Finite Elements in Fluids, Assignment 5, Hybridizable Discontinuous Galerkin

Jose Raul Bravo Martinez, MSc Computational Mechanics

June 1, 2019

Consider the domain $\Omega = [0,1]^2$ such that $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_R$ with $\Gamma_D \cap \Gamma_N = 0$, $\Gamma_D \cap \Gamma_R = 0$, $\Gamma_N \cap \Gamma_R = 0$. More precisely, set:

$$\Gamma_N := \{(x,y) \in \mathbb{R}^2 : y = 0\}$$

$$\Gamma_R := \{(x,y) \in \mathbb{R}^2 : x = 1\}$$

$$\Gamma_D := \partial\Omega \setminus (\Gamma_N \cup \Gamma_R)$$

The following second-order linear scalar partial differential equation is defined

$$\begin{cases} -\nabla \cdot (\kappa \nabla u) = s & \text{in } \Omega, \\ u = u_D & \text{on } \Gamma_D, \\ \boldsymbol{n} \cdot (\kappa \nabla u) = t & \text{on } \Gamma_N, \\ \boldsymbol{n} \cdot (\kappa \nabla u) + \gamma u = g & \text{on } \Gamma_R, \end{cases}$$

where κ and γ are the diffusion and convection coefficients, respectively. $\boldsymbol{n}$ is the outward unit normal vector to the boundary, s is the volumetric source term and u_D, t, and g are the Dirichlet, Neumman and Robin data imposed on the corresponding portions of the boundary $\partial\Omega$.

1. Write the HDG formulation of the problem (P). More precisely, derive the HDG strong and weak forms of the local and global problems.

Local

$$\begin{cases} \nabla \cdot \boldsymbol{q}_i = s & \text{in } \Omega_i \\ \boldsymbol{q}_i + \kappa \nabla u_i = \boldsymbol{0} & \text{in } \Omega_i \\ u_i = u_D & in \text{ on } \partial\Omega_i \cap \Gamma_D \\ u_i = \hat{u} & \text{on } \partial\Omega_i \setminus \Gamma_D \end{cases}$$

Global

$$\begin{cases} [\![\boldsymbol{n} \cdot \boldsymbol{q}]\!] = 0 & \text{on } \Gamma \\ \boldsymbol{n} \cdot \boldsymbol{q} = -t & \text{on } \Gamma_N \\ -\boldsymbol{q} \cdot \boldsymbol{n} + \gamma u = g & \text{on } \Gamma_R \end{cases}$$

After discretizaiton of the local equations, one obtains the local system element by element:

$$\begin{bmatrix} \boldsymbol{A}_{uu} & \boldsymbol{A}_{uq} \\ \boldsymbol{A}_{uq}^T & \boldsymbol{A}_{qq} \end{bmatrix}_i \begin{bmatrix} \boldsymbol{u_i} \\ \boldsymbol{q_i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f_u} \\ \boldsymbol{f_q} \end{bmatrix} + \begin{bmatrix} \boldsymbol{A}_{u\hat{u}} \\ \boldsymbol{A}_{q\hat{u}} \end{bmatrix}_i \hat{u}$$

After discretization of the global equations, one obtains:

$$\sum_{e=1}^{n_{el}} \left\{ \begin{bmatrix} \boldsymbol{A}_{u\hat{u}}^T & \boldsymbol{A}_{q\hat{u}}^T \end{bmatrix}_i \begin{bmatrix} \boldsymbol{u_i} \\ \boldsymbol{q_i} \end{bmatrix} + \begin{bmatrix} \boldsymbol{A}_{\hat{u}\hat{u}} \end{bmatrix}_i \hat{\boldsymbol{u}} \right\} = \sum_{e=1}^{n_{el}} \left\{ \begin{bmatrix} \boldsymbol{f}_{\hat{u}} \end{bmatrix}_i \right\}$$

Where the term $\boldsymbol{A}_{\hat{u}\hat{u}}$ is affected by the Robin boundary condition; while $\boldsymbol{f}_{\hat{u}}$ comes from both the Neumann and Robin boundary conditions. Namely,

$$\boldsymbol{A}_{\hat{u}\hat{u}} = - < \hat{v}, \tau_e \hat{u}_e^h >_{\partial\Omega_e \setminus \Gamma_D} - < \hat{v}, \gamma \hat{u}_e^h >_{\partial\Omega_e \cap \Gamma_R}$$

$$\boldsymbol{f}_{\hat{u}} = < \hat{v}, -g >_{\partial\Omega_e \cap \Gamma_R} + < \hat{v}, -t >_{\partial\Omega_e \cap \Gamma_N}$$

The rest of the terms remain the same as the derivation shown in class with only Dirichlet BCs.

The global system to solve is then:

$$\hat{\boldsymbol{K}}\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}}$$

Where

$$\hat{\boldsymbol{K}} = \Pi_{i=1}^{nel} \begin{bmatrix} \boldsymbol{A}_{u\hat{u}}^T & \boldsymbol{A}_{q\hat{u}}^T \end{bmatrix}_i \begin{bmatrix} \boldsymbol{A}_{uu} & \boldsymbol{A}_{uq} \\ \boldsymbol{A}_{uq}^T & \boldsymbol{A}_{qq} \end{bmatrix}_i^{-1} \begin{bmatrix} \boldsymbol{A}_{u\hat{u}} \\ \boldsymbol{A}_{q\hat{u}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{A}_{\hat{u}\hat{u}} \end{bmatrix}_i$$

$$\hat{\boldsymbol{f}} = \Pi_{i=1}^{nel} \begin{bmatrix} \boldsymbol{f}_{\hat{u}} \end{bmatrix}_i - \begin{bmatrix} \boldsymbol{A}_{u\hat{u}}^T & \boldsymbol{A}_{q\hat{u}}^T \end{bmatrix}_i \begin{bmatrix} \boldsymbol{A}_{uu} & \boldsymbol{A}_{uq} \\ \boldsymbol{A}_{uq}^T & \boldsymbol{A}_{qq} \end{bmatrix}_i^{-1} \begin{bmatrix} \boldsymbol{f}_u \\ \boldsymbol{f}_q \end{bmatrix}_i$$

2. Implement in the Matlab code provided in class the corresponding HDG solver.

In order to implement the HDG solver, the first thing to change was the way to recognize the exterior faces. Before, all exterior faces were listed together, as the code was solving a pure Dirichlet problem. Now, three lists are created, one for Neumann, one for Robin and the last one for Dirichlet faces. Both, GetFaces.m and Preprocess.m were modified.

```
%%%% Find the Neumann and Robin BCs %%%%%%%%
[intFaces,extFaces,extNFaces,extRFaces] = GetFaces2(T(:,1:3), X);

infoFaces.intFaces = intFaces;
infoFaces.extDFaces = extFaces;
infoFaces.extNFaces = extNFaces;
infoFaces.extRFaces = extRFaces;
```

Once the Robin and Neumann faces are identified, modifications had to be implemented in HDGMatrixPoisson.m. Applying the corresponding elemental contributions to the system only for the Neumann and Robin faces. So, inside the face loop:

```
if iface==THENeumannFace
    %NeumanContribution
    %Calculate position of Gauss Point
    X_NeumanFace = N1d*Xf;
    %Sum over Gauss Points
        faux=zeros(nOfFaceNodes,1);
        for ig = 1:length(IPw_f)
            N_ig    = N1d(ig,:);
            %Obtain the traction t at the Gauss point
            traction=BottomNeumannBCPoisson(X_NeumanFace(ig,:));
            dlength=dline(ig);
            faux = faux - traction*N_ig'*dlength;
        end
      %Assemble Neumann Contribution
     f1(ind_face,1)=faux;
end
```

```
if iface==THERobinFace
    All(ind_face,ind_face) = All(ind_face,ind_face) - N1d'*(spdiags(dline,0,ngf,ngf)*N1d)*4; %hardcoding gamma = 4
    %RobinContribution
    %Calculate position of Gauss Point
    X_RobinFace = N1d*Xf;
    %Sum over Gauss Points
        faux=zeros(nOfFaceNodes,1);
        for ig = 1:length(IPw_f)
            N_ig    = N1d(ig,:);
            %Obtain the g at the Gauss point
            g=RightRobinBCPoisson(X_RobinFace(ig,:));
            %
            dlength=dline(ig);
            faux = faux - g*N_ig'*dlength;
        end
      %Assemble Robin Contribution
     f1(ind_face,1)=faux;
end
```

Moreover, the identification of the DOFs had to change, to not take the Neumann and Robin boundaries as known values. This changes are implemented in MainPoissonHDG.m.

```
%Changing the way DOFs are considered
%%%%%%%%%%%%%%%% Identifiying DOFs  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dofNeumann=[];
for i=1:nOfExteriorNFaces
    FaceNumber=(F(infoFaces.extNFaces(i,1),infoFaces.extNFaces(i,2)));
    for j=0:nOfFaceNodes-1
        dofNeumann = [dofNeumann,FaceNumber*nOfFaceNodes-(degree-j)];
    end
end
dofDirichlet= setdiff(dofDirichlet,dofNeumann);
dofRobin=[];
for i=1:nOfExteriorRFaces
    FaceNumber=(F(infoFaces.extRFaces(i,1),infoFaces.extRFaces(i,2)));
    for j=0:nOfFaceNodes-1
        dofRobin = [dofRobin,FaceNumber*nOfFaceNodes-(degree-j)];
    end
end
dofDirichlet= setdiff(dofDirichlet,dofRobin);
dofUnknown =[dofUnknown,dofNeumann,dofRobin];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Finally, on the function HDGPostprocess.m, the variable mu (which in this example is κ) had to be entered to multiply the system.

```
87
88          %Contribution of the current integration point to the elemental matrix
89 -        Bq = Bq + (Nx_g'*qx_g + Ny_g'*qy_g)*dvolu;
90 -        K = K + mu*(Nx_g'*Nx_g + Ny_g'*Ny_g)*dvolu;
91 -        int_u_star = int_u_star + N_g'*dvolu;
92 -        int_u = int_u + u_g*dvolu;
93 -    end
94
```

3. Set $\kappa = 3.5$ and $\gamma = 4$. Consider $u(x, y) = cosh(ax + by) - exp(sin(\gamma\pi x + \kappa\pi x))$ with a=0.3 and b=2. Determine the analytical expressions for the data u_D, t, and g in problem P.

Using the symbolic calculus tools of matlab, one can easily obtain the expressions for t and g. The following script returns the analytical expressions necessary:

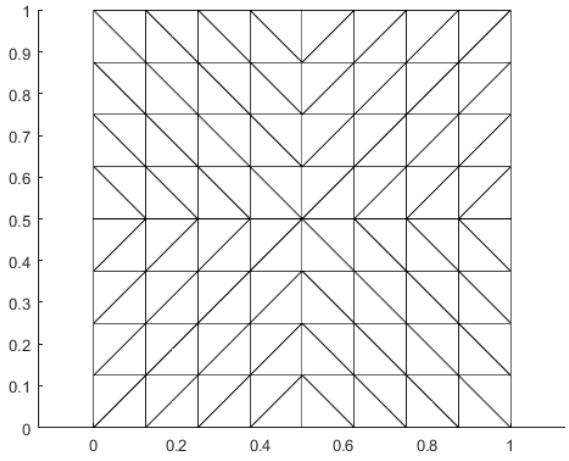
```
1 -  syms a b kappa gamma x y
2
3 -  u = cosh(a*x + b*y) - exp(sin(gamma*pi*x + kappa*pi*x));
4
5    %%% Grad u wrt x
6 -  dudx=diff(u,x);
7    %%% Grad u wrt y
8 -  dudy=diff(u,y);
9
10 - q=-kappa*[dudx;dudy];
11 - t=-kappa*dudy;
12 - g=kappa*dudx + gamma*u;
```

One then has to create functions that return the correct value for the given face. In the code the RighRobinBC.m and ButtomNeumannBC.m. In order to make the functions work, all of the multiplication symbols in the expression "*" need to be changed by ".*". The example of RighRobinBC.m is shown next:

```
1    function g = RightRobinBCPoisson(X)
2    % % % % % Parameters
3 -  kappa = 3.5;
4 -  gamma = 4;
5 -  a=0.3;
6 -  b=2;
7    % Points
8 -  x = X(:,1);
9 -  y = X(:,2);
10 - g=gamma.*(cosh(a.*x + b.*y) - exp(sin(pi.*gamma.*x + pi.*kappa.*x))) + ...
11       kappa.*(a.*sinh(a.*x + b.*y) - exp(sin(pi.*gamma.*x + ...
12       pi.*kappa.*x)).*cos(pi.*gamma.*x + pi.*kappa.*x).*(pi.*gamma + pi.*kappa));
```

4. Solve problem (P) using HDG with different meshses and polynomial degrees of approximation. Starting fromt the plots provided by the Matlab code, discuss the accuracy of the obtained solution u and of the post processed one $u*$.

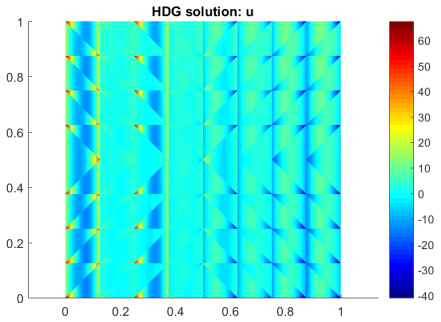
A comparison is done with the same mesh (Mesh 3) for different polynomial approximations. The mesh is shown next:



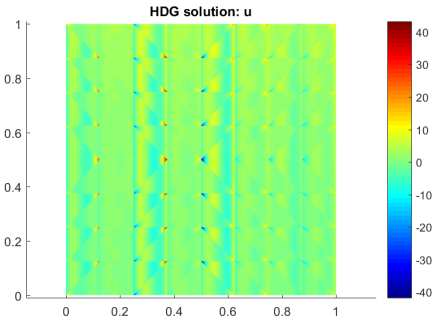
From figures 1 and 2 it can be seen without effort that the results for u* are much better. One can quantitatively see the improvement when taking a look at the following table:

L2 Error of the variables		
	u	u*
Degree 2	6.236559e+00	5.964020e-02
Degree 3	2.635081e+00	1.916840e-02
Degree 4	1.128827e+00	6.162010e-03

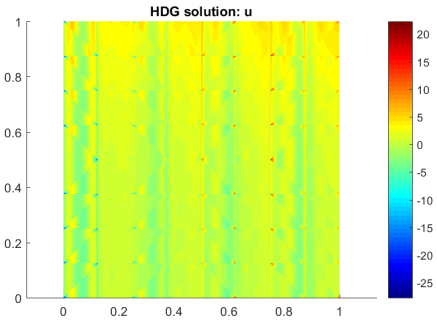
It was observed as well, that discontinuities are evident in the solution u, while the field is smoother for the solution u*. This is one of the characteristics of the HDG method. The discontinuities tend to zero, as the solution improves, therefore, discontinuities are an indicator of how good a solution is.



(a) degree 2

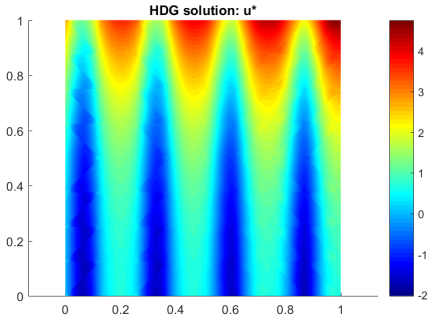


(b) degree 3

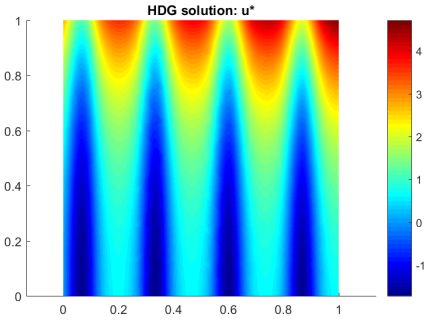


(c) degree 4

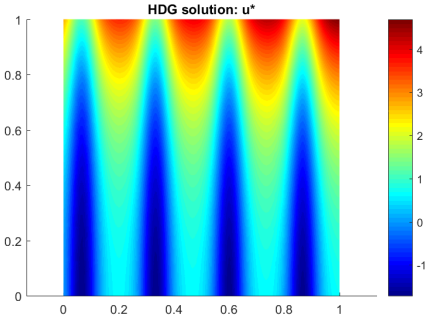
Figure 1: Results for u



(a) degree 2



(b) degree 3



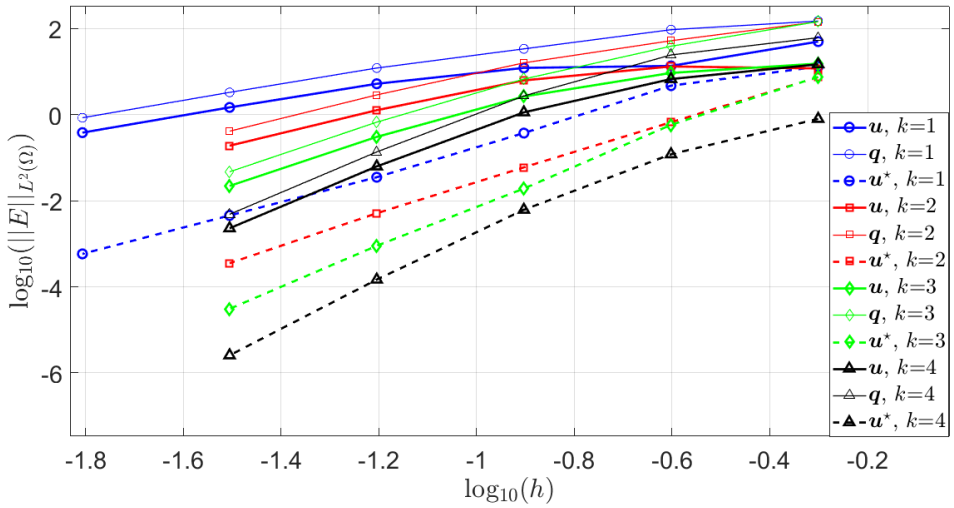
(c) degree 4

Figure 2: Results for u star

5. Compute the errors for u, **q**, and u* in the L2-norm defined on the domain Ω. Perform a convergence study for the primal, u, mixed **q** and postprocessed, u* variables for a polynomial degree of approximation k=1, ..., 4. Discuss the obtained numerical results starting from the theoretical results on the optimal convergence rates of HDG.

For this point, a function for the analytical expression of q had to be implemented. Recall that $\boldsymbol{q} = -\kappa \nabla u$.

Moreover, some modifications had to be made on the L2 norm function, since the result is in this case a vector quantity. After which one can obtain the following plot for the convergence of the quantities:



Notice how $\boldsymbol{q}$ and u have a very similar convergence behavior; while the post-processed variable u* is converging faster.

Convergence for different polynomial degrees			
	u	$\boldsymbol{q}$	u*
Degree 1	1.9464	1.9581	2.9644
Degree 2	2.7401	2.7984	3.8556
Degree 3	3.7815	3.8260	4.8639
Degree 4	4.7707	4.8135	5.8407

The table shows the convergence for the finest mesh of the polynomial degree it refers to. One can observe that the theoretical values of k+1 for the primal and mixed variables; and k+2 for the post-processed variable, are obtained.