0	$4\texttt{ndof}_2-4\texttt{ndof}_1+1$
	c_1	$2\texttt{ndof}_2-\texttt{ndof}_1$
	c_2	$\texttt{ndof}_2$
Tets	c_0	$24\texttt{ndof}_3-36\texttt{ndof}_2+14\texttt{ndof}_1-1$
	c_1	$36\texttt{ndof}_3/7-36\texttt{ndof}_2/7+\texttt{ndof}_1$
	c_2	$2\texttt{ndof}_3-\texttt{ndof}_2$
	c_3	$\texttt{ndof}_3$
Hexes	c_0	$8\texttt{ndof}_3-12\texttt{ndof}_2+6\texttt{ndof}_1-1$
	c_1	$4\texttt{ndof}_3-4\texttt{ndof}_2+\texttt{ndof}_1$
	c_2	$2\texttt{ndof}_3-\texttt{ndof}_2$
	c_3	$\texttt{ndof}_3$

the global matrix for CG(NSC), while only the values for $i=0,..,d-1$ are used for computing the $\texttt{nnz}$ for CG (the interior nodes are statically condensed).

A.3.2. Discontinuous Galerkin methods. For the two discontinuous Galerkin methods (CDG,HDG) considered here, all the internal global nodes have the same connection structure. Specifically, CDG (HDG) is *compact* in the sense that it only connects the nodes of the elements (faces) with the shared faces (all the faces) of the neighboring elements. Note that the global nodes of HDG are on the faces, since the degrees of freedom are condensed through a local procedure.

Compact Discontinuous Galerkin (CDG). For CDG, a node is connected to all the nodes on the same element ($\texttt{ndof}_d$) and the nodes on the adjacent faces of the adjacent elements ($a_{d,d-1}\texttt{ndof}_{d-1}$). Therefore, the total number of connected nodes is:

$$c = a_{d,d-1}\texttt{ndof}_{d-1} + \texttt{ndof}_d, \tag{17}$$

where $a_{d,d-1}$ is the number of adjacent edges (polygons) for the 2D (3D) element, and $\texttt{ndof}_{d-1}$ the total number of nodes on an edge (face) of the 2D (3D) mesh. This expression of the total number of connections is valid for both simplices and parallelotopes.

The structure of the connections of the global nodes for a 2D mesh (triangles and quadrilaterals) is illustrated in Figure 22. Note that a global node is connected to the nodes on the same polygon (element) and on the neighbouring edges (faces). The 3D case is similar, a global node is connected to the nodes on the same cell (element) and on the neighbouring polygons (faces).

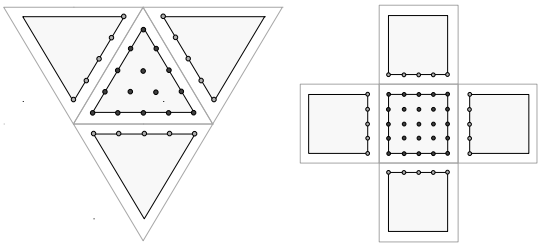

Figure 22. Connections between nodes and entities for the CDG method on a mesh of interpolation degree $p = 4$ composed by: (left) triangles; and (right) quadrilaterals.

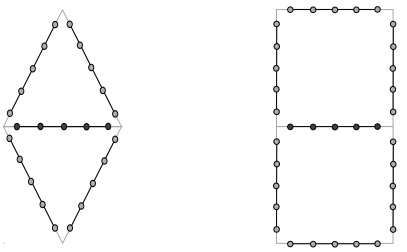

Figure 23. Node connections for the HDG method on a mesh of interpolation degree $p = 4$ composed by: (left) triangles; and (right) quadrilaterals.

Hybridizable Discontinuous Galerkin (HDG). For HDG all the global nodes are on the mesh faces. Moreover, all face nodes are connected to the nodes of all faces of the adjacent elements. That is, a global node of a 2D (3D) mesh that is on an edge (polygon) is connected to all the nodes of the edges (polygons) of the adjacent polygons (cells). Hence, the general expression for the total number of connected nodes is

$$c = (a_{d-1,d}a_{d,d-1} - 1)\,\texttt{ndof}_{d-1}, \tag{18}$$

where for a 2D (3D) mesh, $a_{d-1,d}$ is the number of polygons (cells) sharing an edge (polygon), $a_{d,d-1}$ is the number of edges (polygons) adjacent to a polygon (cell), and $\texttt{ndof}_{d-1}$ is the total number of nodes on an edge (polygon). This expression is valid for both simplices and parallelotopes.

The structure of the connections for a 2D mesh (triangles and quadrilaterals) is illustrated in Figure 23. Note that a global node on an internal edge is connected to the nodes on the edges of the adjacent polygons.

A.4. Number of degrees of freedom and non-zero entries of the global matrix

The expressions for the number of degrees of freedom (`ndof`) and non-zero entries (`nnz`) of the global linear system are presented in Tables XIV and XV, respectively. These expressions are in terms of the previously obtained number of: mesh entities, nodes per entity, and connected nodes. Taking into account Tables VI and VIII, the values of `ndof` and `nnz` can be expressed in terms of the number of mesh elements, and the number of nodes (non-internal) for each type of mesh entity. The resulting expressions of `ndof` and `nnz`, for different methods and mesh types, are presented in Tables XVI and XVII, respectively.

A.4.1. Number of degrees of freedom (`ndof`)

Continuous Galerkin. To avoid to count more than one time a shared degree of freedom on the element boundaries, the total number of degrees of freedom is obtained in terms of the internal degrees of freedom of the mesh entities. Since each i-dimensional mesh entity (n_i) contributes with

Table XIV. Expressions of $\texttt{ndof}$ for the considered Galerkin methods.

	CG(NSC)	CG	CDG	HDG
$\texttt{ndof}$	$\sum_{i=0}^{d} n_i\,\texttt{ndof}_i^{int}$	$\sum_{i=0}^{d-1} n_i\,\texttt{ndof}_i^{int}$	$n_d\,\texttt{ndof}_d$	$n_{d-1}\,\texttt{ndof}_{d-1}$

Table XV. Expressions of $\texttt{nnz}$ for the considered Galerkin methods.

	CG(NSC)	CG	CDG	HDG
$\texttt{nnz}$	$\sum_{i=0}^{d} c_i\,n_i\,\texttt{ndof}_i^{int}$	$\sum_{i=0}^{d-1} c_i\,n_i\,\texttt{ndof}_i^{int}$	$c\,n_d\,\texttt{ndof}_d$	$c\,n_{d-1}\,\texttt{ndof}_{d-1}$

Table XVI. Expressions of $\texttt{ndof}$ in terms of $\texttt{ndof}_i^{int}$ for the considered Galerkin methods.

		$\texttt{ndof}$
Triangles	CG	$n_2\big(3\texttt{ndof}_1-5\big)/2$
	HDG	$3n_2\,\texttt{ndof}_1/2$
	CG(NSC)	$n_2\big(2\texttt{ndof}_2-3\texttt{ndof}_1+1\big)/2$
	CDG	$n_2\,\texttt{ndof}_2$
Quads	CG	$n_2\big(2\texttt{ndof}_1-3\big)$
	HDG	$2n_2\,\texttt{ndof}_1$
	CG(NSC)	$n_2(\texttt{ndof}_2-2\texttt{ndof}_1+1)$
	CDG	$n_2\,\texttt{ndof}_2$
Tets	CG	$n_3\big(12\texttt{ndof}_2-29\texttt{ndof}_1+23\big)/6$
	HDG	$2n_3\,\texttt{ndof}_2$
	CG(NSC)	$n_3\big(6\texttt{ndof}_3-12\texttt{ndof}_2+7\texttt{ndof}_1-1\big)/6$
	CDG	$n_3\,\texttt{ndof}_3$
Hexes	CG	$n_3\big(3\texttt{ndof}_2-9\texttt{ndof}_1+7\big)$
	HDG	$3n_3\,\texttt{ndof}_2$
	CG(NSC)	$n_3(\texttt{ndof}_3-3\texttt{ndof}_2+3\texttt{ndof}_1-1)$
	CDG	$n_3\,\texttt{ndof}_3$

$\texttt{ndof}_i^{int}$ internal degrees of freedom, the total number of global degrees of freedom on all the i-dimensional mesh entities is $n_i\texttt{ndof}_i^{int}$. Therefore, the summation of $n_i\texttt{ndof}_i^{int}$ for all the possible global i-dimensional entities is the total number of degrees of freedom. Note that the limit for the summation is $d-1$ for a condensed method such as CG.

Discontinuous Galerkin. For the CDG method (HDG method), the mesh is composed by n_d (n_{d-1}) elements (faces), and each one contains $\texttt{ndof}_d$ ($\texttt{ndof}_{d-1}$) global nodes.

A.4.2. Number of non-zero entries ($\texttt{nnz}$)

Continuous Galerkin. There is a total of number of n_i i-dimensional entities, each one containing $\texttt{ndof}_i^{int}$ entities. Therefore, there is a total number $n_i \cdot \texttt{ndof}_i^{int}$ of nodes on i-dimensional entities. Moreover, each interior node of an internal i-dimensional entity is connected to c_i nodes. Thus, $\texttt{nnz}$ is the summation of the product of the number of internal nodes on i-dimensional entities ($n_i \cdot \texttt{ndof}_i^{int}$) with the number of connected nodes (c_i). The limit for the summation is $d-1$ for a condensed continuous method such as CG.

Discontinuous Galerkin. On the one hand, all the global nodes of the considered discontinuous Galerkin methods are connected to the same number of nodes. Specifically, for CDG (HDG) each global node on an internal element (face) is connected with c global nodes. On the other hand, for CDG (HDG) there are n_d (n_{d-1}) elements (faces), and each one contains $\texttt{ndof}_d$ ($\texttt{ndof}_{d-1}$) nodes. Therefore, the $\texttt{nnz}$ is $cn_d\texttt{ndof}_d$ and $cn_{d-1}\texttt{ndof}_{d-1}$ for CDG and HDG, respectively.

Table XVII. Expressions of $\texttt{nnz}$ in terms of $\texttt{ndof}_i^{int}$ for the considered Galerkin methods.

		$\texttt{nnz}$
Triangles	CG	$n_2\big(15\texttt{ndof}_1^2 - 36\texttt{ndof}_1 + 19\big)/2$
	HDG	$15n_2\,\texttt{ndof}_1^2/2$
	CG(NSC)	$n_2\big(2\texttt{ndof}_2^2 - 3\texttt{ndof}_1^2 + 1\big)/2$
	CDG	$n_2\,\texttt{ndof}_2(\texttt{ndof}_2 + 3\texttt{ndof}_1)$
Quads	CG	$n_2\big(14\texttt{ndof}_1^2 - 32\texttt{ndof}_1 + 17\big)$
	HDG	$14n_2\,\texttt{ndof}_1^2$
	CG(NSC)	$n_2(\texttt{ndof}_2^2 - 2\texttt{ndof}_1^2 + 1)$
	CDG	$n_2\,\texttt{ndof}_2(\texttt{ndof}_2 + 4\texttt{ndof}_1)$
Tets	CG	$n_3\big(84\texttt{ndof}_2^2 - 288\texttt{ndof}_2\,\texttt{ndof}_1 + +223\texttt{ndof}_1^2 + 192\texttt{ndof}_2 - 288\texttt{ndof}_1 + 95\big)/6$
	HDG	$14n_3\,\texttt{ndof}_2^2$
	CG(NSC)	$n_3\big(6\texttt{ndof}_3^2 - 12\texttt{ndof}_2^2 + 7\texttt{ndof}_1^2 - 1\big)/6$
	CDG	$n_3\,\texttt{ndof}_3(\texttt{ndof}_3 + 4\texttt{ndof}_2)$
Hexes	CG	$3n_3\big(11\texttt{ndof}_2^2 - 48\texttt{ndof}_2\,\texttt{ndof}_1 + +49\texttt{ndof}_1^2 + 32\texttt{ndof}_2 - 64\texttt{ndof}_1 + 21\big)$
	HDG	$33n_3\,\texttt{ndof}_2^2$
	CG(NSC)	$n_3\big(\texttt{ndof}_3^2 - 3\texttt{ndof}_2^2 + 3\texttt{ndof}_1^2 - 1\big)$
	CDG	$n_3\,\texttt{ndof}_3\big(\texttt{ndof}_3 + 6\texttt{ndof}_2\big)$

B. COST OF ONE ITERATION FOR AN ITERATIVE LINEAR SOLVER

Given the number of DOF and the number of non-zero entries in the matrix the cost per iteration of an iterative method estimated because it is dominated by the sparse matrix-vector product, and by applying the pre-conditioning operator. As noted earlier, an average number of non-zero entries per row can be computed as $\texttt{nnzpr} := \texttt{nnz}/\texttt{ndof}$. This average is quite precise for structured schemes as discontinuous Galerkin, both CDG and HDG, but it is only and estimate for CG because of the different connectivity of vertices and edge/face nodes. The matrix-vector product requires $\texttt{nnzpr}$ multiplications and $\texttt{nnzpr} - 1$ additions/subtractions per rows ($\texttt{ndof}$ rows). Thus, the operation count for the sparse matrix-vector product is

$$\texttt{SpMV} = (2\texttt{nnzpr} - 1)\texttt{ndof} = 2\texttt{nnz} - \texttt{ndof}.$$

A typical pre-conditioner is the incomplete LU factorization with zero fill-in (ILU0); this method is chosen because its cost per iteration can be estimated if the cost of a sparse forward and backward substitution is determined. Since it is assumed that the matrix has $\texttt{nnzpr}$ non-zero entries per row, the operation count for both triangular matrices (upper and lower) is counted at the same time, knowing that in total $\texttt{nnzpr}$ terms are on each row. In this manner no extra hypothesis is needed for the banded (or not) structure of the matrix. In this case both the back-substitution and the forward-substitution have:

$$\begin{cases} \texttt{nnzpr}\,\texttt{ndof} = \texttt{nnz} & \text{multiplications,} \\ \texttt{nnzpr}\,\texttt{ndof} = \texttt{nnz} & \text{additions,} \\ 2\texttt{ndof} & \text{divisions.} \end{cases}$$

Consequently, the total number of operations is:

$$\texttt{SpFS} + \texttt{SpBS} = 2\texttt{nnz} + 2\texttt{ndof}$$

This implies that the total cost per iteration, sum of the cost for the sparse matrix-vector product and the cost for the forward and backward substitution, becomes

$$G_p^{iter} = \texttt{SpMV} + \texttt{SpFS} + \texttt{SpBS} = 4\texttt{nnz} + \texttt{ndof}$$

REFERENCES

1. Vos PEJ, Sherwin SJ, Kirby RM. From h to p efficiently: implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations. *J. Comput. Phys.* 2010; **229**(13):5161–5181.
2. Cantwell CD, Sherwin SJ, Kirby RM, Kelly PHJ. From h to p efficiently: strategy selection for operator evaluation on hexahedral and tetrahedral elements. *Comput. Fluids* 2011; **43**(1):23–28.
3. Cantwell CD, Sherwin SJ, Kirby RM, Kelly PHJ. From h to p efficiently: selecting the optimal spectral/hp discretisation in three dimensions. *Math. Model. Nat. Phenom.* 2011; **6**(3):84–96.
4. Löhner R. Error and work estimates for high-order elements. *Int. J. Numer. Methods Fluids* 2011; **67**(12):2184–2188.
5. Kirby RM, Sherwin SJ, Cockburn B. To CG or to HDG: a comparative study. *J. Sci. Comput.* 2012; **51**(1):183–212.
6. Huerta A, Roca X, Aleksandar A, Peraire J. Are High-order and Hybridizable Discontinuous Galerkin methods competitive? *Oberwolfach Rep.* 2012; **9**(1):485–487. Abstracts from the workshop held February 12–18, 2012, Organized by Olivier Allix, Carsten Carstensen, Jörg Schröder and Peter Wriggers, Oberwolfach Reports. Vol. 9, no. 1.
7. Wang Z, Fidkowski K, Abgrall R, Bassi F, Caraeni D, Cary A, Deconinck H, Hartmann R, Hillewaert K, Huynh H, *et al.*. High-order CFD methods: current status and perspective. *Int. J. Numer. Methods Fluids* 2013; **72**(8):811–845.
8. Hartmann R. Higher-order and adaptive discontinuous Galerkin methods with shock-capturing applied to transonic turbulent delta wing flow. *Int. J. Numer. Methods Fluids* 2013; **72**(8):883–894.
9. Giorgiani G, Fernández-Méndez S, Huerta A. Hybridizable Discontinuous Galerkin p-adaptivity for wave propagation problems. *Int. J. Numer. Methods Fluids* 2013; doi:10.1002/fld.3784. To appear.
10. Giorgiani G, Modesto D, Fernández-Méndez S, Huerta A. High-order continuous and discontinuous Galerkin methods for wave problems. *Int. J. Numer. Methods Fluids* 2013; Accepted for publication.
11. Löhner R. Improved error and work estimates for high-order elements. *Int. J. Numer. Methods Fluids* 2013; doi: 10.1002/fld.3783. To appear.
12. Cockburn B, Dong B, Guzmán J. A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems. *Math. Comput.* 2008; **77**:1887–1916.
13. Cockburn B, Gopalakrishnan J, Lazarov R. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM J. Numer. Anal.* 2009; **47**(2):1319–1365.
14. Nguyen NC, Peraire J, Cockburn B. An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations. *J. Comput. Phys.* 2009; **228**(9):3232–3254.
15. Nguyen N, Peraire J, Cockburn B. A hybridizable discontinuous Galerkin method for Stokes flow. *Comput. Meth. Appl. Mech. Eng.* 2010; **199**(9-12):582 – 597.
16. Nguyen NC, Peraire J, Cockburn B. An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations. *J. Comput. Phys.* 2011; **230**(4):1147–1170.
17. Bagheri B, Scott L, Zhang S. Details regarding the implementation of high order finite element methods. *Technical Report*, Department of Mathematics, University of Houston 1992.
18. Meyer CD. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, 2000.
19. Peraire J, Persson PO. The compact discontinuous Galerkin (CDG) method for elliptic problems. *SIAM J. Sci. Comput.* 2007; **30**(4):25.
20. Wang D, Tezaur R, Toivanen J, Farhat C. Overview of the discontinuous enrichment method, the ultra-weak variational formulation, and the partition of unity method for acoustic scattering in the medium frequency regime and performance comparisons. *Int. J. Numer. Methods Eng.* 2012; **89**(4):403–417.
21. Farhat C, Harari I, Franca LP. The discontinuous enrichment method. *Comput. Methods Appl. Mech. Eng.* 2001; **190**(48):6455–6479.
22. Cessenat O, Despres B. Application of an ultra weak variational formulation of elliptic PDEs to the two-dimensional Helmholtz problem. *SIAM J. Numer. Anal.* 1998; **35**(1):255–299.
23. Melenk JM, Babuška I. The partition of unity finite element method: basic theory and applications. *Comput. Methods Appl. Mech. Eng.* 1996; **139**(1–4):289–314.
24. Roca X, Nguyen NC, Peraire J. GPU-accelerated sparse matrix-vector product for a hybridizable discontinuous Galerkin method. *49th AIAA Aerospace Sciences Meeting*, 2010.
25. Roca X, Nguyen NC, Peraire J. A GPU-accelerated iterative solver for a hybridized discontinuous Galerkin method. *23rd Parallel CFD*, 2011.
26. Duff I. Analysis of sparse systems. PhD Thesis, University of Oxford 1972.
27. George A. Nested dissection of a regular finite-element mesh. *SIAM J. Numer. Anal.* 1973; **10**(2):345–363.
28. Angeloski A, Roca X, Peraire J, Huerta A. Computational cost of simplices versus parallelotopes for Galerkin methods. *21st International Meshing Roundtable*, 2012.
29. Remacle J, Shephard M. An algorithm oriented mesh database. *Int. J. Numer. Methods Eng.* 2003; **58**(2):349–374.