Combined CG-HDG Method for Elliptic Problems: Performance Model

Martin Vymazal[1]   David Moxey[2]   Chris Cantwell[1]
Spencer Sherwin[1]   Robert M. Kirby[3]

November 22, 2018

We combine continuous and discontinuous Galerkin methods in the setting of a model diffusion problem. Starting from a hybrid discontinuous formulation, we replace element interiors by more general subsets of the computational domain - groups of elements that support a piecewise-polynomial continuous expansion. This step allows us to identify a new weak formulation of Dirichlet boundary condition in the continuous framework. We examine the expected performance of a Galerkin solver that would use continuous Galerkin method with weak Dirichlet boundary conditions in each mesh partition and connect partitions weakly using trace variable as in HDG method.

1 Motivation for combining CG and HDG

High-order methods on unstructured grids are now increasingly being used to improve the accuracy of flow simulations since they simultaneously provide geometric flexibility and high fidelity. We are particularly interested in efficient algorithms for incompressible Navier-Stokes equations that employ high-order space discretization and a time splitting scheme. The cost of one step in time is largely determined by the amount of work needed to obtain the pressure field, which is defined as a solution to a scalar elliptic problem. Several Galerkin-type methods are available for this task, each of them have specific advantages and drawbacks.

High-order continuous Galerkin (CG) method is the oldest. Compared to its discontinuous counterparts, it involves a smaller number of unknowns (figure 1), especially in a low-order setting. The CG solution can be accelerated by means of static condensation, which produces a globally coupled system involving only those degrees of freedom on the mesh skeleton. The element interior unknowns are subsequently obtained from the mesh skeleton data by solving independent local problems that do not require any parallel communication. The amount of information interchanged while constructing and solving the statically condensed system, however, is determined by the topology of the underlying grid. Unstructured mesh generators often produce meshes with high vertex valency (number of elements incident to given vertex)

---

[1]Department of Aeronautics, Imperial College London, London, UK
[2]College of Engineering, Mathematics and Physical Sciences, University of Exeter, UK
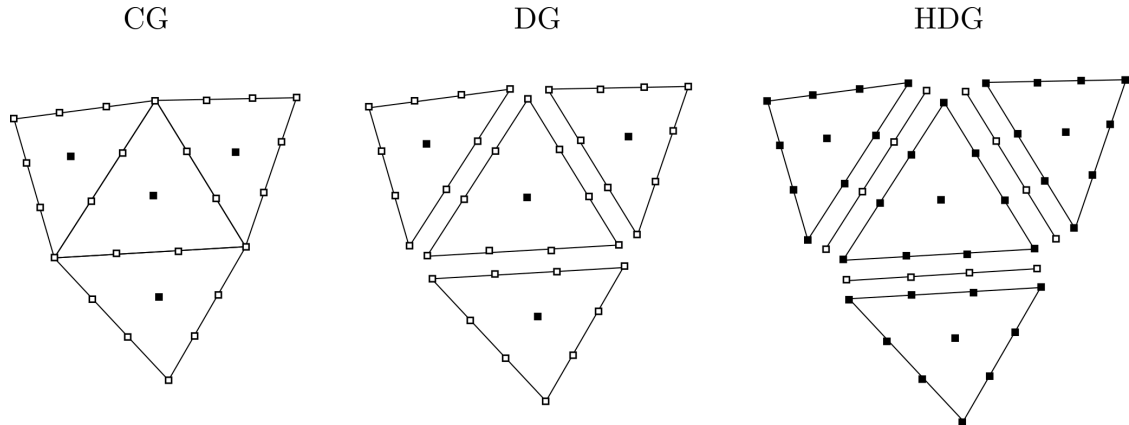[3]School of Computing, Univ. of Utah, Salt Lake City, UT, USA

Figure 1: Distribution of unknowns for continuous and discontinuous Galerkin methods.

and CG therefore has rather complex communication patterns in parallel runs, which has a negative impact on scaling [5].

Discontinuous Galerkin (DG) methods [1], on the other hand, duplicate discrete variables on element boundaries, thus decoupling mesh elements and requiring at most pairwise communication between them. This is at the expense of larger linear system and more time spent in the linear solver. Discontinuous discretization is therefore expected to scale better on parallel computers, but the improved scaling is not necessarily reflected in significantly smaller CPU times when compared to a CG solver.

Hybrid discontinuous Galerkin (HDG) methods [2] address this problem by introducing an additional (hybrid) variable on the mesh skeleton. The hybrid degrees of freedom determine the rank of the global system matrix and HDG therefore produces a statically condensed system that is similar in size to the CG case. In contrast with CG, the static condensation in HDG takes place by construction rather than being an optional iterative technique. Similarly to the classical DG method, HDG scales favourably in comparison with CG, but the work-to-communication ratio is once again improved due to increased amount of intra-node work rather than due to better overall efficiency.

To maximize the potential of each Galerkin variant in a unified setting, we study a finite element discretization that combines the continuous and discontinuous approach by considering a hybrid discontinuous Galerkin method applied to connected groups of elements supporting a globally continuous polynomial basis. This settings leads naturally to a formulation of weak Dirichlet boundary conditions for the CG method.

The next section reviews the Hybridizable Discontinuous Galerkin Method which is then modified in order to obtain the mixed CG-HDG solver.

2 Overview of the formulation of HDG method

We begin with a brief recap of the standard HDG formulation for a finite element mesh, following a similar approach to that taken in [3] and [5].
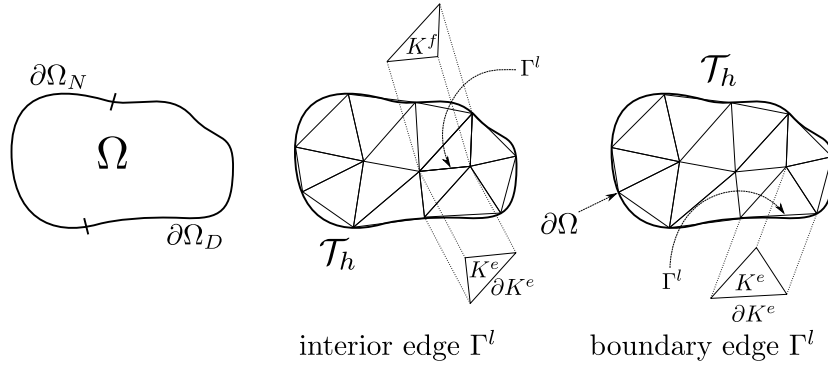
Figure 2: Computational domain and its tesselation demonstrating notation used in the text.

2.1 Continuous problem

We seek the solution of a Poisson equation as a representative elliptic problem

$$
\begin{aligned}
-\nabla^2 u(\boldsymbol{x}) &= f(\boldsymbol{x}) & \boldsymbol{x} \in \Omega, \\
u(\boldsymbol{x}) &= g_D(\boldsymbol{x}) & \boldsymbol{x} \in \partial\Omega_D, \\
\boldsymbol{n} \cdot \nabla u(\boldsymbol{x}) &= g_N(\boldsymbol{x}) & \boldsymbol{x} \in \partial\Omega_N,
\end{aligned} \tag{1}
$$

on a domain Ω with Dirichlet ($\partial\Omega_D$) and Neumann ($\partial\Omega_N$) boundary conditions, where $\partial\Omega_D \bigcup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \bigcap \partial\Omega_N = \emptyset$. To formulate the HDG method, we consider a mixed form of (1) by introducing an auxiliary variable $\boldsymbol{q} = \nabla u$:

$$
\begin{aligned}
-\nabla \cdot \boldsymbol{q} &= f(\boldsymbol{x}) & \boldsymbol{x} \in \Omega, & \tag{2} \\
\boldsymbol{q} &= \nabla u(\boldsymbol{x}) & \boldsymbol{x} \in \Omega, & \tag{3} \\
u(\boldsymbol{x}) &= g_D(\boldsymbol{x}) & \boldsymbol{x} \in \partial\Omega_D, & \tag{4} \\
\boldsymbol{q} \cdot \boldsymbol{n} &= g_N(\boldsymbol{x}) & \boldsymbol{x} \in \partial\Omega_N. & \tag{5}
\end{aligned}
$$

The gradient variable $\boldsymbol{q}$ is approximated together with the primal variable u, which contrasts with the CG method and other discontinuous methods for (1).

2.2 HDG interpolation spaces and discretization

We limit ourselves to two-dimensional problems for sake of simplicity, but the formal description remains unchanged in three dimensions. We assume that in the discrete setting, the computational domain Ω is approximated by its tesselation $\mathcal{T}_h$ consisting of non-overlapping and conformal elements K^e such that for each pair of distinct indices $e_i \neq e_j$, $K^{e_i} \cap K^{e_j} = \emptyset$. The symbol Γ^l denotes an interior edge of the tesselation $\mathcal{T}_h$, i.e. an edge $\Gamma^l = \bar{K}^i \cap \bar{K}^j$ where K^i and K^j are two distinct elements of the tesselation. We say that Γ^l is a boundary edge of the tesselation $\mathcal{T}_h$ if there exists an element K^e such that $\Gamma^l = K^e \cap \partial\Omega$ and the length of Γ^l is not zero, as shown in figure 2. The set of all internal edges is denoted by $\mathcal{E}_h^0$, while $\mathcal{E}_h^\partial$ is a set of all boundary edges. Their union $\mathcal{E}_h$ comprises of all mesh edges, $\mathcal{E}_h = \mathcal{E}_h^0 \cup \mathcal{E}_h^\partial$.

In order to describe some terms in the HDG formulation, it is also useful to introduce mappings that relate elements to their local edges, as shown in figure 3. Let ∂K_j^e be the j-th edge of element K^e, and suppose that this is also the l-th edge Γ^l in the global edge
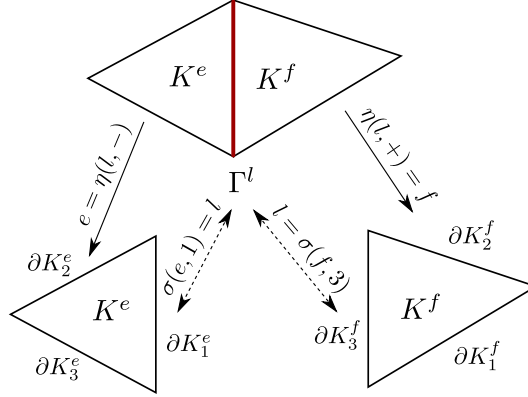
Figure 3: Index mappings relating edge and element ids.

numbering. Then we define the local-to-global edge mapping σ by setting $\sigma(e,j) = l$ so that we can write $\partial K_j^e = \Gamma^{\sigma(e,j)}$. An interior edge Γ^l is the intersection of the boundaries of two elements K^e and K^f, hence we set $\eta(l,+) = e$ and $\eta(l,-) = f$ in order to be able to write $\Gamma^l = \partial K^{\eta(l,+)} \cap \partial K^{\eta(l,-)}$.

2.3 Approximation spaces

The finite element spaces supported by the (two-dimensional) tesselation $\mathcal{T}_h$ are defined as follows:

$$
\begin{aligned}
V_h &:= \{v \in L^2(\Omega) &&: && v|_{K^e} \in \mathcal{P}(K^e) && \forall K^e \in \mathcal{T}_h\}, \\
\mathbf{\Sigma}_h &:= \{\boldsymbol{w} \in [L^2(\Omega)]^2 &&: && \boldsymbol{w}|_{K^e} \in \mathbf{\Sigma}(K^e) && \forall K^e \in \mathcal{T}_h\}, \\
\mathcal{M}_h &:= \{\mu \in L^2(\Omega) &&: && \mu|_{\Gamma^l} \in \mathcal{P}(\Gamma^l) && \forall \Gamma^l \in \Gamma\},
\end{aligned}
$$

where $\mathcal{P}(\Gamma^l) = \mathcal{S}_P(\Gamma^l)$ is the polynomial space over the standard segment

$$
\mathcal{S}_P(\Gamma^l) = \{s^p \ : \ 0 \leq p \leq P, \ [x_1(s), x_2(s)] \in \Gamma^l, \ -1 \leq s \leq 1\},
$$

and $\mathcal{P}(K^e)$ is the space of polynomials of degree P defined on a standard region, which can either be the standard triangle

$$
\mathcal{P}(K^e) = \mathcal{T}_P(K^e) = \{\xi_1^p \xi_w^q \ : \ 0 \leq p+q \leq P, \ [x_1(\xi_1,\xi_2), x_2(\xi_1,\xi_2)] \in K^e, \ -1 \leq \xi_1 + \xi_2 \leq 0\},
$$

or standard quadrilateral

$$
\mathcal{P}(K^e) = \mathcal{Q}_P(K^e) = \{\xi_1^p \xi_2^q \ : \ 0 \leq p,q \leq P, \ [x_1(\xi_1,\xi_2), (x_2(\xi_1,\xi_2)] \in K^e, \ -1 \leq \xi_1, \xi_2 \leq 1\}.
$$

Similarly $\mathbf{\Sigma}(K^e) = [\mathcal{T}_P(K^e)]^2$ or $\mathbf{\Sigma}(K^e) = [\mathcal{Q}_P(K^e)]^2$. There is no requirement on global continuity of the expansion. This is also true for the trace space $\mathcal{M}_h$: a discrete variable $\lambda \in \mathcal{M}_h$ is multi-valued at every mesh vertex shared by multiple interior edges.

4

2.4 Global formulation for HDG problem

Given an element $K \in \mathcal{T}_h$ and two functions $u, v \in L^2(\mathcal{T}_h)$, we define their L^2 scalar product by

$$(u, v)_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} (u, v)_K, \quad \text{where} \quad (u, v)_K = \int_K uv \, d\boldsymbol{x}.$$

Similarly, the L^2 product of functions u and v that are square-integrable on element traces are defined by:

$$\langle u, v \rangle_{\partial \mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} \langle u, v \rangle_{\partial K} \quad \text{where} \quad \langle u, v \rangle_{\partial K} = \int_{\partial K} uv \, d\boldsymbol{s}$$

The DG method seeks an approximation pair $(u^{DG}, \boldsymbol{q}^{DG})$ to u and $\boldsymbol{q}$, respectively, in the space $V_h \times \boldsymbol{\Sigma}_h$. The solution is required to satisfy the weak form of (2) and (3)

$$\left(\boldsymbol{q}^{DG}, \nabla v\right)_{\mathcal{T}_h} = (f, v)_{\mathcal{T}_h} + \left\langle \boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^{DG}, v \right\rangle_{\partial \mathcal{T}_h} \tag{6}$$

$$\left(\boldsymbol{q}^{DG}, \boldsymbol{w}\right)_{\mathcal{T}_h} = -\left(u^{DG}, \nabla \cdot \boldsymbol{w}\right)_{\mathcal{T}_h} + \left\langle \tilde{u}^{DG}, \boldsymbol{w} \cdot \boldsymbol{n}^e \right\rangle_{\partial \mathcal{T}_h} \tag{7}$$

for all $(v, \boldsymbol{w}) \in V_h(\Omega) \times \boldsymbol{\Sigma}_h(\Omega)$, where the numerical traces $\tilde{u}^{DG}$ and $\tilde{\boldsymbol{q}}^{DG}$ have to be suitably defined in terms of the approximate solution $(u^{DG}, \boldsymbol{q}^{DG})$. For details, we refer the reader to [2]. This choice of trace variables allows us to construct the discrete HDG system involving only trace degrees of freedom $\tilde{u}^{DG}$. Once $\tilde{u}$ is known, the element-interior degrees of freedom represented by both the primal variable u and gradient $\boldsymbol{q}$ can be reconstructed from element-boundary values.

We note that the element-interior variable u restricted to element traces is not equal to the hybrid variable $\tilde{u}$, but only approximates it: due to the definition of approximation spaces V_h and $\mathcal{M}_h$, u must be continuous along element boundaries, while $\tilde{u}^{DG}$ is allowed to have jumps in element vertices.

2.5 Local solvers in the HDG method

Assume that the function

$$\lambda := \tilde{u}^{DG} \in \mathcal{M}_h, \tag{8}$$

is given. Then the solution restricted to element K^e is a function $u^e, \boldsymbol{q}^e$ in $P(K^e) \times \boldsymbol{\Sigma}(K^e)$ that satisfies the following equations:

$$(\boldsymbol{q}^e, \nabla v)_{K^e} = (f, v)_{K^e} + \left\langle \boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^e, v \right\rangle_{\partial K^e} \tag{9}$$

$$(\boldsymbol{q}^e, \boldsymbol{w})_{K^e} = -(u^e, \nabla \cdot \boldsymbol{w})_{K^e} + \left\langle \lambda, \boldsymbol{w} \cdot \boldsymbol{n}^e \right\rangle_{\partial K^e}, \tag{10}$$

for all $(v, \boldsymbol{w}) \in P(K^e) \times \boldsymbol{\Sigma}(K^e)$. For a unique solution of the above equations to exist, the numerical trace of the flux must depend only on λ and on $(u^e, \boldsymbol{q}^e)$:

$$\tilde{\boldsymbol{q}}^e(\boldsymbol{x}) = \boldsymbol{q}^e(\boldsymbol{x}) - \tau\left(u^e(\boldsymbol{x}) - \lambda(\boldsymbol{x})\right)\boldsymbol{n}^e \quad \text{on } \partial K^e \tag{11}$$

for some positive function τ. The analysis presented in [2] reveals that as long as $\tau > 0$, its value can be arbitrary without degrading the robustness of the solver. For the limiting value of $\tau \to \infty$, one obtains a statically condensed continuous Galerkin formulation. In this sense, τ plays the role of a method selector as opposed to traditional penalty parameter used in Nitsche's method, for example.

2.6 Global problem for trace variable

We denote by $(U_\lambda, \boldsymbol{Q}_\lambda)$ and by $(U_f, \boldsymbol{Q}_f)$ the solution to the local problem (9), (10) when $\lambda = 0$ and $f = 0$, respectively. Due to the linearity of the original problem (1) and its mixed form, the solution satisfies

$$(u^{HDG}, \boldsymbol{q}^{HDG}) = (U_\lambda, \boldsymbol{Q}_\lambda) + (U_f, \boldsymbol{Q}_f). \tag{12}$$

In order to uniquely determine λ, we require that the boundary conditions be weakly satisfied and the normal component of the numerical trace of the flux $\tilde{\boldsymbol{q}}$ given by (11) is single valued, rendering the numerical trace conservative.

We say that λ is the element of $\mathcal{M}_h$ such that

$$\lambda = \mathbb{P}_h(g_D) \quad \text{on } \partial\Omega_D \tag{13}$$

$$\langle \mu, \tilde{\boldsymbol{q}} \cdot \boldsymbol{n} \rangle_{\partial\mathcal{T}} = \langle \mu, g_N \rangle_{\partial\Omega_N}, \tag{14}$$

for all $\mu \in \mathcal{M}_h^0$ such that $\mu = 0$ on $\partial\Omega_D$. Here $\mathbb{P}_h$ denotes the L^2-projection into the space of restrictions to $\partial\Omega_D$ of functions of $\mathcal{M}_h$.

In the following, we consider $u^e(\boldsymbol{x}), \boldsymbol{q}^e(\boldsymbol{x}) = [q_1, q_2]^T$ and $\lambda^l(\boldsymbol{x})$ to be finite expansions in terms of basis functions $\phi_j^e(\boldsymbol{x})$ for the expansions over elements and the basis $\psi_j^l(\boldsymbol{x})$ over the traces of the form:

$$u^e(\boldsymbol{x}) = \sum_{j=1}^{N_u^e} \phi_j^e(\boldsymbol{x}) \underline{\hat{u}}^e[j] \qquad \boldsymbol{q}_k^e(\boldsymbol{x}) = \sum_{j=1}^{N_q^e} \phi_j^e(\boldsymbol{x}) \underline{\hat{q}}_k^e[j] \qquad \lambda^l(\boldsymbol{x}) = \sum_{j=1}^{N_\lambda^l} \psi_j^l(\boldsymbol{x}) \underline{\hat{\lambda}}^l[j]$$

3 Discrete form of HDG local solver

We now define several local matrices stemming from standard Galerkin formulation, where scalar test functions v^e are represented by $\phi_i^e(\boldsymbol{x})$, with $i = 1, \ldots, N_u^e$.

$$\mathbf{D}_k^e[i,j] = \left(\phi_i^e, \frac{\partial \phi_j^e}{\partial x_k} \right)_{K_e} \qquad\qquad \mathbf{M}^e[i,j] = \left(\phi_i^e, \phi_j^e \right)_{K_e}$$

$$\mathbf{E}_l^e[i,j] = \left\langle \phi_i^e, \phi_j^e \right\rangle_{\partial K_l^e} \qquad\qquad \widetilde{\mathbf{E}}_{kl}^e[i,j] = \left\langle \phi_i^e, \phi_j^e n_k^e \right\rangle_{\partial K_l^e}$$

$$\mathbf{F}_l^e[i,j] = \left\langle \phi_i^e, \psi_j^{\sigma(e,l)} \right\rangle_{\partial K_l^e} \qquad\qquad \widetilde{\mathbf{F}}_{kl}^e[i,j] = \left\langle \phi_i^e, \psi_j^{\sigma(e,l)} n_k^e \right\rangle_{\partial K_l^e}$$

If the trace expansion matches the expansions used along the edge of the elemental expansion and the local coordinates are aligned, that is $\psi_i^{\sigma(e,l)}(s) = \phi_{k(i)}(s)$ then $\mathbf{E}_l^e$ contains the same entries as $\mathbf{F}_l^e$ and similarly $\widetilde{\mathbf{E}}_{kl}^e$ contains the same entries as $\widetilde{\mathbf{F}}_{kl}^e$.

Inserting the finite expansions of the trial functions into equations (9) and (10), and using the definition of the flux (11) yields the matrix form of *local solvers*

$$\sum_{k=1,2} \left\{ (\mathbf{D}_k^e)^T - \sum_{l=1}^{N_b^e} \left[\widetilde{\mathbf{E}}_{kl}^e \right] \right\} \underline{\hat{q}}_k^e + \sum_{l=1}^{N_b^e} \tau^{e,l} \left[\mathbf{E}_l^e \underline{\hat{u}}^e - \mathbf{F}_l^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right] = \underline{f}^e \tag{15}$$

$$\mathbf{M}^e \underline{\hat{q}}_k^e = -(\mathbf{D}_k^e)^T \underline{\hat{u}}^e + \sum_{l=1}^{N_b^e} \widetilde{\mathbf{F}}_{kl}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \qquad k = 1, 2 \tag{16}$$

The *global equation for* λ can be obtained by discretizing the transmission condition (14). We introduce local element-based and edge-based matrices

$$\overline{\mathbf{F}}^{l,e}[i,j] = \left\langle \psi_i^l, \phi_j^e \right\rangle_{\Gamma^l} \qquad \widetilde{\underset{\sim}{\mathbf{F}}}_k^{l,e}[i,j] = \left\langle \psi_i^l, \phi_j^e n_k^e \right\rangle_{\Gamma^l} \qquad \bar{\mathbf{G}}^l[i,j] = \left\langle \psi_i^l, \psi_j^l \right\rangle_{\Gamma^l}$$

and define

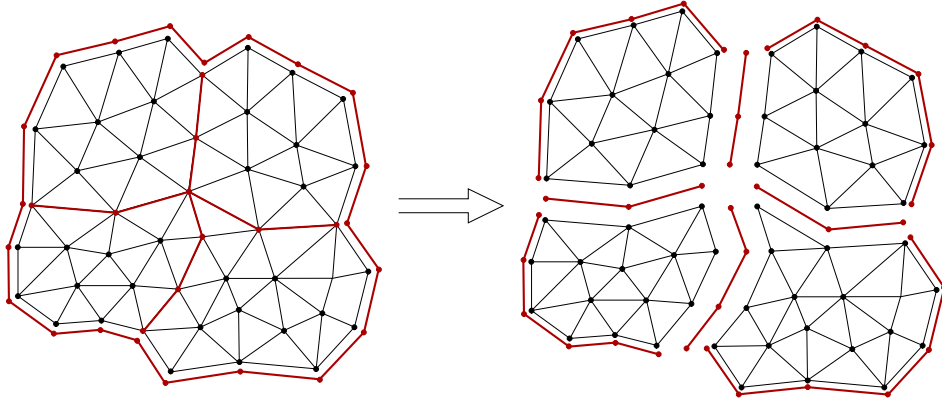$$\underline{g}_N^l[i] = \left\langle g_n, \psi_i^l \right\rangle_{\Gamma^l \bigcap \partial\Omega_N}.$$

The transmission condition in matrix form is then

$$\begin{bmatrix} \underset{\sim}{\mathbf{F}}_1^{l,e} & \underset{\sim}{\mathbf{F}}_2^{l,e} \end{bmatrix} \begin{bmatrix} \hat{\underline{q}}_1^e \\ \hat{\underline{q}}_2^e \end{bmatrix} + \begin{bmatrix} \underset{\sim}{\mathbf{F}}_1^{l,f} & \underset{\sim}{\mathbf{F}}_2^{l,f} \end{bmatrix} \begin{bmatrix} \hat{\underline{q}}_1^f \\ \hat{\underline{q}}_2^f \end{bmatrix} + (\tau^{e,i} + \tau^{f,j})\bar{\mathbf{G}}^l \hat{\underline{\lambda}}^l - \tau^{e,i}\bar{\mathbf{F}}^{l,e}\underline{u}^e - \tau^{f,j}\bar{\mathbf{F}}^{l,f}\underline{u}^f = \underline{g}_N^l,$$

where we are assuming that $l = \sigma(e,i) = \sigma(f,j)$.

4 Combined Continuous-Discontinuous Formulation

To take advantage of the efficiency and lower memory requirements of continuous Galerkin method together with the flexibility and more favorable communication patterns of discontinuous Galerkin methods in domain-decomposition setting, we combine both as follows. Each mesh partition is seen as a 'macro-element', where the governing equation is discretized by continuous Galerkin solver, while the patches are coupled together weakly as in HDG. This means that the scalar flux (hybrid variable) λ is only defined on inter-partition boundaries.

5 Continuous-Discontinuous Solver

Given the hybrid CG-HDG setup, the HDG local solver defined previously for one element would now be applied to a group of elements supporting a piecewise-continuous basis. The motivation of this section is to take the matrix form of the HDG solver and apply it in such piecewise-continuous setting. We will show that the discrete weak form reduces to the 'standard' Laplace operator plus extra terms, which will be only applied on elements adjacent to partition boundaries, providing weak coupling between each partition and the global trace variable.

We start again from the weak mixed problem (9), (10), but integrate the second term in the flux equation (10) by parts once again. This modified flux form allows for a symmetric boundary contribution to the linear system as will be explained shortly. In order to distinguish between the standard HDG local solver within a single element and HDG applied to the whole domain tesselation $\mathcal{T}_h$, the superscript 'e' has been replaced by $\mathcal{T}_h$ where appropriate. The 'macro element' form yields a system

$$\left(\boldsymbol{q}^{\mathcal{T}_h}, \nabla v\right)_{\mathcal{T}_h} = (f, v)_{\mathcal{T}_h} + \left\langle \boldsymbol{n}^{\mathcal{T}_h} \cdot \tilde{\boldsymbol{q}}^{\mathcal{T}_h}, v \right\rangle_{\partial \mathcal{T}_n} \tag{17}$$

$$\left(\boldsymbol{q}^{\mathcal{T}_h}, \boldsymbol{w}\right)_{\mathcal{T}_h} = \left(\nabla u^{\mathcal{T}_h}, \boldsymbol{w}\right)_{\mathcal{T}_h} - \left\langle u^{\mathcal{T}_h}, \boldsymbol{w} \cdot \boldsymbol{n}^{\mathcal{T}_h} \right\rangle_{\partial \mathcal{T}_h} + \left\langle \lambda, \boldsymbol{w} \cdot \boldsymbol{n}^{\mathcal{T}_h} \right\rangle_{\partial \mathcal{T}_h} \tag{18}$$

The numerical approximation $u^{\mathcal{T}_h}$ belongs to the space $V_h^{\mathcal{T}_h}$ and $\boldsymbol{q}^{\mathcal{T}_h}$ lies in $\boldsymbol{\Sigma}_h^{\mathcal{T}_h}$, which are defined as

$$V_h^{\mathcal{T}} := \{v \in \mathcal{C}^0(\Omega) \quad : \quad v|_{K^e} \in P(K^e) \quad \forall K^e \in \mathcal{T}_h\},$$
$$\boldsymbol{\Sigma}_h^{\mathcal{T}} := \{\boldsymbol{w} \in [L^2(\Omega)]^2 : \quad \boldsymbol{w}|_{K^e} \in \boldsymbol{\Sigma}(K^e) \quad \forall K^e \in \mathcal{T}_h\}.$$

Using the definition of the trace flux

$$\tilde{\boldsymbol{q}}^{\mathcal{T}_h}(\boldsymbol{x}) = \boldsymbol{q}^{\mathcal{T}_h}(\boldsymbol{x}) - \tau(u^{\mathcal{T}_h}(\boldsymbol{x}) - \lambda(\boldsymbol{x}))\boldsymbol{n}^{\mathcal{T}_h} \quad \text{on } \partial \mathcal{T}_h,$$

and the fact that the integral over $\mathcal{T}_h$ can be written as a sum of integrals over all $K^e \in \mathcal{T}_h$, equations (17) and (18) become

$$\sum_{K^e \in \mathcal{T}_h} (\nabla v, \boldsymbol{q}^e)_{K^e} - \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle v, \boldsymbol{n}^e \cdot \boldsymbol{q}^e \rangle_{\partial K^e} + \tau \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle v, u^e \rangle_{\partial K^e}$$

$$-\tau \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle v, \lambda \rangle_{\partial K^e} = \sum_{K^e \in \mathcal{T}_h} (v, f)_{K^e} \tag{19}$$

$$\sum_{K^e \in \mathcal{T}_h} (\boldsymbol{w}, \boldsymbol{q}^e)_{K^e} + \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle u^e, \boldsymbol{w} \cdot \boldsymbol{n}^e \rangle_{\partial K^e}$$

$$-\sum_{K^e \in \mathcal{T}_h} (\boldsymbol{w}, \nabla u^e)_{K^e} - \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle \boldsymbol{w} \cdot \boldsymbol{n}^e, \lambda \rangle_{\partial K^e} = 0 \tag{20}$$

A continuous Galerkin solver with Dirichlet data prescribed by the variable λ can be obtained by eliminating the flux variable from the system and reverting back to primal form for the unknown u. The mass matrix which appears in the second equation of the local solver after evaluating the dot product $\left(\boldsymbol{q}^{\mathcal{T}_h}, \boldsymbol{w}\right)_{\mathcal{T}_h}$ is now block-diagonal as a consequence of the discontinuous nature of the discrete flux $\boldsymbol{q}^{\mathcal{T}_h}$, hence the elimination of $\boldsymbol{q}^{\mathcal{T}_h}$ from the system can be performed element-wise. The matrix equivalent of (19), (20) written for a single element $K^e \in \mathcal{T}_h$ adjacent to Dirichlet boundary reads

$$\sum_{k=1,2} \left\{ (\mathbf{D}_k^e)^T - \sum_{l=1}^{N_b^e} \widetilde{\mathbf{E}}_{kl}^e \right\} \underline{\hat{q}}_k^e + \sum_{l=1}^{N_b^e} \tau^{e,l} \left[\mathbf{E}_l^e \underline{\hat{u}}^e - \mathbf{F}_l^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right] = \underline{f}^e \tag{21}$$

$$\mathbf{M}^e \underline{\hat{q}}_k^e = \left\{ (\mathbf{D}_k^e) - \sum_{l=1}^{N_b^e} \widetilde{\mathbf{E}}_{kl}^e \right\} \underline{\hat{u}}^e + \sum_{l=1}^{N_b^e} \widetilde{\mathbf{F}}_{kl}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \qquad k = 1, 2 \tag{22}$$

The discrete flux $\hat{\underline{q}}_k^e$ expressed from (22) and substituted in equation (21) yields element-wise contribution to the left- and right-hand side of the linear system which can be expressed as

$$\sum_{k=1,2} \left\{ \underbrace{(\mathbf{D}_k^e)^T (\mathbf{M}^e)^{-1} \mathbf{D}_k^e}_{\boxed{1}} - \underbrace{\left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) (\mathbf{M}^e)^{-1} \mathbf{D}_k^e}_{\boxed{2a}} \right\} \hat{\underline{u}}^e$$

$$- \sum_{k=1,2} \left\{ \underbrace{(\mathbf{D}_k^e)^T (\mathbf{M}^e)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right)}_{\boxed{2b}} + \underbrace{\left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) (\mathbf{M}^e)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right)}_{\boxed{3}} \right\} \hat{\underline{u}}^e + \underbrace{\sum_{l=1}^{N_b^e} \tau^{(e,l)} \mathbf{E}_l^e \hat{\underline{u}}^e}_{\boxed{4}}$$

$$= \underline{f}^e + \sum_{k=1,2} \left\{ \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e - (\mathbf{D}_k^e)^T \right) (\mathbf{M}^e)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{kl}^e \hat{\underline{\lambda}}^{\sigma(e,l)} \right) \right\} + \sum_{l=1}^{N_b^e} \tau^{(e,l)} \mathbf{F}_l^e \hat{\underline{\lambda}}^{\sigma(e,l)}$$

Term $\boxed{1}$ on the left-hand side is a discrete Laplacian that arises from the standard continuous Galerkin discretization, which would typically be accompanied by the forcing term $\underline{f}^e$ on the right hand side. The additional numbered terms represent a *weak imposition of Dirichlet boundary data represented by* $\hat{\underline{\lambda}}$. (More details on weak Dirichlet boundary conditions can be found in our paper [4]). This new expression therefore denotes a modification of the existing matrix system and right hand side, which makes implementation relatively straightforward. The matrix expressions $\boxed{2a}$, $\boxed{2b}$, $\boxed{3}$ and $\boxed{4}$ appear in the formulation only for elements K^e containing at least one edge on Dirichlet boundary of Ω. In addition, expressions $\boxed{3}$ and $\boxed{4}$ are symmetric as a consequence of symmetry of $\tilde{\mathbf{E}}_{kl}^e, \mathbf{E}_l^e$ and $(\mathbf{M}^e)^{-1}$. The products $\boxed{2a}$ and $\boxed{2b}$ are transposes of each other, hence their sum is again symmetric. The modifications to the symmetric discrete Laplacian therefore preserve symmetry of the discrete weak form, meaning that efficient iterative solvers such as the conjugate gradient method can be used to obtain solutions.

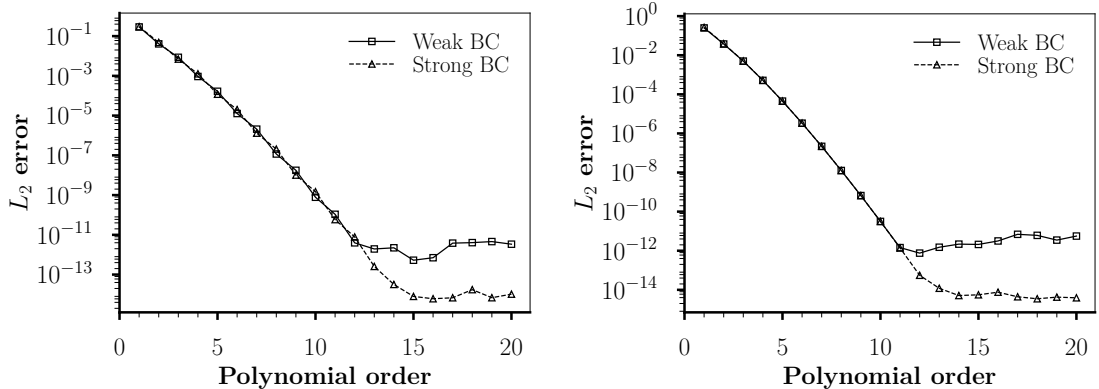## 5.1 Convergence rates comparison: weak vs. strong boundary conditions

The convergence rates were verified on a scalar Helmholtz problem with nonzero Dirichlet boundary conditions

$$\nabla^2 u - \lambda u = f \tag{23}$$

in a square domain $(-1, 1)^2$ with $\lambda = 1$ and $f(x, y)$ chosen so that the exact solution is of the form

$$u(x, y) = \sin(10\pi x)\cos(10\pi y) + x + y \tag{24}$$

Figure 4 compares the L^2 error for polynomial orders varying between 1 and 20 when the Dirichlet boundary conditions are imposed strongly and weakly. The behaviour of both strong and weak methods produces nearly identical errors up to $p = 12$ on the structured grid and $p = 11$ on triangles. With further increase of polynomial degree of the basis, however, the weak errors fail to further decrease. The observed differences are not surprising, because the HDG-based algorithm only penalizes the solution in order to satisfy boundary conditions, while the strong implementation completely eliminates known degrees of freedom and moves them to the right-hand side of the linear system, thus fulfilling the boundary conditions

(a) Convergence to exact solution, strong vs. weak boundary conditions on unstructured triangular mesh.

(b) Convergence to exact solution, strong vs. weak boundary conditions on quadrilateral mesh.

Figure 4: Convergence to the exact solution in L_2 norm.

exactly by construction. Furthermore, the stiffness matrix with weak constraints is larger, hence less favourably conditioned and round-off errors in the linear solver become important as the error values approach the limits of finite-precision arithmetic on given machine.

6 Expected Performance

6.1 Cost in terms of FLOPs

We assume that the discrete Poisson problem is solved in two stages, both of which will significantly contribute to the overall CPU time spent in the solver. The stages are:

1. **Assembly and solution of statically condensed system.** This step involves processing unknowns on entity boundaries, where 'entity' would be each single element in the context of continuous and hybrid discontinuous Galerkin methods and one mesh patch (a group of elements spanned by a continuous polynomial basis) in the combined continuous-discontinuous Galerkin method.

   The main difference between CG and HDG is that in the continuous case, trace variables are identical to variables located on element boundaries and are shared by neighbouring elements. The HDG method, on the other hand, introduces an additional hybrid variable, thus requiring more memory storage. This variable is not globally continuous, hence degrees of freedom on face boundaries are duplicated. As a consequence, the HDG interior solve on each element has to process a slightly larger local system.

2. **Interior solve.** Given solution on entity boundary, the solution in entity interior is reconstructed during this stage. Interior solve involves the inverse of a potentially large matrix.

Setup costs (for example precomputing and storing the matrix inverses needed in interior solve above) are not taken into account.

10

Domain Description

We assume a structured grid divided into $P \times P$ patches, each patch consisting of $N_e^{1D} \times N_e^{1D}$ elements (figure 5). Each element has a polynomial basis of degree p, i.e. $(p+1) \times (p+1)$ degrees of freedom. These may or may not be shared with neighbouring elements, depending
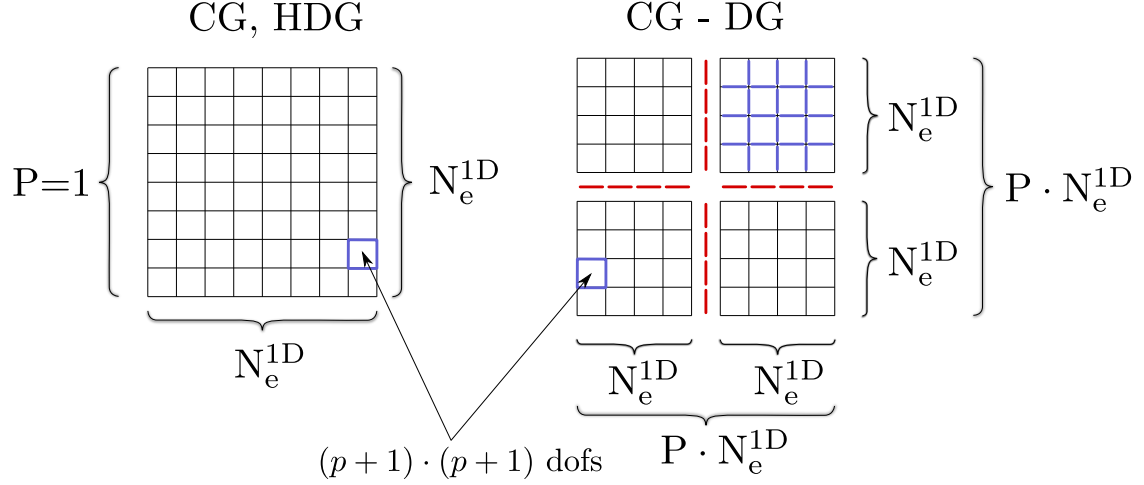

Figure 5: Idealized mesh divided into $P \times P$ patches, each patch containing $N_e^{1D} \times N_e^{1D}$ elements of order p.

on the setup (CG vs. DG vs. HDG) and global continuity of the polynomial bases. The number of inter-patch edges (red edges in figure 5) is

$$N_{interpatch}^{edges} = 2P \cdot (P-1) \cdot N_e^{1D},$$

and each patch contains N_{patch}^{edges} interior edges, with

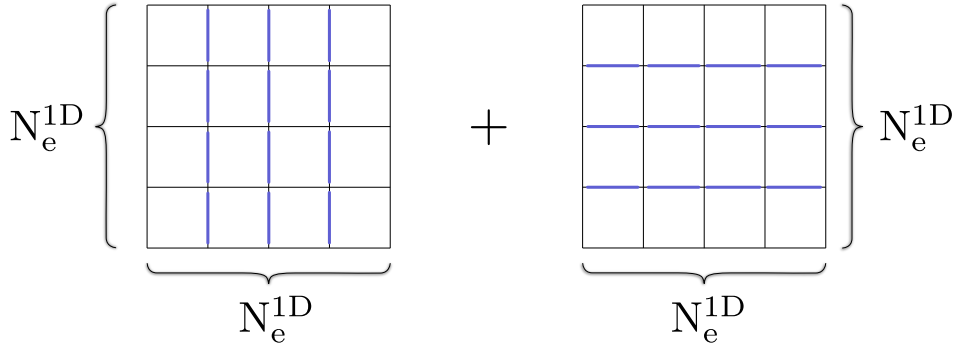$$N_{patch}^{edges} = 2N_e^{1D} \cdot (N_e^{1D} - 1),$$

see figure 6.

Figure 6: Interior edges (blue) within a patch.

Stage I: Solution of Statically Condensed System

Continuous Galerkin

Since the total number of elements along each side of the mesh is $P \cdot N_e^{1\mathrm{D}}$ in 2D, the total number of unknowns before static condensation (assuming Dirichlet boundary condition everywhere) is

$$N_{CG}^{dof} = \left(P \cdot N_e^{1\mathrm{D}} \cdot p - 1\right)^2.$$

This is also the rank of global system matrix. In case of one-level static condensation, the global system has the form

$$\begin{bmatrix} \mathbf{M}_b & \mathbf{M}_c \\ \mathbf{M}_c^T & \mathbf{M}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_b \\ \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_b \\ \boldsymbol{f}_i \end{bmatrix}$$

and the rank of $\mathbf{M}_b$ is *approximately* (counting the boundary modes on the skeleton of the mesh) equal to

$$N_{CG}^{\lambda} = (N_{interpatch}^{edges} + P \cdot N_{patch}^{edges}) \cdot p = (2P \cdot (P-1) \cdot N_e^{1\mathrm{D}} + P \cdot 2N_e^{1\mathrm{D}} \cdot (N_e^{1\mathrm{D}} - 1)) \cdot p$$
$$= 2PN_e^{1\mathrm{D}}(P + N_e^{1\mathrm{D}} - 2)p.$$

The remaining values of u in element-interior degrees of freedom can be obtained by inverting $(P \cdot N_e^{1\mathrm{D}})^2$ local matrices of rank $p - 1$. This means that the total cost of solving the CG problem is

$$C_{CG} = \mathcal{O}\left(cgsolve(PN_e^{1\mathrm{D}}(P + N_e^{1\mathrm{D}})p)\right) + (PN_e^{1\mathrm{D}})^2 \cdot \mathcal{O}\left((p-1)^3\right),$$

where $cgsolve(n)$ is the cost function of solving a sparse system of rank n with conjugate gradients. The cost of the second term is small if the blocks of $\mathbf{M}_i$ are inverted and stored during setup phase. The second term in the estimate assumes that the inverse of each diagonal block of $\mathbf{M}_i$ costs as much as Gauss elimination/LU decomposition of a matrix of rank $p - 1$, which has cubic time complexity.

HDG

The discrete transmission condition (14) generates a sparse system of rank

$$N_{HDG}^{\lambda} = (N_{interpatch}^{edges} + P \cdot N_{patch}^{edges}) \cdot (p+1) = (2P(P-1)N_e^{1\mathrm{D}} + P \cdot 2N_e^{1\mathrm{D}}(N_e^{1\mathrm{D}} - 1)) \cdot (p+1)$$
$$= 2PN_e^{1\mathrm{D}}(P + N_e^{1\mathrm{D}} - 2)(p+1).$$

In addition, we need to invert $(PN_e^{1\mathrm{D}})^2$ local systems $\in \mathbb{R}^{(p+1)\times(p+1)}$ as in the CG case. The backsolve is more expensive however, because we have d mixed variables $q_1, \ldots q_d$ in d dimensions. The element local inversion can be again precomputed and stored during setup.

The overall cost of solving for all unknowns scales as

$$C_{HDG} = \mathcal{O}\left(cgsolve(PN_e^{1\mathrm{D}}(P + N_e^{1\mathrm{D}})(p+1))\right) + (PN_e^{1\mathrm{D}})^2 \cdot \mathcal{O}\left((p+1)^3\right).$$

Combined CG-DG Solver

The number of hybrid degrees of freedom on interfaces between patches is

$$N^\lambda_{CG-DG} = N^{edges}_{interpatch}(p+1) = 2P(P-1)N^{1D}_e(p+1).$$

Each patch contains *approximately* $(N^{1D}_e p)^2$ interior degrees of freedom, hence the total cost is

$$C_{CG-DG} = \mathcal{O}\big(cgsolve(P^2 N^{1D}_e(p+1))\big) + P^2 \cdot \mathcal{O}\big((N^{1D}_e p)^3\big).$$

In the limiting case where each patch coincides with one single element (i.e. $P := N^{1D}_e$ and $N^{1D}_e = 1$), the three estimates C_{CG}, C_{HDG} and C_{CG-DG} predict the same asymptotic cost.

Cost of Solving the Statically Condensed System

Standard HDG Algorithm

The cost of linear solve in the PCG (preconditioned conjugate gradient) solver will mainly depend on the cost of evaluating matrix-vector multiplications. For a matrix of rank n, this cost is $\mathcal{O}(n^2)$. Nektar++ solves the statically condensed system in matrix-free manner by performing the above matrix-vector multiplications *element-wise* and then summing them together. Suppose the (structured) mesh consists of quadrilaterals in 2D and hexahedra in 3D. Furthermore, we will assume that the triangular mesh is obtained by splitting each quadrilateral into 2 triangles and tetrahedral mesh is created by dividing each hexahedron into 6 tetrahedra.

The number of trace degrees of freedom of one element is

- $3 \cdot (p+1)$ for **triangles**

- $4 \cdot (p+1)$ for **quadrilaterals**

- $4 \cdot \frac{(p+1)(p+2)}{2}$ for **tetrahedra**

- $6 \cdot (p+1)^2$ for **hexahedra**

Under this assumption, *one matrix-vector multiplication* for the whole system (but performed on element-wise basis) will take

- $\mathcal{O}\big(2(N^{1D}_e)^2\big[3 \cdot (p+1)\big]^2\big) = \mathcal{O}\big(18(N^{1D}_e)^2(p+1)^2\big)$ operations on **triangles** in 2D

- $\mathcal{O}\big((N^{1D}_e)^2\big[4 \cdot (p+1)\big]^2\big) = \mathcal{O}\big(16(N^{1D}_e)^2(p+1)^2\big)$ operations on **quadrilaterals** in 2D

- $\mathcal{O}\big(6(N^{1D}_e)^3\big[2 \cdot (p+1)(p+2)\big]^2\big) = \mathcal{O}\big(24(N^{1D}_e)^3(p+1)^2(p+2)^2\big)$ operations on **tetrahedra** in 3D

- $\mathcal{O}\big((N^{1D}_e)^3\big[6 \cdot (p+1)^2\big]^2\big) = \mathcal{O}\big(36(N^{1D}_e)^3(p+1)^4\big)$ operations on **hexahedra** in 3D

HDG Algorithm Applied to Groups of Continuously Connected Elements

Now suppose that the trace system is built between patches and each patch has $N^{1D}_e \times N^{1D}_e$ quadrilaterals in 2D and $N^{1D}_e \times N^{1D}_e \times N^{1D}_e$ hexahedra in 3D. The number of unknowns on the trace of one patch now becomes

- $4 \cdot N_e^{1D} \cdot p$ in 2D (**triangles** and **quadrilaterals**) and

- $6 \cdot (N_e^{1D})^2 \cdot p^2$ in 3D (**tetrahedra** and **hexahedra**),

which will require

- $\mathcal{O}\big(16(N_e^{1D})^2 \cdot p^2\big)$ operations per matrix-vector multiplication in 2D and

- $\mathcal{O}\big(36(N_e^{1D})^4 \cdot p^4\big)$ operations in 3D

This means that the PCG algorithm in CG-HDG case scales one order worse when measured in terms of number of elements along patch face $\big(\mathcal{O}((N_e^{1D})^4)\big)$ than the standard HDG algorithm $\big(\mathcal{O}((N_e^{1D})^3)\big)$.

Remark 1. *Note that the number on the surface of the patch is the same for triangles and quadrilaterals and for tetrahedra and hexahedra, respectively. For a continuous expansion, the number of DOFs on one quadrilateral face of a hexahedron is $(p+1)^2$, and $2 \cdot \frac{(p+1)(p+2)}{2} - (p+1) = (p+1)^2$ for two triangles covering the same quadrilateral face.*
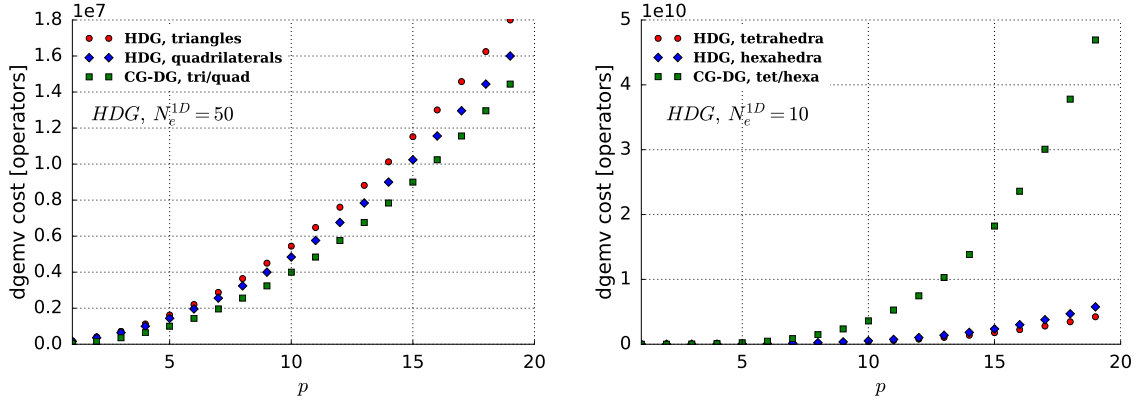

Figure 7: Asymptotic cost of matrix-vector multiplication measured by operation counts for HDG and combined CG-HDG methods.

Stage II: Interior Solve

The reconstruction of interior degrees of freedom involves the solution of a linear system with the matrix

$$
\mathbf{A}^e = \begin{pmatrix} \sum\limits_{l=1}^{N_b^e} \tau^{(e,l)}\mathbf{E}_l^e & -\mathbf{D}_1^e & -\mathbf{D}_2^e \\ \big(\mathbf{D}_1^e\big)^T & \mathbf{M}^e & \mathbf{0} \\ \big(\mathbf{D}_2^e\big)^T & \mathbf{0} & \mathbf{M}^e \end{pmatrix}
$$

The superscript ^{e} no longer refers to a single element as was the case of HDG. For CG-HDG method, all the blocks in $\mathbf{A}$ are a result of a continuous Galerkin discretization in the whole partition/patch. The sparse matrix $\mathbf{A}^e$ is potentially large and its explicit inverse will be dense, i.e. require significant storage.

The expensive interior solve together with increased operation count when inverting the statically condensed system in 3D indicates that the benefit of reduced communication pattern in continuous-discontinuous discretization might be outweighed by extra CPU cost and there is relatively little performance (if any) to be gained by combining the continuous and discontinuous Galerkin discretization into one hybrid solver.

6.2 Cost in terms of memory requirements

Stage I: Solution of Statically Condensed System

We again assume that the global system is solved by performing multiple PCG iterations, where global matrix-vector multiply is executed in matrix-free fashion. For each elemental multiplication, the data containing the input matrix and vector and resulting vector must be loaded into processor cache. We discuss amount of data transferred for each method *during one PCG iteration* here.

Continuous Galerkin

On triangles, one element contains $3 \cdot p$ trace degrees of freedom (this is irrespective of global continuity of the solution, because we perform the multiplication element-by element and hence whether the DOFs are shared with neighbouring elements or not is irrelevant). The elemental statically condensed matrix has rank $(3 \cdot p)^2$ and the amount of data to move in and out of cache is therefore

$$(3 \cdot p)^2 + 2 \cdot (3 \cdot p) = 9p^2 + 6p$$

(The term $2 \cdot (3 \cdot p)$ takes into account two vectors needed for elemental matrix-vector multiplication.) This is repeated for each element, and therefore the total number of floating-point values transferred is

$$(2 \cdot N_e^{1D})(9p^2 + 6p)$$

Repeating similar calculation for other element types, we arrive to the following estimates for continuous Galerkin method and different element types:

- **Triangles**: $N_{CG,tri} = (2(N_e^{1D})^2(9p^2 + 6p)$ floating point values

- **Quadrilaterals**: $N_{CG,quad} = (N_e^{1D})^2(16p^2 + 8p)$ floating point values

- **Tetrahedra:** $N_{CG,tet} = (6 \cdot N_e^{1D})^3\big((2p(p+1))^2 + 4p(p+1)\big)$ floating point values

- **Quadrilaterals:** $N_{CG,hex} = (N_e^{1D})^3\big((6p^2)^2) + 12p^2)\big)$ floating point values

6.2.1 Discontinuous Galerkin

Data for HDG are very similar with the exception that the trace values are discontinuous, which means that the elemental matrices are slightly bigger:

- **Triangles:** $N_{HDG,tri} = (2(N_e^{1D})^2(9(p+1)^2 + 6(p+1))$ floating point values

- **Quadrilaterals:** $N_{HDG,quad} = (N_e^{1D})^2(16(p+1)^2 + 8(p+1))$ floating point values

- **Tetrahedra:** $N_{HDG,tet} = (6 \cdot N_e^{1D})^3 \big((2(p+1)(p+2))^2 + 4(p+1)(p+2)\big)$ floating point values

- **Quadrilaterals:** $N_{HDG,hex} = (N_e^{1D})^3 \big((6(p+1)^2)^2 + 12(p+1)^2\big)$ floating point values

6.2.2 CG-HDG

The hybrid CG-HDG method has a system matrix of rank $(4 \cdot N_e^{1D} \cdot p)$, which means that the number of floating point values involved in one matrix-vector multiply will be

$$N_{CG-HDG,2D} = (4 \cdot N_e^{1D} \cdot p)^2 + (8 \cdot N_e^{1D} \cdot p) = 16(N_e^{1D})^2 p^2 + 8 N_e^{1D} p \text{ in 2D}$$

and similarly in 3D, where the number of DOFs on patch surface is $6 \cdot (N_e^{1D})^2 \cdot p^2$:

$$N_{CG-HDG,3D} = (6 \cdot (N_e^{1D})^2 \cdot p^2)^2 + (12 \cdot (N_e^{1D})^2 \cdot p^2) \text{ in 3D.}$$

Stage II: Interior Solve

Continuous Galerkin

The number of interior degrees of freedom in one high-order triangle is $(p+1)(p+2)/2 - 3p = (p-2)(p-1)/2$ and we suppose that this is the rank of elemental Schur complement which has to be inverted and stored. In the case of continuous Galerkin system, the element-interior matrix, left- and right- hand side vectors hold

$$N_{CG,tri} = ((p-2)(p-1)/2)^2 + 2 \cdot \big((p-2)(p-1)/2\big)$$

This cost has to be multiplied by number of elements present in the mesh. Cost for different element shapes is summarized below

- **Triangles**: $N_{CG,tri} = (2(N_e^{1D})^2 \big[((p-2)(p-1)/2)^2 + (p-2)(p-1)\big]$ floating point values

- **Quadrilaterals**: $N_{CG,quad} = (N_e^{1D})^2 \big[(p-1)^2 + 2(p-1)\big]$ floating point values

Discontinuous Galerkin

In HDG, each elements has its 'own' DOFS not shared with the hybrid variable, hence elemental matrices are again slightly bigger:

- **Triangles**: $N_{HDG,tri} = (2(N_e^{1D})^2 \big[((p+1)(p+2)/2)^2 + (p+1)(p+2)\big]$ floating point values

- **Quadrilaterals**: $N_{HDG,quad} = (N_e^{1D})^2 \big[(p+1)^2 + 2(p+1)\big]$ floating point values

CG-HDG

The 'interior matrix' is sparse, but involves all DOFS of the patch, whose count is approximately $(N_e^{1D})^2 p^2$. The matrix and corresponding storage would then be

$$\textcolor{red}{N_{CG-HDG,quad} = \big((N_e^{1D})^2 p^2\big)^2 + 2(N_e^{1D})^2 p^2}$$

Which is again orders of magnitude worse estimate than for CG and HDG.

7 Conclusion

This paper proposes a new method for combining the CG and HDG solvers and derives an algorithm for the imposition of Dirichlet boundary conditions for elliptic PDEs of Helmholtz type which enforces the constraints weakly, i.e. by amending the underlying weak form with penalty terms instead of lifting known boundary values from the linear system.

The presented technique is conceptually based on hybrid Discontinuous Galerkin method, but replaces the polynomial space typically used in element interiors (a finite element basis defined in single element) by a piecewise continuous multi-element Galerkin expansion. We demonstrate that the method is conceptually feasible and it combines some attractive features of CG and HDG, but it fails to deliver the expected performance. Even if we stored the inverted local solvers to effectively recover degrees of freedom located on each mesh partition, the cost of assembly and solution of the discrete transmission condition in three dimensions remains prohibitively expensive.

Acknowledgments

References

[1] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and L Donatella Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[2] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[3] Robert M. Kirby, Spencer J. Sherwin, and Bernardo Cockburn. To CG or to HDG: A comparative study. *Journal of Scientific Computing*, 51(1):183–212, 2011.

[4] Martin Vymazal, David Moxey, Spencer Sherwin, Chris Cantwell, and Robert M. Kirby. On Weak dirichlet boundary conditions for elliptic problems in the continuous Galerkin method. *Journal of Scientific Computing*, Submitted.

[5] Sergey Yakovlev, David Moxey, Robert M. Kirby, and Spencer J. Sherwin. To CG or to HDG: A comparative study in 3D. *Journal of Scientific Computing*, 67(1):192–220, 2016.