Advanced Topics in Numerical Methods for Partial Differential Equations

16.930

Massachusetts Institute of Technology

Project 5 - HDG for the Convection-Diffusion Equation

Renato Trono Figueras

May 2023

HDG Formulation for the Convection-Diffusion Problem

The objective of this project is to extend the code by implementing the Hybridizable Discontinuous Galerkin (HDG) method to solve the 2D convection-diffusion equation. First, we look at the formulation of the problem, and then we briefly summarize the implementation aspects.

Strong Form of the Equation

We are interested in solving the following system of equation:

$$\mathbf{q} - \nabla u = \mathbf{0}, \quad \text{in } \Omega \tag{1}$$

$$-\nabla \cdot (\kappa \mathbf{q}) + \nabla \cdot (\mathbf{c}u) = f, \quad \text{in } \Omega \tag{2}$$

$$u = g, \quad \text{on } \partial\Omega \tag{3}$$

where $\Omega \subset \mathbb{R}^d$, with d the spatial dimension of the problem ($d = 2$ in this case).

Approximation Spaces

We now introduce the approximation spaces for the computed solutions using the HDG method:

$$u_h \in W_h^k, \quad W_h^k := \{w \in L^2(\mathcal{T}_h) : w|_K \in \mathcal{P}_k(K),\ \forall K \in \mathcal{T}_h\} \tag{4}$$

$$\mathbf{q}_h \in \mathbf{V}_h^k, \quad \mathbf{V}_h^k := \{\mathbf{v} \in [L^2(\mathcal{T}_h)]^d : \mathbf{v}|_K \in [\mathcal{P}_k(K)]^d,\ \forall K \in \mathcal{T}_h\} \tag{5}$$

$$\hat{u}_h \in M_h^k, \quad M_h^k := \{\mu \in L^2(\mathcal{E}_h) : \mu|_F \in \mathcal{P}_k(F),\ \forall F \in \mathcal{E}_h\} \tag{6}$$

where $\mathcal{T}_h$ is the triangulation, K refers to element K, $\mathcal{P}_k$ is the space of polynomials up to degree k, $\mathcal{E}_h$ is the set of faces, and F refers to face F.

Weak formulation

For the weak formulation of the problem, we consider each element separately on the one hand, and each faces separately on the other hand. Hence, for element K, if we know the value of the solution on ∂K, say $\hat{u}$, we can solve for the solution in the interior of element K; this is known as the local problem. On the other hand, we also need to solve for $\hat{u}$ on all the faces of $\mathcal{T}_h$, which is known as the global problem. Let's see in more detail the local and global problems.

Local problem

The weak formulation of the local problem in element K is as follows. Given $\hat{u}_h$, which is an approximation for $\hat{u}$ on ∂K, find $(\mathbf{q}_h, u_h) \in [\mathcal{P}_k(K)]^d \times \mathcal{P}_k(K)$ such that:

$$(\kappa^{-1}\mathbf{q}_h, \mathbf{v})_K + (u_h, \nabla \cdot \mathbf{v})_K - \langle \hat{u}_h, \mathbf{v} \cdot \mathbf{n} \rangle_{\partial K} = 0, \quad \forall \mathbf{v} \in [\mathcal{P}_k(K)]^d \tag{7}$$

$$(\mathbf{q}_h, \nabla w)_K - (\mathbf{c}u_h, \nabla w)_K - \langle \mathbf{q} \cdot \mathbf{n}, w \rangle_{\partial K} + \langle (\mathbf{c} \cdot \mathbf{n} - \tau)\hat{u}_h, w \rangle_{\partial K} + \langle \tau u_h, w \rangle_{\partial K} = (f, w)_K, \ \forall w \in \mathcal{P}_k(K) \tag{8}$$

where $\tau > 0$ is the stabilization parameter.

Global problem

The global problem solves for $\hat{u}_h$, imposing the boundary conditions and the jump condition for the flux across elements. In particular; find $\hat{u}_h \in M_h^k$ such that:

$$\left\langle \hat{\mathbf{f}}_h \cdot \mathbf{n}, \mu \right\rangle_{\partial \mathcal{T}_h \setminus \partial\Omega} + \langle \hat{u}_h - g_D, \mu \rangle_{\partial \Omega_D} + \left\langle \hat{\mathbf{f}}_h \cdot \mathbf{n} - g_N, \mu \right\rangle_{\partial \Omega_N} = 0, \ \forall \mu \in M_h^k \tag{9}$$

where $\hat{\mathbf{f}}_h = \mathbf{q}_h - \mathbf{c}\hat{u} - \tau(u_h - \hat{u}_h)\mathbf{n}$.

Complete formulation and matricial form

The discretization of weak formulation for the local and global problems leads to the following matricial form:

$$\begin{bmatrix} A & B & C \\ -B^T & D & -E \\ C^T & -E^T & M \end{bmatrix} \begin{bmatrix} U \\ Q \\ \hat{U} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ F \\ G \end{bmatrix} \tag{10}$$

where the matrix:

$$\begin{bmatrix} A & B \\ -B^T & D \end{bmatrix} \tag{11}$$

is block-diagonal and invertible if $\tau > 0$. and where U, Q and $\hat{U}$ are the vectors of unknowns for u_h $\mathbf{q}_h$ ($Q = [Q_x, Q_y]^T$), and $\hat{u}$, respectively. Hence, we can eliminate U and Q and form the following system for $\hat{U}$:

$$H\hat{U} = R \tag{12}$$

where:

$$H = M + \begin{bmatrix} C^T & -E^T \end{bmatrix} \begin{bmatrix} A & B \\ -B^T & D \end{bmatrix}^{-1} \begin{bmatrix} C \\ E \end{bmatrix} \tag{13}$$

$$R = G - \begin{bmatrix} C^T & -E^T \end{bmatrix} \begin{bmatrix} A & B \\ -B^T & D \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ F \end{bmatrix} \tag{14}$$

where all the matrices and vectors are built according to the different terms in Eqs. 7 - 9.

Once the global problem is solved, the local problem on each element can be solved to finally get the complete solution of the problem.

Local Postprocessing

Due to the way we solve the problem, both u_h and $\mathbf{q}_h$ are approximated with polynomials of the same order, and so the solutions are of the same order. Hence, since $\mathbf{q} = \nabla u$, we can integrate $\mathbf{q}_h$ to obtain a better solution for u than it is u_h, name it u_h^*. This process is known as local postprocessing. In particular, we want $u_h^* \in \mathcal{P}_{k+1}(K)$ such that on every element $K \in \mathcal{T}_h$:

$$(\kappa \nabla u_h^*, \nabla w)_K = (\mathbf{q}_h, \nabla w)_K, \quad \forall w \in \mathcal{P}_{k+1}(K) \tag{15}$$

$$(u_h^*, 1)_K = (u_h, 1)_K \tag{16}$$

where the second condition is imposed to avoid a singular system, since the problem is a pure Neumann problem.

Implementation Aspects

The extension of the code consists essentially on three new functions: *localprob()*, *hdg_ solve* and *hdg_ postprocess*. The function *localprob()* was already given, and was slightly modified to allow for different values of τ for interior or boundary faces. What this function does is to solve the local problem for an element, given $\hat{u}_h$ on its boundary, and given F.

On the other hand, *hdg_ solve* solves first the global problem, and then uses its solution to call *localprob()* for each element, computing then the complete solution.

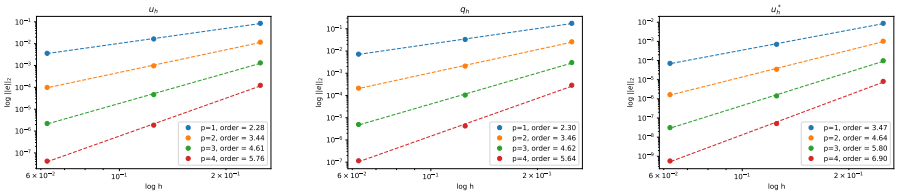
Finally, the output of *hdg_ solve* is passed to *hdg_ postprocess*, which recover u_h^* by performing the local postprocessing explained in the pervious subsection.

Convergence Study for the Diffusion Problem

We first tested the implementation on the pure diffusion equation (i.e. $\mathbf{c} = \mathbf{0}$) with homogeneous Dirichlet boundary conditions. For this, we used the unit square domain, and set the source term f such that the exact solution is:

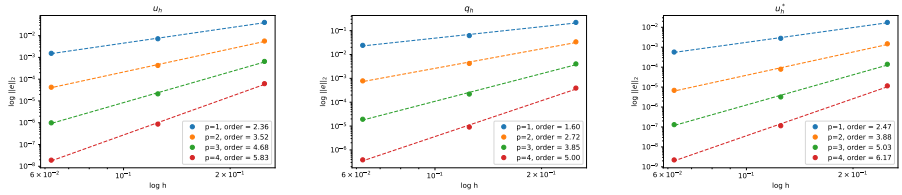
$$u(x, y) = \sin(\pi x) \sin(\pi y) \tag{17}$$

We used three different values for the stabilization parameter τ: 1, h and $1/h$, where h is a measure of the element size. The L_2 norm the errors for u_h, $\mathbf{q}_h$ and u_h^* are shown in Figs. 1 - 3, for the different τ cases.



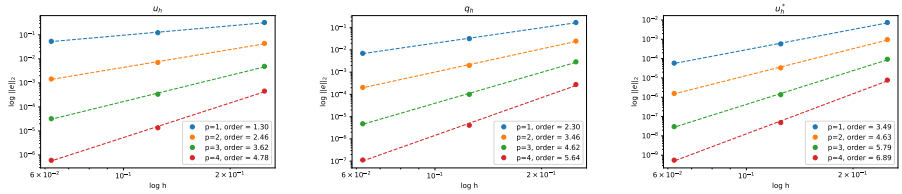
(a) u_h. (b) $\mathbf{q}_h$. (c) u_h^*.

Figure 1: $\tau = 1$



(a) u_h. (b) $\mathbf{q}_h$. (c) u_h^*.

Figure 2: $\tau = 1/h$.



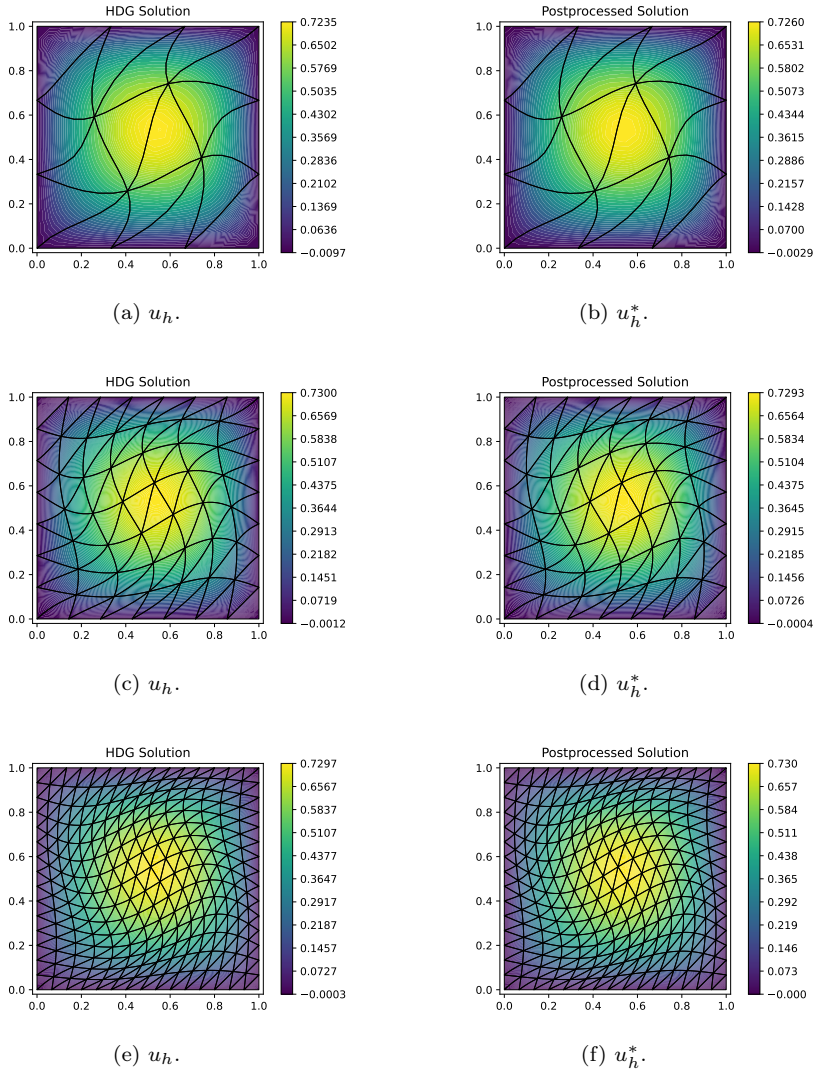
(a) u_h. (b) $\mathbf{q}_h$. (c) u_h^*.

Figure 3: $\tau = h$.

For the case $\tau = 1$, we obtain optimal convergence for u_h and $\mathbf{q}_h$ (i.e. $p+1$), and superconvergence for u_h^* (at least $p+2$). For the case $\tau = 1/h$, we obtain optimal convergence for u_h but not for $\mathbf{q}_h$ on the lower orders. Hence, for the lower orders we do not obtain superconvergence for u_h^* either, although we obtain superconvergence for $p = 3$ and $p = 4$. Finally, for the case $\tau = h$ we do not obtain optimal convergence for u_h, but we do for $\mathbf{q}_h$. Therefore, since u_h^* is obtained from $\mathbf{q}_h$, we get superconvergence for u_h^*.

Comparison between the HDG solution and the Postprocessed Solution for the Convection-Diffusion Problem

Secondly, we tested the implementation with the convection-diffusion problem, for three cases: diffusion-dominant $\mathbf{c} = (1, 1)$, convection diffusion $\mathbf{c} = (10, 10)$, and convection dominant $\mathbf{c} = (100, 100)$. In particular, we compared the solution u_h with the postprocessed solution u_h^* for curved meshes and different grid sizes on each case, using again the unit square domain with homogeneous Dirichlet boundary conditions, and using $f = 10$. Also, the stabilization parameter used was $\tau = 1 + |\mathbf{c} \cdot \mathbf{n}|$. The results are shown in Figs. 4-6.

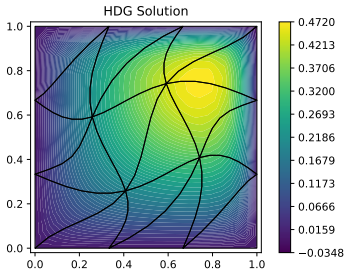


(a) u_h. (b) u_h^*.

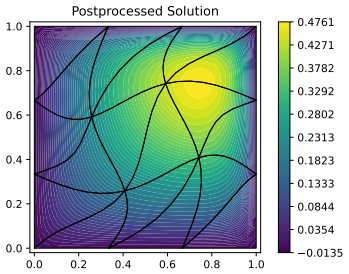
(c) u_h. (d) u_h^*.

(e) u_h. (f) u_h^*.

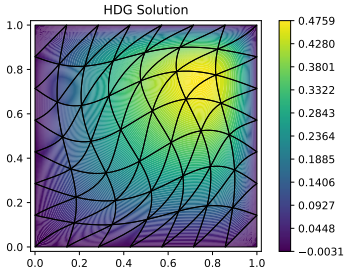
Figure 4: $\mathbf{c} = (1, 1)$



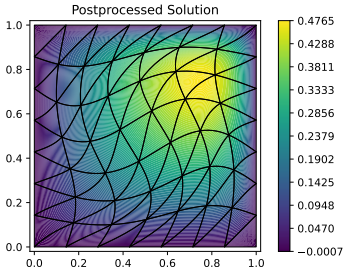
(a) u_h.



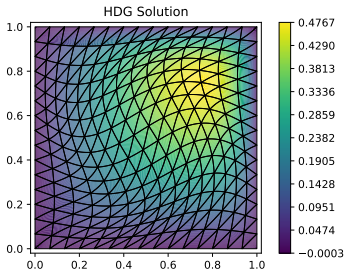
(b) u_h^*.



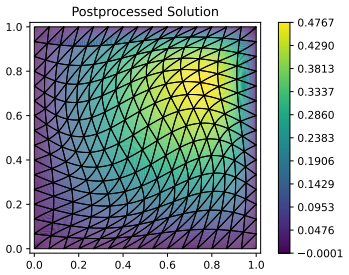
(c) u_h.



(d) u_h^*.

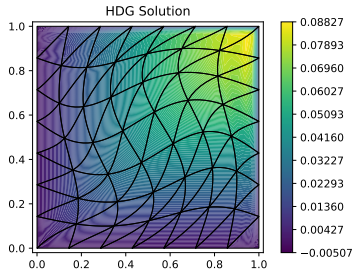


(e) u_h.

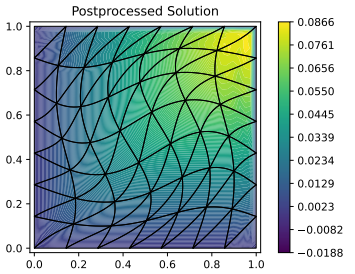


(f) u_h^*.

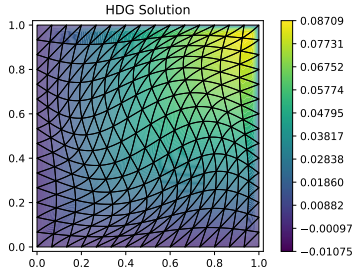
Figure 5: $\mathbf{c} = (10, 10)$



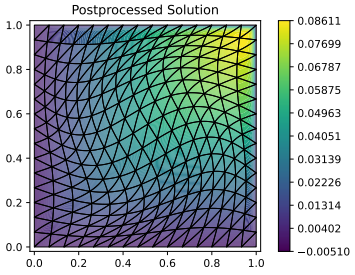
(a) u_h.



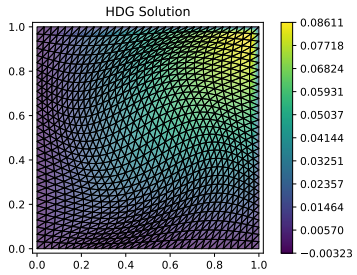
(b) u_h^*.



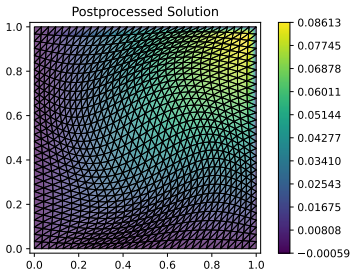
(c) u_h.



(d) u_h^*.



(e) u_h.



(f) u_h^*.

Figure 6: $\mathbf{c} = (100, 100)$

It can be seen that the postprocessed solution is smoother than the solution coming directly from the HDG method, with this more noticeable in cases with coarse grids.

Final Project Proposal

The main idea for the final project is to extend the code to solve the time-dependent 1D Burgers Equation. The time-like variable will be treated as a second spatial dimension, and hence no need to march the solution in time. However, since the equation is non-linear, a non-linear solver will have to be implemented. After that, a point-wise sensor for shock capturing based on the

entropy will be tested.

Add 1D Unsteady Inviscid Burgers Equation

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = \alpha(x)u \tag{18}$$

The source term might be needed to localize the shock and avoid a singular system. The idea is to solve the equation as a 2D steady problem, implementing also a non-linear solver.

Entropy Based Shock Sensor

We motivate the idea by looking at the Burgers Equation with a thermoviscous difussion term:

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = \frac{\partial}{\partial x}\left(\frac{1}{\Gamma}\frac{\partial u}{\partial x}\right) \tag{19}$$

Multiplying by u:

$$u\frac{\partial u}{\partial t} - u\frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = u\frac{\partial}{\partial x}\left(\frac{1}{\Gamma}\frac{\partial u}{\partial x}\right) \tag{20}$$

$$u\frac{\partial u}{\partial t} - u\frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) - \frac{\partial}{\partial x}\left(\frac{u}{\Gamma}\frac{\partial u}{\partial x}\right) = -\frac{1}{\Gamma}\left(\frac{\partial u}{\partial x}\right)^2 \leq 0 \tag{21}$$

From that, for the inviscid case:

$$\eta := u\frac{\partial u}{\partial t} - u\frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) \leq 0 \rightarrow \text{Entropy production} \tag{22}$$

The hypothesis is that a shock will produce non-physical results, making $\eta > 0$ in the shock neighborhood. So, we should add artificial viscosity where $\eta > 0$. The new PDE is:

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = \mathrm{AV}(u, H) \tag{23}$$

$$u\left[\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right)\right] = u\mathrm{AV}(u, H) \tag{24}$$

We have to design $u\mathrm{AV}(u, H)$ such that:

$$u\mathrm{AV}(u, H) \leq 0 \tag{25}$$

One option could be physical AV:

$$\mathrm{AV} = \frac{\partial}{\partial x}\left(\frac{1}{\Gamma_{\mathrm{AV}}}\frac{\partial u}{\partial x}\right) \tag{26}$$

and so:

$$u\mathrm{AV} = u\frac{\partial}{\partial x}\left(\frac{1}{\Gamma_{\mathrm{AV}}}\frac{\partial u}{\partial x}\right) = \frac{\partial}{\partial x}\left(\frac{u}{\Gamma_{\mathrm{AV}}}\frac{\partial u}{\partial x}\right) - \frac{1}{\Gamma_{\mathrm{AV}}}\left(\frac{\partial u}{\partial x}\right)^2 \tag{27}$$

All togheter:

$$u\left[\frac{\partial u}{\partial t}-\frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right)\right]=\frac{\partial}{\partial x}\left(\frac{u}{\Gamma_{\mathrm{AV}}}\frac{\partial u}{\partial x}\right)-\frac{1}{\Gamma_{\mathrm{AV}}}\left(\frac{\partial u}{\partial x}\right)^2 \tag{28}$$

We want:

$$\frac{1}{\Gamma_{\mathrm{AV}}}\left(\frac{\partial u}{\partial x}\right)^2>\eta^+ \Rightarrow \frac{1}{\Gamma_{\mathrm{AV}}}>\frac{\eta^+}{(\partial u/\partial x)^2} \tag{29}$$

where we define:

$$\eta^+=\max(\eta,0) \tag{30}$$

So, we can compute η^+ point-wise and add this artificial viscosity where needed.

In case of having extra time (unlikely)

Solving for the Adjoint

We could take advantage of the non-linear solver to also solve for the dual problem and get the adjoint solution. The final piece would be to implement an adjoint based error estimate, which should also take into account properly the addition of artificial viscosity, which is not in the actual equation we are solving.