UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Swansea University Prifysgol Abertawe

CENTRALE NANTES

Universität Stuttgart

MSc in Computational Mechanics

Finite Elements in Fluids

MATLAB Assignment (version 11): Hybridisable Discontinuous Galerkin

Submitted by:
Mario Alberto Méndez Soto

Submitted to:
Prof. Matteo Giacomini
Prof. Antonio Huerta

Spring Semester, 2019

1 Problem statement

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

$$\begin{aligned}\Gamma_N &:= \{(x,y) \in \mathbb{R}^2 \ : \ y = 0\} \\ \Gamma_R &:= \{(x,y) \in \mathbb{R}^2 \ : \ x = 1\} \\ \Gamma_D &:= \partial\Omega \backslash (\Gamma_N \cup \Gamma_R)\end{aligned}$$

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

$$\begin{cases} -\nabla \cdot (\kappa \nabla u) = s & \text{in } \Omega, \\ u = u_D & \text{on } \Gamma_D \\ \mathbf{n} \cdot (\kappa \nabla u) = t & \text{on } \Gamma_N \\ \mathbf{n} \cdot (\kappa \nabla u) + \gamma u = g & \text{on } \Gamma_R \end{cases} \tag{1.1}$$

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

1. Write the HDG formulation of the problem (1.1). More precisely, derive the HDG strong and weak forms of the local and global problems.[*Hint: the hybrid variable $\hat{u}$ needs to be introduced on both on Γ_N and Γ_R*]

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

3. Set $\kappa = 5$ and $\gamma = 3$. Consider $u(x,y) = sinh(asin(\kappa\pi x) + bcos(\pi(x + \gamma y)))$, with $a = 1.2$ and $b = 0.75$. Determine the analytical expressions of the data u_D, t and g in problem (1.1). [*Hint: Use Matlab tools for symbolic calculus*]

4. Solve problem (1.1) using HDG with different meshes and polynomial degrees of approximation. Starting from the plots provided by the Matlab code, discuss the accuracy of the obtained solution u and of the post-processed one u^*.

5. Compute the errors for u, $\mathbf{q}$, and u^* in the $\mathcal{L}_2$-norm defined in the domain Ω. Perform a convergence study for the primal, u, mixed, $\mathbf{q}$ and post-processed, 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.

2 HDG formulation

An equivalent strong form to the problem in (1.1) can be written in the broken computational domain as:

$$\begin{cases} -\nabla \cdot (\kappa \nabla u) = s & \text{in } \Omega_i, \text{ and for } i = 1, ..., n_{el}, \\ u = u_D & \text{on } \Gamma_D \\ \mathbf{n} \cdot (\kappa \nabla u) = t & \text{on } \Gamma_N \\ \mathbf{n} \cdot (\kappa \nabla u) + \gamma u = g & \text{on } \Gamma_R \\ [\![u\mathbf{n}]\!] = \mathbf{0} & \text{on } \Gamma \\ [\![\mathbf{n} \cdot \nabla u]\!] = 0 & \text{on } \Gamma \end{cases} \tag{2.1}$$

Then, the problem can be solved as two separate problems. First, the local problem with Dirichlet boundary conditions is defined as:

$$\begin{cases} \nabla \cdot \mathbf{q}_i = s & \text{in } \Omega_i, \\ \mathbf{q}_i + \kappa \nabla u_i = \mathbf{0} & \text{in } \Omega_i, \\ u_i = u_D & \text{on } \partial\Omega_i \cap \Gamma_D, \\ u_i = \hat{u} & \text{on } \partial\Omega_i \backslash \Gamma_D, \end{cases} \tag{2.2}$$

for $i = 1, .., n_{el}$.

Furthermore, a global problem where a primal variable $\hat{u}$ is imposed on both Γ_N and Γ_R can be derived. Thus, this global problem acquires the following form:

$$\begin{cases} [\![\mathbf{n} \cdot \mathbf{q}]\!] = 0 & \text{on } \Gamma \\ \mathbf{n} \cdot \mathbf{q} = -t & \text{on } \Gamma_N \\ \mathbf{n} \cdot \mathbf{q} = \gamma \hat{u} - g & \text{on } \Gamma_R \end{cases} \tag{2.3}$$

The condition $[\![u\mathbf{n}]\!] = \mathbf{0}$ is not explicitly considered since it is imposed automatically because $\hat{u}$ is unique for adjacent elements.

For the derivation of the weak forms, the following scalar and vectors spaces will be used:

$$\begin{aligned} \mathcal{W}(D) &= \{\mathbf{w} \in [\mathcal{H}^1(D)]^2, D \subset \Omega\} \\ \mathcal{V}(D) &= \{v \in \mathcal{H}^1(D), D \subset \Omega\} \\ \mathcal{M}(S) &= \{\mu \in \mathcal{L}_2(S), S \subset \Gamma \cup \partial\Omega\} \end{aligned}$$

Thus, given u_D on Γ_D and $\hat{u}$ on $\Gamma \cup \Gamma_N \cup \Gamma_R$, the weak formulation of the local problem aims to find $(\mathbf{q}_i, u_i) \in \mathcal{W}(\Omega_i) \times \mathcal{V}(\Omega_i)$ that satisfies:

$$\begin{aligned} -(\nabla v, \mathbf{q}_i)_{\Omega_i} + \langle v, \mathbf{n}_i \cdot \hat{\mathbf{q}}_i \rangle_{\partial\Omega_i} &= (v, s)_{\Omega_i} \\ -(\mathbf{w}, \mathbf{q}_i)_{\Omega_i} + \kappa(\nabla \cdot \mathbf{w}, u_i)_{\Omega_i} &= \kappa \langle \mathbf{n}_i \cdot \mathbf{w}, u_D \rangle_{\partial\Omega_i \cap \Gamma_D} + \kappa \langle \mathbf{n}_i \cdot \mathbf{w}, \hat{u} \rangle_{\partial\Omega_i \backslash \Gamma_D} \end{aligned} \tag{2.4}$$

where the numerical traces of the fluxes $\hat{\mathbf{q}}_i$ are defined, for stability purposes, as:

$$\mathbf{n}_i \cdot \hat{\mathbf{q}}_i := \begin{cases} \mathbf{n}_i \cdot \mathbf{q}_i + \tau_i(u_i - u_D) & \text{on } \partial\Omega_i \cap \Gamma_D \\ \mathbf{n}_i \cdot \mathbf{q}_i + \tau_i(u_i - \hat{u}) & \text{elsewhere} \end{cases} \tag{2.5}$$

Similarly, the weak form of the global problem is defined simply as finding $\hat{u} \in \mathcal{M}(\Gamma \cup \Gamma_N \cup \Gamma_R)$ for all $\mu \in \mathcal{M}(\Gamma \cup \Gamma_N \cup \Gamma_R)$ such that:

$$\sum_{n=1}^{n_{el}} \langle \mu, \mathbf{n}_i \cdot \hat{\mathbf{q}}_i \rangle_{\partial\Omega_i \setminus \partial\Omega} + \sum_{n=1}^{n_{el}} \langle \mu, \mathbf{n}_i \cdot \hat{\mathbf{q}}_i + t \rangle_{\partial\Omega_i \cap \Gamma_N} + \sum_{n=1}^{n_{el}} \langle \mu, \mathbf{n}_i \cdot \hat{\mathbf{q}}_i + g - \gamma\hat{u} \rangle_{\partial\Omega_i \cap \Gamma_R} = 0 \tag{2.6}$$

Replacing the definition of the fluxes (2.5) into the previous equation, the global problem becomes:

$$\sum_{n=1}^{n_{el}} \left\{ \langle \mu, \mathbf{n}_i \cdot \mathbf{q}_i \rangle_{\partial\Omega_i \setminus \Gamma_D} + \langle \mu, \tau_i u_i \rangle_{\partial\Omega_i \setminus \Gamma_D} - \langle \mu, \tau_i \hat{u} \rangle_{\partial\Omega_i \setminus \Gamma_D} - \langle \mu, \gamma\hat{u} \rangle_{\partial\Omega_i \cap \Gamma_R} \right\}$$
$$= -\sum_{n=1}^{n_{el}} \left\{ \langle \mu, g \rangle_{\partial\Omega_i \cap \Gamma_R} + \langle \mu, t \rangle_{\partial\Omega_i \cap \Gamma_N} \right\} \tag{2.7}$$

For the discretization of the local (2.4-2.5) and global (2.7) problems, discrete finite-element spaces $\mathcal{W}^h$, $\mathcal{V}^h$, and $\mathcal{M}^h$ are defined as follows:

$$\begin{array}{rll} \mathcal{W}^h(\Omega) = & \{\mathbf{w} \in [\mathcal{L}_2(\Omega)]^2;\ w|_{\Omega_i} \in [\mathcal{P}^p(\Omega_i)]^2\ \forall\Omega_i\} & \subset \mathcal{W}(\Omega) \\ \mathcal{V}^h(\Omega) = & \{v \in [\mathcal{L}_2(\Omega)]^2;\ v|_{\Omega_i} \in \mathcal{P}^p(\Omega_i)\ \forall\Omega_i\} & \subset \mathcal{V}(\Omega) \\ \mathcal{M}^h(S) = & \{\mu \in [\mathcal{L}_2(S)]^2;\ \mu|_{\Gamma_i} \in \mathcal{P}^p(\Gamma_i)\ \forall\Omega_i \subset S \subset \Gamma \cup \partial\Omega\} & \subset \mathcal{M}(S) \end{array} \tag{2.8}$$

which allows the introduction of element-by-element interpolations of the form:

$$\mathbf{q} \approx \mathbf{q}^h = \sum_{n=1}^{n_{el}} N_j \mathbf{q}_j \quad \in \mathcal{W}^h \tag{2.9}$$

$$u \approx u^h = \sum_{n=1}^{n_{el}} N_j u_j \quad \in \mathcal{V}^h \tag{2.10}$$

$$\hat{u} \approx \hat{u}^h = \sum_{n=1}^{n_{el}} \hat{N}_j \hat{u}_j \quad \in \mathcal{M}^h(\Gamma \cup \Gamma_N \cup \Gamma_R) \text{ or} \mathcal{M}^h(\Gamma) \tag{2.11}$$

Thus, using this interpolation and the matrix notation presented in [1] and [2], the following elemental system of equations emerges:

$$\begin{bmatrix} \mathbf{A_{uu}} & \mathbf{A_{uq}} \\ \kappa\mathbf{A}_{\mathbf{uq}}^T & \mathbf{A_{qq}} \end{bmatrix}_i \begin{bmatrix} \mathbf{u_i} \\ \mathbf{q_i} \end{bmatrix} = \begin{bmatrix} \mathbf{f_u} \\ \kappa\mathbf{f_q} \end{bmatrix}_i + \begin{bmatrix} \mathbf{A_{u\hat{u}}} \\ \kappa\mathbf{A_{q\hat{u}}} \end{bmatrix}_i \hat{\mathbf{u}}_i \tag{2.12}$$

Similarly, applying the interpolation to (2.7) produces the following system of equations:

$$\sum_{n=1}^{n_{el}} \left\{ \begin{bmatrix} \mathbf{A}_{\mathbf{u\hat{u}}}^T & \mathbf{A}_{\mathbf{q\hat{u}}}^T \end{bmatrix}_i \begin{bmatrix} \mathbf{u_i} \\ \mathbf{q_i} \end{bmatrix} + [\mathbf{A_{\hat{u}\hat{u}}}]_i \hat{\mathbf{u}}_i + \left[\mathbf{A}_{\mathbf{\hat{u}\hat{u}}}^R\right]_i \hat{\mathbf{u}}_i \right\} = \sum_{n=1}^{n_{el}} \left\{ [\mathbf{f_{\hat{u}}}]_i + \left[\mathbf{f}_{\mathbf{\hat{u}}}^R\right]_i \right\} \tag{2.13}$$

where matrices $\mathbf{A}^R_{\hat{\mathbf{u}}\hat{\mathbf{u}}}$ and $\mathbf{f}^R_{\hat{\mathbf{u}}}$ are associated to the Robin boundary condition of the problem and can be defined as:

$$\mathbf{A}^R_{\hat{\mathbf{u}}\hat{\mathbf{u}}} = -\sum_{\partial\Omega_i\cap\Gamma_R} \gamma \sum_{g=1}^{n^f_{ip}} \hat{\mathbf{N}}_{\mathbf{n}}(\xi^{\mathbf{f}}_{\mathbf{g}})\hat{\mathbf{N}}^T(\xi^{\mathbf{f}}_{\mathbf{g}})w^f_g$$

$$\mathbf{f}^R_{\hat{\mathbf{u}}} = -\sum_{\partial\Omega_i\cap\Gamma_R} \sum_{g=1}^{n^f_{ip}} \mathbf{N}(\xi^{\mathbf{f}}_{\mathbf{g}})g(\mathbf{x}(\xi^{\mathbf{f}}_{\mathbf{g}}))w^f_g$$

After substituting the local solution (2.12) in (2.13), the global system becomes:

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

with

$$\hat{\mathbf{K}} = \bigA_{i=1}^{n_{el}} \begin{bmatrix} \mathbf{A}^T_{\mathbf{u}\hat{\mathbf{u}}} & \mathbf{A}^T_{\mathbf{q}\hat{\mathbf{u}}} \end{bmatrix}_i \begin{bmatrix} \mathbf{A}_{\mathbf{uu}} & \mathbf{A}_{\mathbf{uq}} \\ \kappa\mathbf{A}^T_{\mathbf{uq}} & \mathbf{A}_{\mathbf{qq}} \end{bmatrix}^{-1}_i \begin{bmatrix} \mathbf{A}_{\mathbf{u}\hat{\mathbf{u}}} \\ \kappa\mathbf{A}_{\mathbf{q}\hat{\mathbf{u}}} \end{bmatrix}_i + [\mathbf{A}_{\hat{\mathbf{u}}\hat{\mathbf{u}}}]_i + \left[\mathbf{A}^R_{\hat{\mathbf{u}}\hat{\mathbf{u}}}\right]_i$$

and

$$\hat{\mathbf{f}} = \bigA_{i=1}^{n_{el}} [\mathbf{f}_{\hat{\mathbf{u}}}]_i + \left[\mathbf{f}^R_{\hat{\mathbf{u}}}\right]_i - \begin{bmatrix} \mathbf{A}^T_{\mathbf{u}\hat{\mathbf{u}}} & \mathbf{A}^T_{\mathbf{q}\hat{\mathbf{u}}} \end{bmatrix}_i \begin{bmatrix} \mathbf{A}_{\mathbf{uu}} & \mathbf{A}_{\mathbf{uq}} \\ \kappa\mathbf{A}^T_{\mathbf{uq}} & \mathbf{A}_{\mathbf{qq}} \end{bmatrix}^{-1}_i \begin{bmatrix} \mathbf{f}_{\mathbf{u}} \\ \kappa\mathbf{f}_{\mathbf{q}} \end{bmatrix}_i$$

3 Computational implementation

Prior to modifying the originally given code, analytical expression for u_D, t, and g were computed using the MATLAB symbolic tools. The results of these computations are presented below.

$$u_D = \begin{cases} \sinh\left(\frac{3\cos(3\pi y)}{4}\right) & \text{for } x = 0 \\ \sinh\left(\frac{6\sin(5\pi x)}{5} + \frac{3\cos(\pi(x+3))}{4}\right) & \text{for } y = 1 \end{cases} \tag{3.1}$$

$$t = 11.25\,\pi \cosh\left(\frac{3\cos(\pi x)}{4} + \frac{6\sin(5\pi x)}{5}\right) \sin(\pi x) \tag{3.2}$$

$$g = 3\sinh\left(\frac{3\cos(\pi\,(3y+1))}{4}\right) - 5\cosh\left(\frac{3\cos(3\pi y)}{4}\right)\left(6\pi - \frac{3\pi\sin(3\pi y)}{4}\right) \tag{3.3}$$

Corresponding expressions were introduced in the existing functions `analyticalPoisson.m` and `sourcePoisson.m` for the computations of the source term and the exact solution, which also includes the computations for the Dirichlet boundary values. Moreover, new analogous functions were added for the mathematical expressions for t and g: `neumanPoisson.m` and `robinPoisson.m`, respectively.

The provided code separates the element faces into internal and external. Using this information, a newly introduced function called `ExtFace_class.m` divides the external boundary faces into Neumann, Dirichlet and Robin. The code implemented is shown in the listing below.

Listing 3.1 – Function *ExtFace_class.m*

```
function infoFaces = ExtFace_class(infoFaces,X,T)
k=1; q=1; m=1;
for i = 1:length(infoFaces.extFaces)
    El_num = T(infoFaces.extFaces(i,1),1:3); %calls the on-edge nodes
    Nod_1 = X(El_num(1),:);
    Nod_2 = X(El_num(2),:);
    Nod_3 = X(El_num(3),:);
    if Nod_1(2) == 0 && Nod_2(2) == 0 || Nod_1(2) == 0 && Nod_3(2)==0
       || Nod_3(2) == 0 && Nod_2(2)==0  % y=0 (Neumann faces)
        infoFaces.extFaces_N(k,:) = infoFaces.extFaces(i,:);
        k = k+1;
    elseif Nod_1(1) == 1 && Nod_2(1) == 1 || Nod_1(1) == 1 && Nod_3(1)
      ==1 || Nod_3(1) == 1 && Nod_2(1)==1 % x=1 (Robin faces)
         infoFaces.extFaces_R(m,:) = infoFaces.extFaces(i,:);
        m=m+1;
    else
        infoFaces.extFaces_D(q,:) = infoFaces.extFaces(i,:);
        q = q+1;
    end
end
end
```

Moreover, the definition of the matrix F, which contains reciprocal information of the faces, was modified to allocate the Dirichlet boundary faces to the highest indices. The listing below shows the introduced changes in function `hdg_preprocess(T,X)`.

Listing 3.2 – Changes introduced in function *hdg_preprocess(T,X)*

```
F = zeros(nOfElements,3);
for iFace = 1:nOfInteriorFaces %numbering of internal faces
    infoFace = intFaces(iFace,:);
    F(infoFace(1),infoFace(2)) = iFace;
    F(infoFace(3),infoFace(4)) = iFace;
end

for iFace = 1:nOfExteriorFaces_N %numbering of Neuman faces
    infoFace = infoFaces.extFaces_N(iFace,:);
    F(infoFace(1),infoFace(2)) = iFace + nOfInteriorFaces;
end

for iFace = 1:nOfExteriorFaces_R %numbering of Robin faces
    infoFace = infoFaces.extFaces_R(iFace,:);
    F(infoFace(1),infoFace(2)) = iFace + nOfInteriorFaces +
       nOfExteriorFaces_N;
end

for iFace = 1:nOfExteriorFaces_D %Numbering of Dirichlet faces
    infoFace = infoFaces.extFaces_D(iFace,:);
    F(infoFace(1),infoFace(2)) = iFace + nOfInteriorFaces +
       nOfExteriorFaces_N +nOfExteriorFaces_R;
end
```

Since the nodes on the Dirichlet boundaries are assigned to the highest index values, the degrees of freedom of the system need to be re-defined as follows:

Listing 3.3 – Redefinition of doF in main file

```
uDirichlet = computeProjectionFaces(@analyticalPoisson,infoFaces.
   extFaces_D,X,T,referenceElement);

dofDirichlet= (nOfInteriorFaces+nOfExteriorFaces_N+nOfExteriorFaces_R)*
   nOfFaceNodes + (1:nOfExteriorFaces_D*nOfFaceNodes);

dofUnknown = 1:(nOfInteriorFaces*nOfFaceNodes+nOfExteriorFaces_N*
   nOfFaceNodes+nOfExteriorFaces_R*nOfFaceNodes);
```

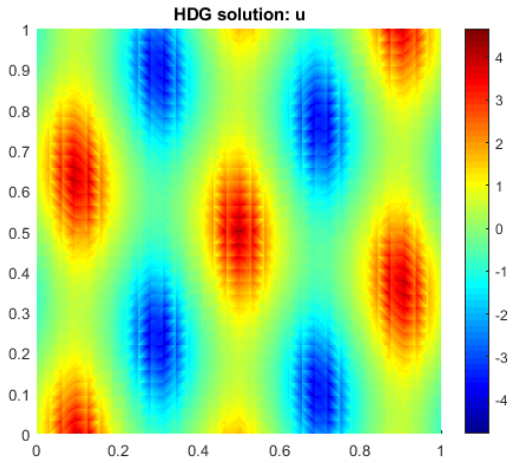
Finally, codes for the computations of matrices involving the Neumann and Robin boundaries were written in function `hdgMatrixPoisson.m`. As it can be noticed, the matrices are only computed if the element has faces on the Neumann or Robin boundaries and the matrix elements are allocated accordingly.

Listing 3.4 – Coding for the additional matrices in `hdgMatrixPoisson.m`

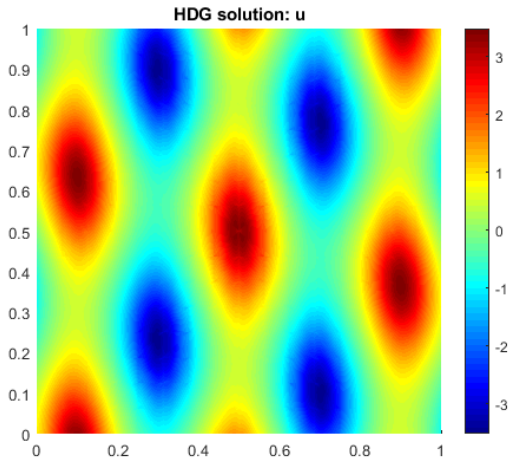
```
%Robin matrix (All_R)
if Fext_R == 1
    nodes = faceNodes(face_R_id,:); Xf = Xe(nodes,:);
    dxdxi = Nx1d*Xf(:,1); dydxi = Nx1d*Xf(:,2);
    dxdxiNorm = sqrt(dxdxi.^2+dydxi.^2); dline = dxdxiNorm.*IPw_f';
    ind_face = (face_R_id-1)*nOfFaceNodes + (1:nOfFaceNodes);
    Auu_f = N1d'*(spdiags(dline,0,ngf,ngf)*N1d)*gamma;
    Arr(ind_face,ind_face) = -Auu_f;
end

%Neumann force vector
fqN = zeros(nOfFaces*nOfFaceNodes,1);
if Fext_N == 1
    nodes_N = faceNodes(face_N_id,:);
    Xf_N = Xe(nodes_N,:);
    dxdxi = Nx1d*Xf_N(:,1); dydxi = Nx1d*Xf_N(:,2);
    dxdxiNorm = sqrt(dxdxi.^2+dydxi.^2); dline = dxdxiNorm.*IPw_f';
    aux_f = -N1d'*(spdiags(dline,0,ngf,ngf)*neumanPoisson(N1d*Xf_N));
    if face_N_id == 1
        fqN(1:nodes_of_face) = aux_f;
    elseif face_N_id == 2
        fqN(nodes_of_face+1:2*nodes_of_face) = aux_f;
    else
        fqN(2*nodes_of_face+1:3*nodes_of_face) = aux_f;
    end
end

%Robin force vector
fqR = zeros(nOfFaces*nOfFaceNodes,1);
if Fext_R == 1
    nodes_R = faceNodes(face_R_id,:);
    Xf_R = Xe(nodes_R,:);
    dxdxi = Nx1d*Xf_R(:,1); dydxi = Nx1d*Xf_R(:,2);
    dxdxiNorm = sqrt(dxdxi.^2+dydxi.^2); dline = dxdxiNorm.*IPw_f';
    aux_f = -N1d'*(spdiags(dline,0,ngf,ngf)*robinPoisson(N1d*Xf_R));
    if face_R_id == 1
        fqR(1:nodes_of_face) = aux_f;
    elseif face_R_id == 2
        fqR(nodes_of_face+1:2*nodes_of_face) = aux_f;
    else
        fqR(2*nodes_of_face+1:3*nodes_of_face) = aux_f;
    end
end
```

4 Results and discussion

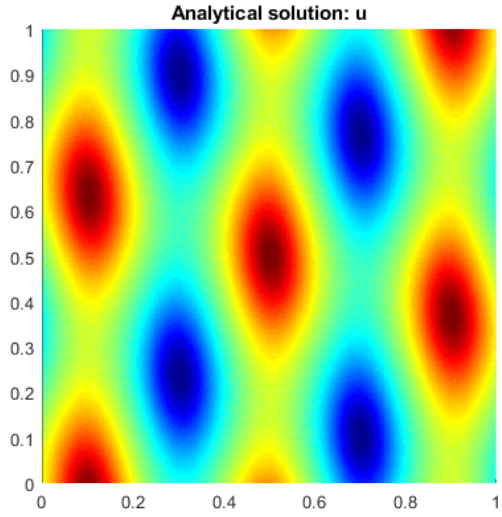
Examples of the solutions obtained using linear and cubic approximation with different meshes are depicted in Figure (4.1).



(a) Linear interpolation and 2048 elements



(b) Cubic interpolation and 512 elements



(c) Analytical solution

Figure 4.1 – Comparison of numerical results using different degrees of interpolation and meshes with the analytical solution

Although a finer mesh was used for the linear interpolation, the associated computer error was higher ($e = 1.718371 \cdot 10^{-1}$ in $\mathcal{L}_2(\Omega)$) in comparison with the one of the cubic interpolation ($e = 2.179106 \cdot 10^{-2}$ in $\mathcal{L}_2(\Omega)$).

Figure (4.2) shows the numerical solution computed on an equally-refined mesh with a degree of approximation $p = 2$ and $p = 4$. It is worth noting how the accuracy becomes higher as the degree of the approximation increases resulting in a error reduction of an order of magnitude.

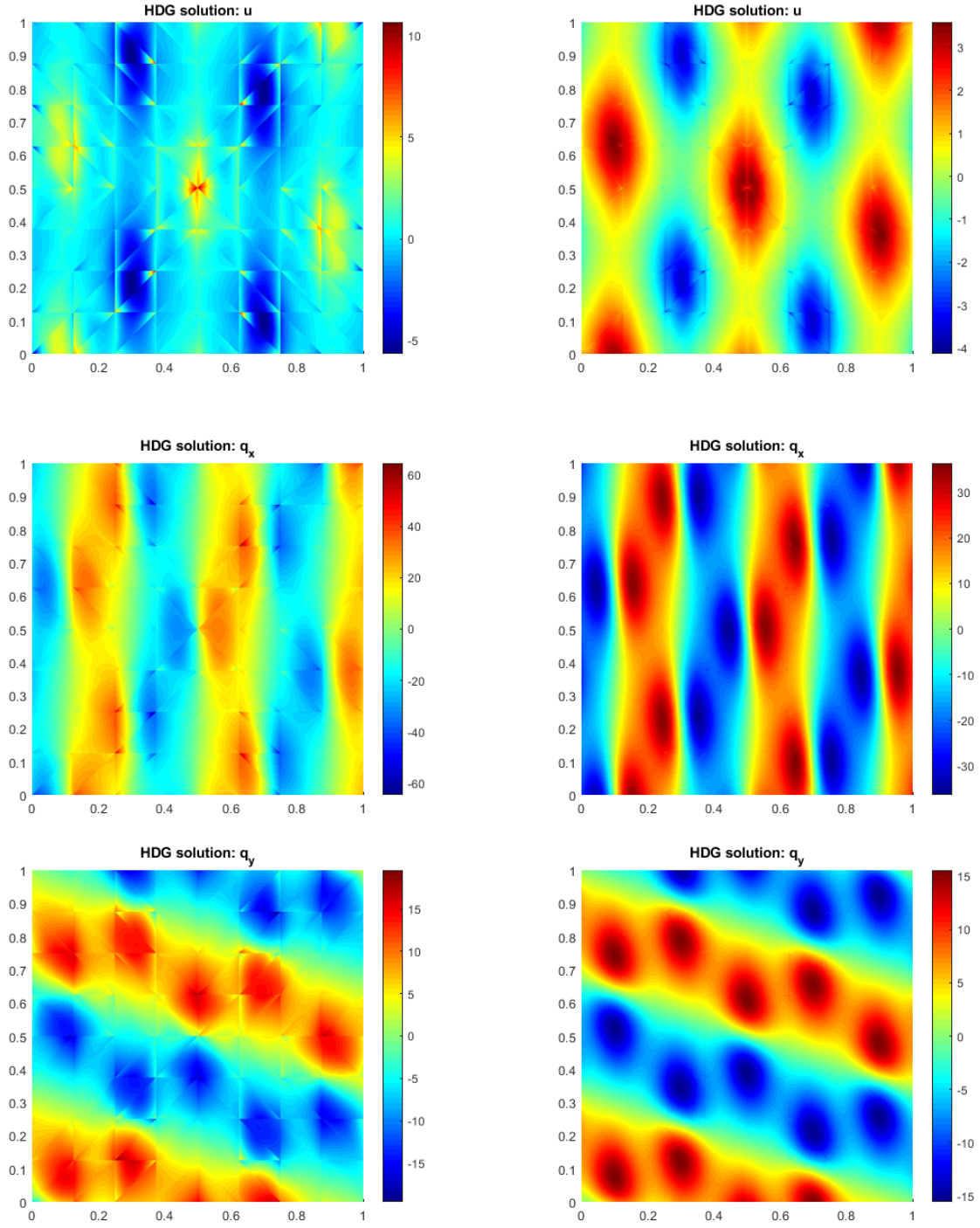


Figure 4.2 – Model problem solution for $p = 2$ (left) and $p = 4$ (right) on mesh with 128 elements

Using the implementation given for the post-processing technique, Figures (4.3) and (4.4) show a comparison of the original and post-processed solutions, where the gain in accuracy induced by the post-processing is clearly observed for both coarse and refined meshes. For the 4th-order interpolation in Figure (4.3), the solution u_*^h has an error in the $\mathcal{L}_2(\Omega)$ norm of $1.811941 \cdot 10^{-1}$ whereas, by post-processing, this the value decreases to $2.359702 \cdot 10^{-2}$. Similarly, in the case of the linear interpolation in Figure (4.4), the error reduction is from $6.316016 \cdot 10^{-1}$ to $1.375870 \cdot 10^{-2}$. It is important to stress that the additional accuracy gained by the post-processing requires only the solution of a element-by-element problem, having a comparatively insignificant associated cost.

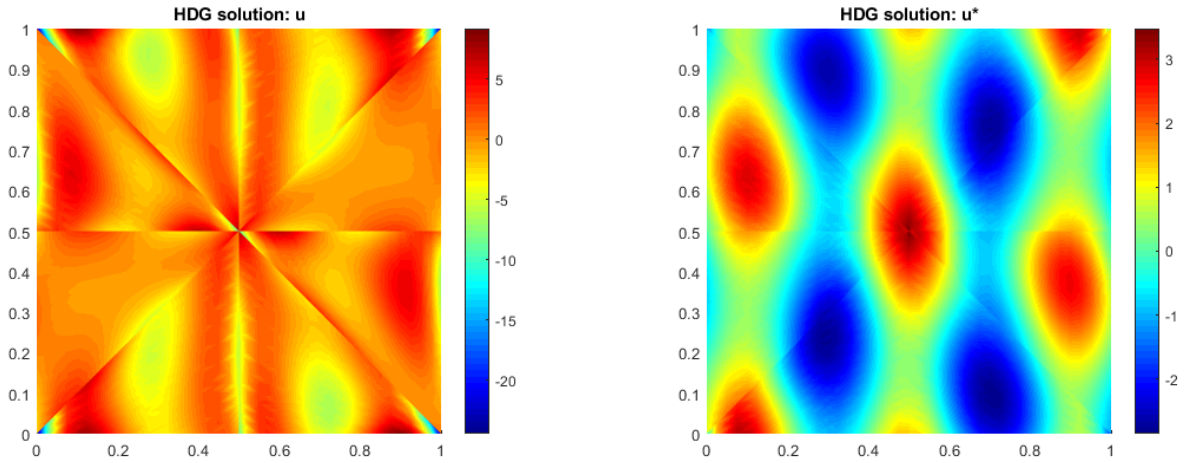


Figure 4.3 – Effect of post-processing procedure for solution using 8 elements with $p = 4$

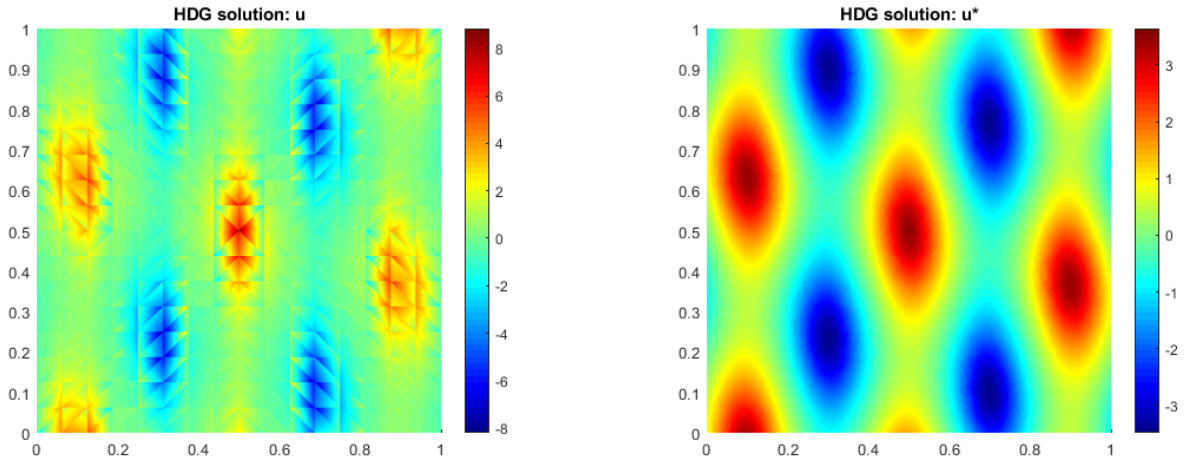


Figure 4.4 – Effect of post-processing procedure for solution using 512 elements with $p = 1$

Finally, an h-convergence study of the error of solution and the post-processed solution is performed. Figure (4.5) shows the results of the convergence study where the difference between the optimal rate of convergence of the solution $(p+1)$ and post-processed $(p+2)$ is clearly illustrated. As predicted by the error estimator, a similar rate of convergence is observed for the post-processed solution u_*^h that results from a computation with degree of approximation p and the solution u^h computed with a degree of approximation $p+1$. Nevertheless, even though the rate of convergence might be similar, the post-processed solutions of degree p are more accurate in comparison with the solution with a $p+1$ approximation.

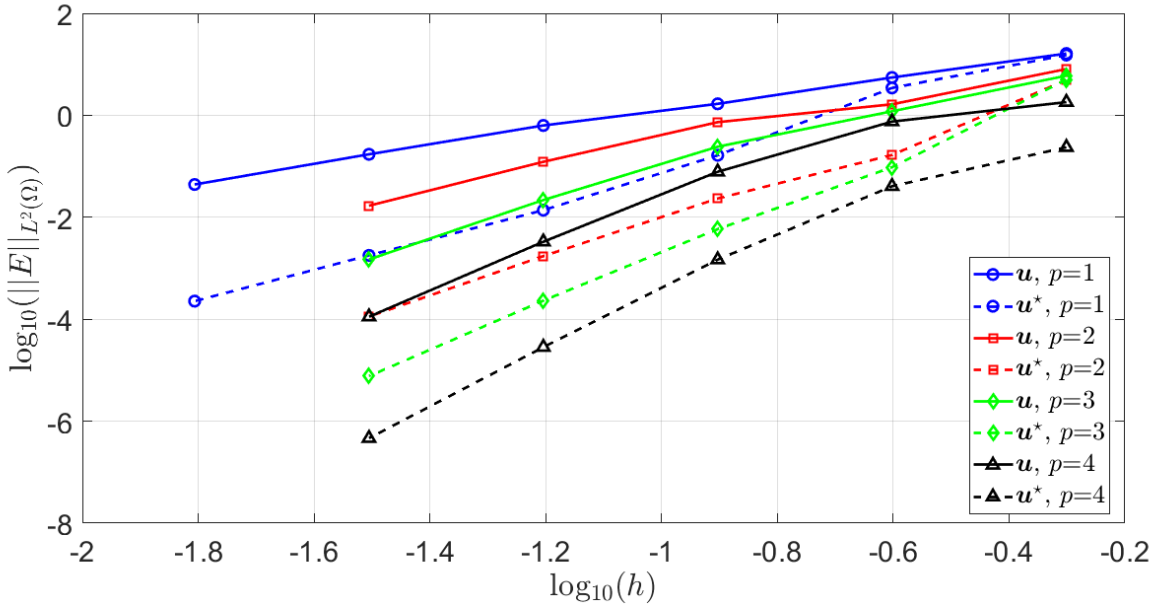


Figure 4.5 – Error of the solution and the post-processed solution in the $\mathcal{L}_2(\Omega)$ norm as a function of the characteristic element size h for different values of the approximation degree p

Bibliography

[1] Ruben Sevilla. *Hybridisable discontinuous Galerkin for second-order elliptic problems. Notes for DG Summer School - Barcelona*. 2017.

[2] Ruben Sevilla and Antonio Huerta. "Tutorial on Hybridizable Discontinuous Galerkin (HDG) for Second-Order Elliptic Problems". In: May 2016, pp. 105–129. ISBN: 978-3-319-31923-0. DOI: 10.1007/978-3-319-31925-4_5.