HDG assignment #3

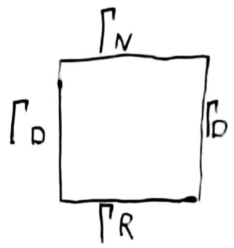
Name: Oriol Falip Garcia
Date: 05/06/2019

Problem Statement

Consider the domain $\Omega = [0,1]^2$ such that $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_R$
and $\Gamma_N \cap \Gamma_R = \phi$, $\Gamma_D \cap \Gamma_N = \phi$ and $\Gamma_D \cap \Gamma_N = \phi$

where:

$$\left.\begin{aligned} \Gamma_N &\equiv \{(x,y) \in \mathbb{R}^2 : y = 1\} \\ \Gamma_R &\equiv \{(x,y) \in \mathbb{R}^2 : y = 0\} \\ \Gamma_D &\equiv \partial\Omega \setminus (\Gamma_N \cup \Gamma_R) \end{aligned}\right] \rightarrow$$



The following differential equation is defined:

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

We can split our domain in n_{el} elements such that

$$\Omega = \bigcup_{i=1}^{n_{el}} \Omega_i \quad \text{and} \quad \Omega_i \cap \Omega_j = \phi$$

which define internal boundaries (interface) Γ:

$$\Gamma \equiv \left[\bigcup_{i=1}^{n_{el}} \partial\Omega_i\right] \setminus \partial\Omega$$

CS Scanned with CamScanner

with that splitting an equivalent strong form can be written

$$
\begin{cases}
-\nabla\cdot(k\nabla u) = s & \text{in } \Omega_i \quad i=1,\dots,n_{el} \\
u = u_D & \text{on } \Gamma_D \\
\hat{n}\cdot(k\nabla u) = t & \text{on } \Gamma_N \\
\hat{n}\cdot(k\nabla u) + \gamma u = g & \text{on } \Gamma_R \\
[\![u\hat{n}]\!] = \vec{0} & \text{on } \Gamma \\
[\![\hat{n}\cdot\nabla u]\!] = 0 & \text{on } \Gamma
\end{cases}
$$

If we define the flux as $\vec{q} = -k\nabla u$ then we can write:

$$
\begin{cases}
\vec{q} + k\nabla u = \vec{0} & \text{in } \Omega_i \quad i=1,\dots,n_{el} \\
\nabla\cdot\vec{q} = s & \text{in } \Omega_i \quad i=1,\dots,n_{el} \\
u = u_D & \text{on } \Gamma_D \\
\hat{n}\cdot\vec{q} = -t & \text{on } \Gamma_N \\
-\hat{n}\cdot\vec{q} + \gamma u = g & \text{on } \Gamma_R \\
[\![u\hat{n}]\!] = \vec{0} & \text{on } \Gamma \\
[\![\hat{n}\cdot\vec{q}]\!] = 0 & \text{on } \Gamma
\end{cases}
$$

with that we can write the Hybridizable Discontinuous Galerking method as:

Scanned with CamScanner

HDG (strong)

Local problem (strong)

$$\begin{cases} \nabla \vec{q}_i = s & \text{in } \Omega_i \\ \vec{q}_i + k \nabla u_i = \vec{0} & \text{in } \Omega_i \\ u = u_0 & \text{on } \partial\Omega_i \cap \Gamma_D \\ u = \hat{u} & \text{on } \partial\Omega_i \setminus \Gamma_D \end{cases}$$

Global problem (strong)

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

→ it can be ignored since it is being fulfilled in the Local problem.

To write the weak form it must be noticed that the Local problem (strong) remains the same if we have Neumann or Robin B.C. So the weak form will be the same as the one in the Tutorial on Hybridizable Discontinuous Galerkin (HDG) for Second order Elliptic Problems

Scanned with CamScanner

So

HDG (weak)

Local problem (weak)

$$\begin{cases} -(\nabla v, \vec{q}_i)_{\Omega_i} + \langle v, \hat{n}_i \cdot \vec{q}_i \rangle_{\partial\Omega_i} = (v, f)_{\Omega_i} \\ -(\vec{w}, \vec{q}_i)_{\Omega_i} + (\nabla \cdot \vec{w}, u_i)_{\Omega_i} = \langle \hat{n}_i \cdot \vec{w}, u_D \rangle_{\partial\Omega_i \cap \Gamma_D} + \langle \hat{n}_i \cdot \vec{w}, \hat{u} \rangle_{\partial\Omega_i \setminus \Gamma_D} \end{cases}$$

Global problem (weak)

At this point we must notice that we have an extra equation coming from Robin Boundary Conditions. With that the Global problem in weak form reads

$$\begin{cases} \sum_{i=1}^{n_{el}} \left[\langle \mu, \hat{n}_i \cdot \vec{q}_i \rangle_{\partial\Omega_i \setminus \partial\Omega} + \langle \mu, \hat{n}_i \cdot \vec{q} \rangle_{\partial\Omega_i \cap \Gamma_N} \right] = \sum_{i=1}^{n_{el}} \langle \mu, t \rangle_{\partial\Omega_i \cap \Gamma_N} \\ \sum_{i=1}^{n_{el}} \left[-\langle \nu, \hat{n}_i \cdot \vec{q}_i \rangle_{\partial\Omega_i \setminus \partial\Omega} + \langle \nu, \gamma \hat{u} \rangle_{\partial\Omega_i \setminus \partial\Omega} - \langle \nu, \hat{n}_i \cdot \vec{q}_i \rangle_{\partial\Omega_i \cap \Gamma_R} + \langle \nu, \gamma \hat{u} \rangle_{\partial\Omega_i \cap \Gamma_R} \right] = \\ \qquad = \sum_{i=1}^{n_{el}} \langle \nu, g \rangle_{\partial\Omega_i \cap \Gamma_R} \end{cases}$$

Scanned with CamScanner

Problem 3

considering the analytical solution $u(x,y) = \exp(ax - ky) - \sin(\gamma\pi x) - b\pi y)$

with $\begin{bmatrix} a = 0.1 & k = 1.2 \\ b = 6 & \gamma = 3 \end{bmatrix}$

we can obtain the analytical source term as:

$$-\nabla\cdot(K\nabla u) = S = -k\left[a^2 \exp(ax-ky) + \gamma^2\pi^2 \sin(\gamma\pi x - b\pi y) + \right.$$
$$\left. + k^2 \exp(ax-ky) + b^2\pi^2 \sin(\gamma\pi x - b\pi y)\right]$$

traction forces (or fluxes) can be obtained via $t = \hat{n}\cdot(K\nabla u)$ on Γ_N

since our Neumann B.C are on the top face $\hat{n} = (0,1)$ then

$$\boxed{t = K\left[-k \exp(ax-ky) + b\pi \cos(\gamma\pi x - b\pi y)\right]}$$

Regarding g: $g = \hat{n}\cdot(K\nabla u) + \gamma u$ with $\hat{n} = (0,-1)$ then

$$\boxed{g = K\left[k \exp(ax-ky) - b\pi \cos(\gamma\pi x - b\pi y)\right] + \gamma\left[\exp(ax-ky) - \sin(\gamma\pi x - b\pi y)\right]}$$

Regarding u_D we have two faces defined by $x=0$ and $x=1$ then

Face 1: $u_D = \exp(-ky) - \sin(-b\pi y)$

Face 2: $u_D = \exp(a - ky) - \sin(\gamma\pi - b\pi y)$

CS Scanned with CamScanner

Results

Considering that particular problem, in the following section the results will be analysed in order to ensure that the implementation of the method has been done properly.

Starting by analysing the behaviour of the solution with different meshes, let's consider a polynomial degree equal to on (p=1) and different meshes, each one one level more fine that the previous one. The following figure shows the meshes used:

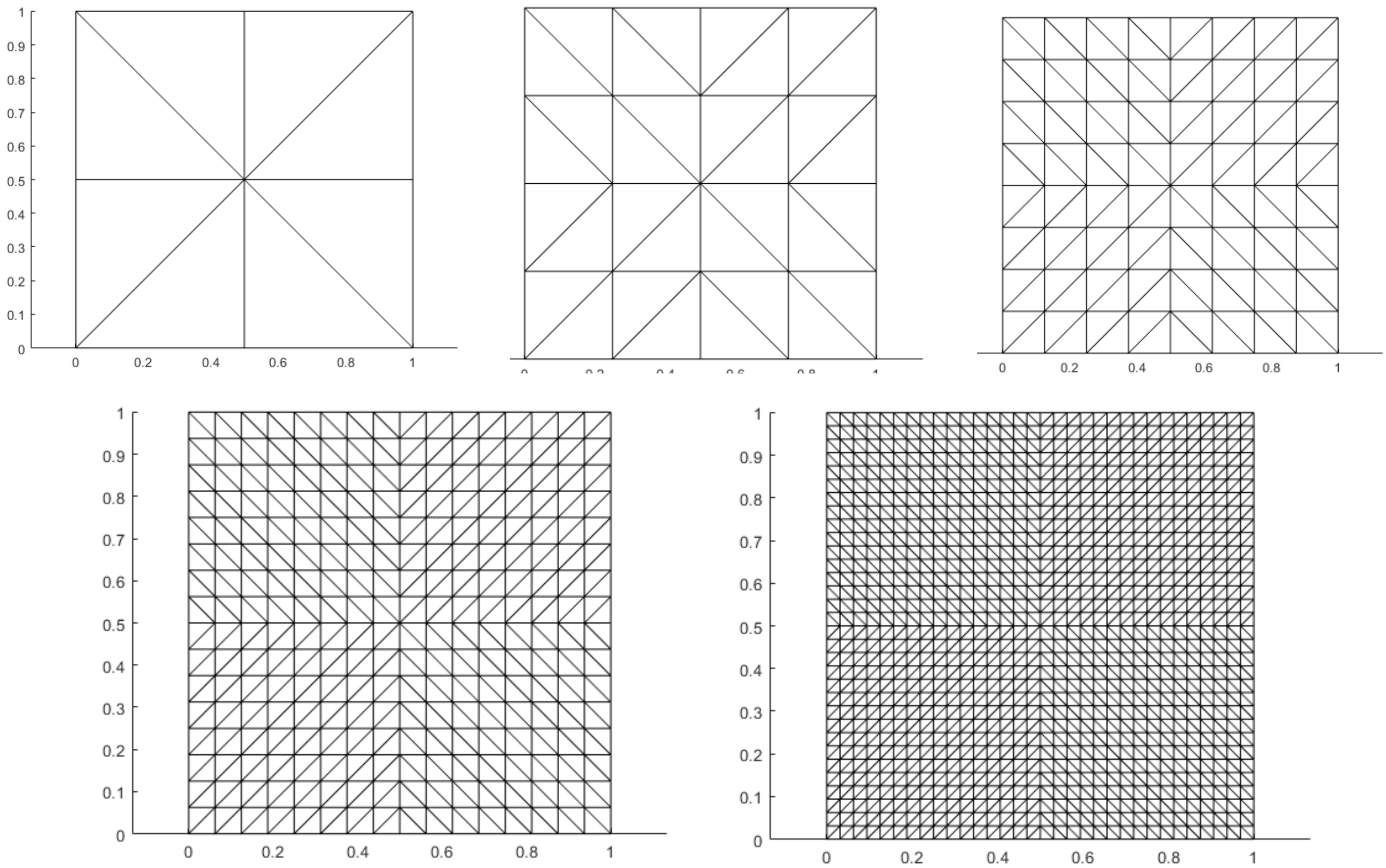


Figure 1. Discretization of the geometry for different mesh refinement.

Numbering the meshes from one to five starting from the coarser mesh, the following results have been obtained:

Mesh 1

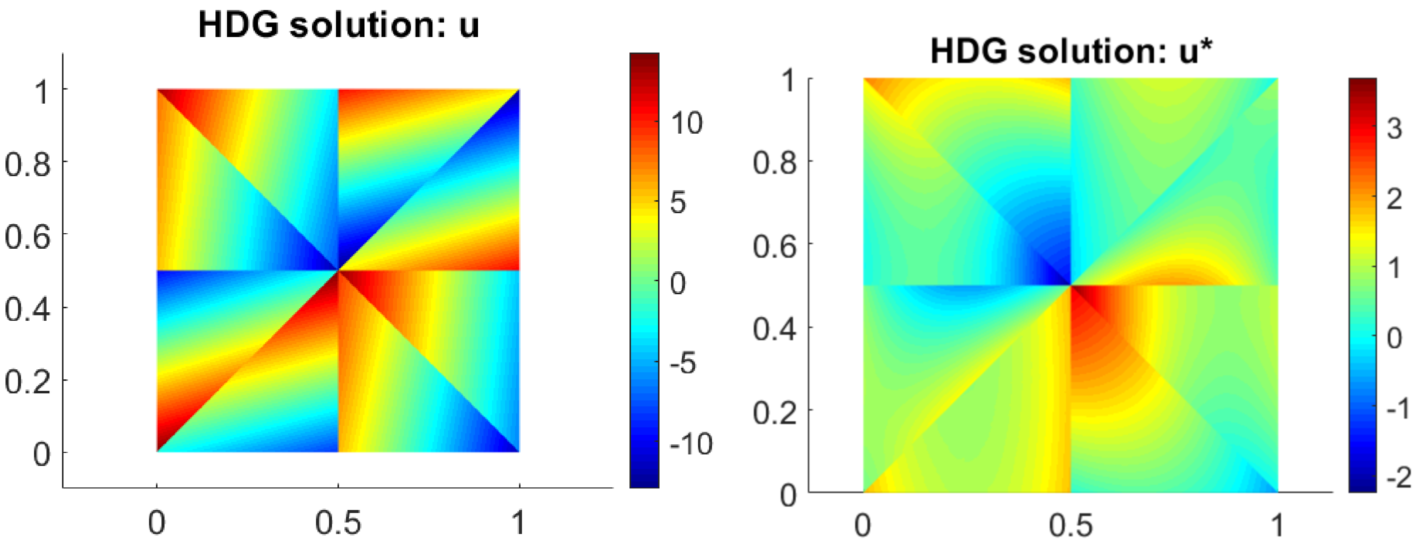


Figure 2. Solution of our problem 'u' and post-processed solution 'u*' for mesh 1 with polynomial degree p=1.

Mesh 2

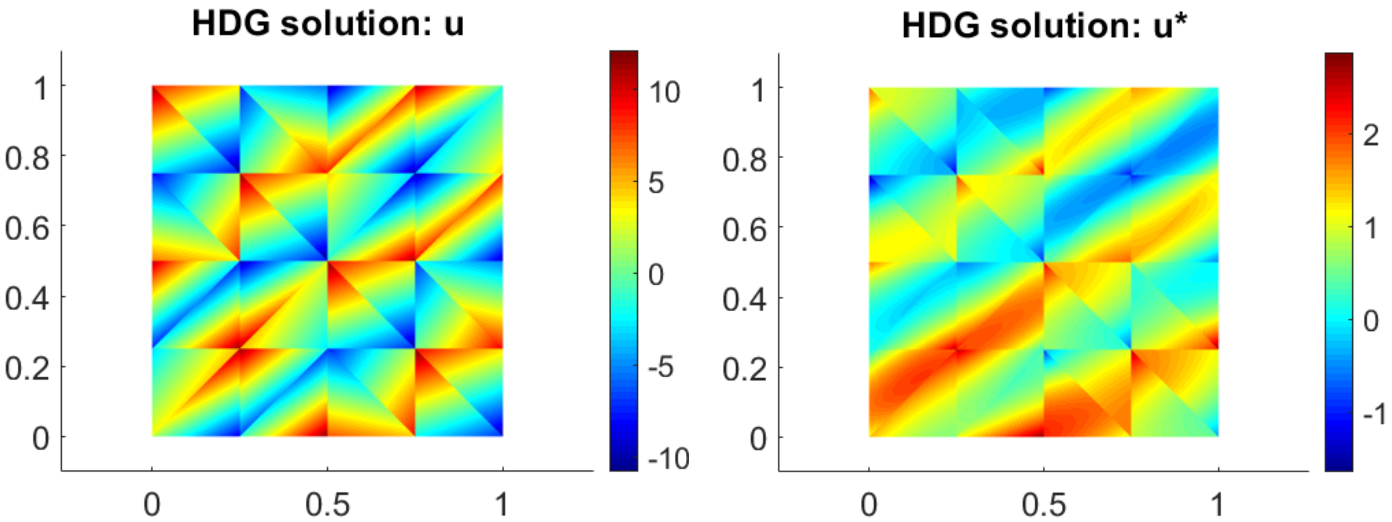


Figure 3. Solution of our problem 'u' and post-processed solution 'u*' for mesh 2 with polynomial degree p=1.

Mesh 3

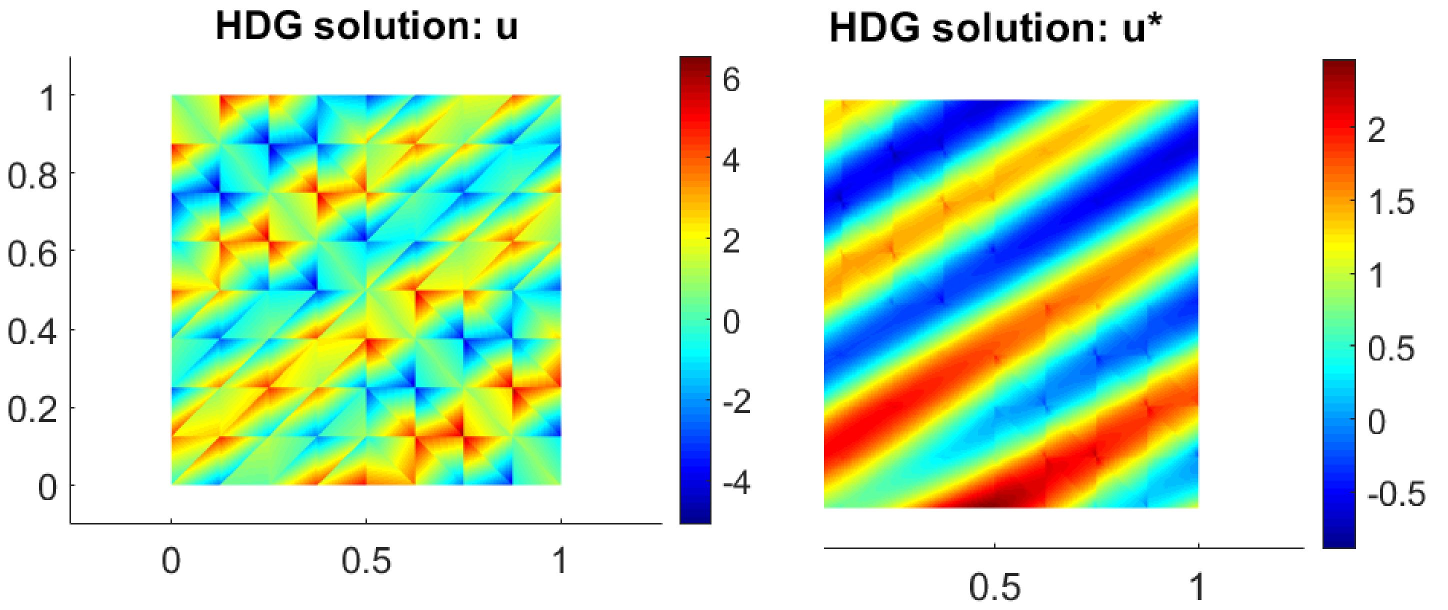


Figure 4. Solution of our problem 'u' and post-processed solution 'u*' for mesh 3 with polynomial degree p=1.

Mesh 4

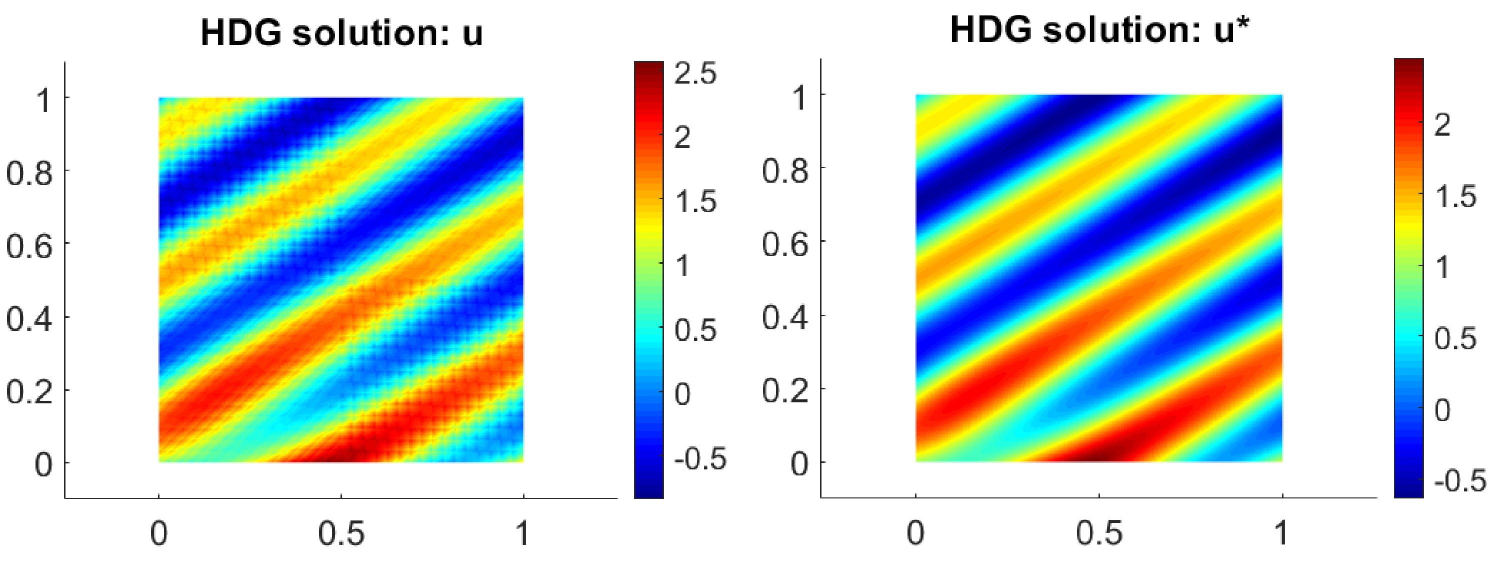


Figure 5. Solution of our problem 'u' and post-processed solution 'u*' for mesh 4 with polynomial degree p=1.

Mesh 5

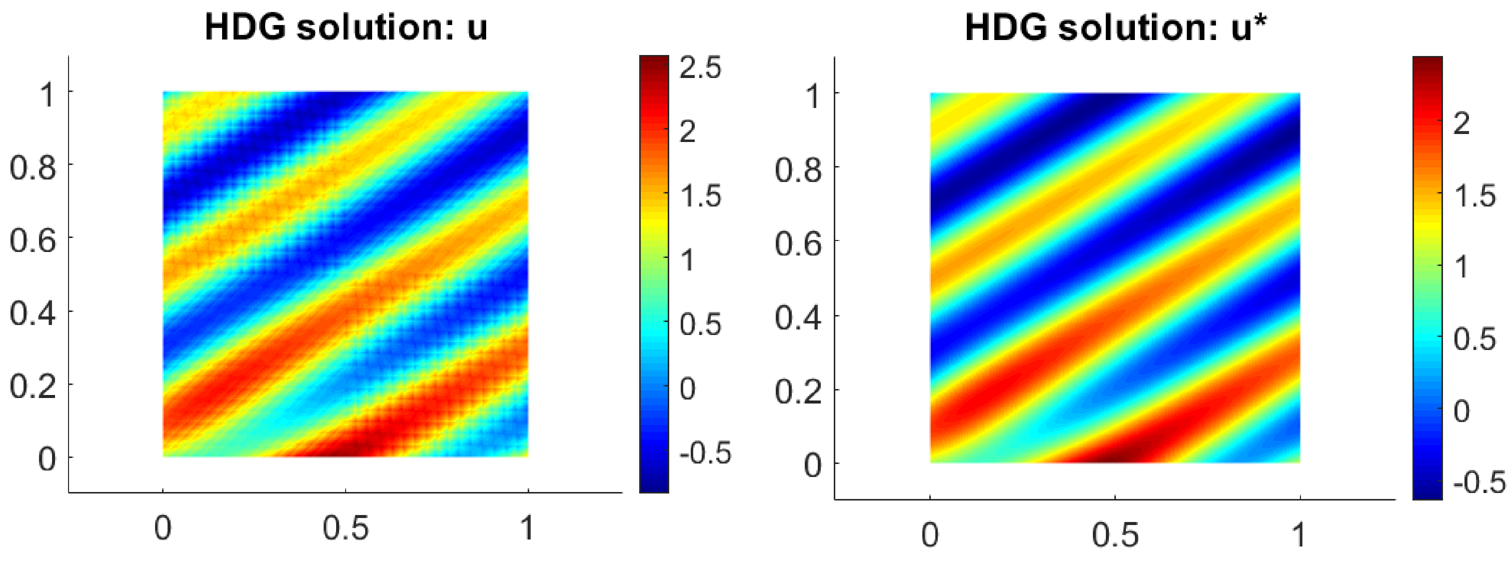


Figure 6. Solution of our problem 'u' and post-processed solution 'u*' for mesh 5 with polynomial degree p=1.

From all previous figures, it is clear that even when keeping the polynomial degree at just p=1, when refining the mesh the results improve quite drastically with just three of four levels of refinement. Although this result was expected, it is worth showing that since it is a good indicator that our implementation was done properly.

To show the effects of increasing the polynomial degree in HDG method, the following strategy is followed:

By keeping the mesh the same for all cases, Mesh 2 is chosen. From this, the polynomial degree is increasing from 1 to 4 to show the effects produced in the solution.

p=1

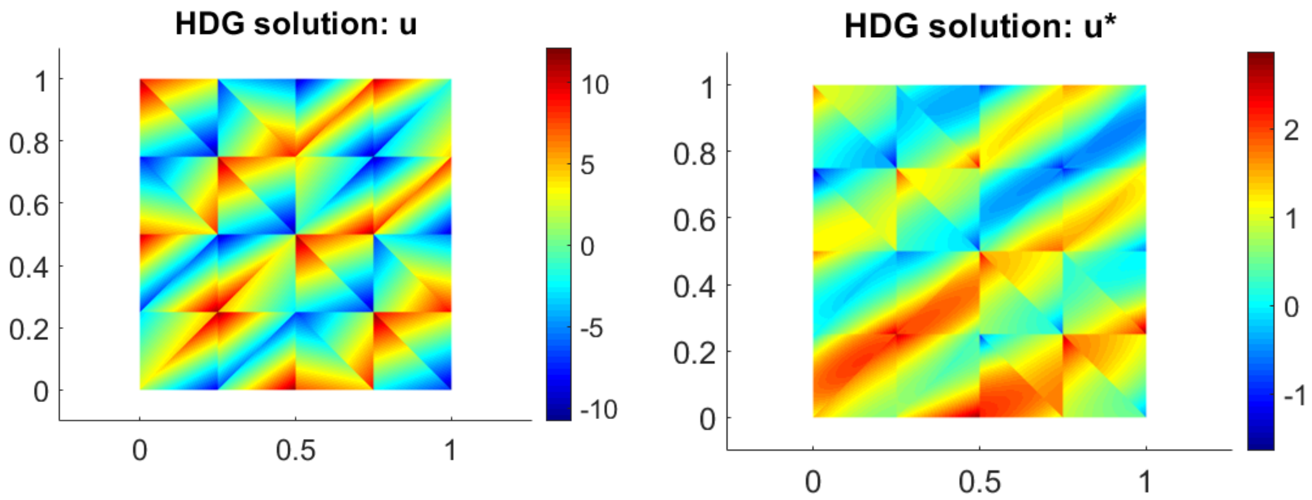


Figure 7. Solution of our problem ‘u’ and post-processed solution ‘u*’ for the Mesh 2 with polynomial degree p=1.

p=2

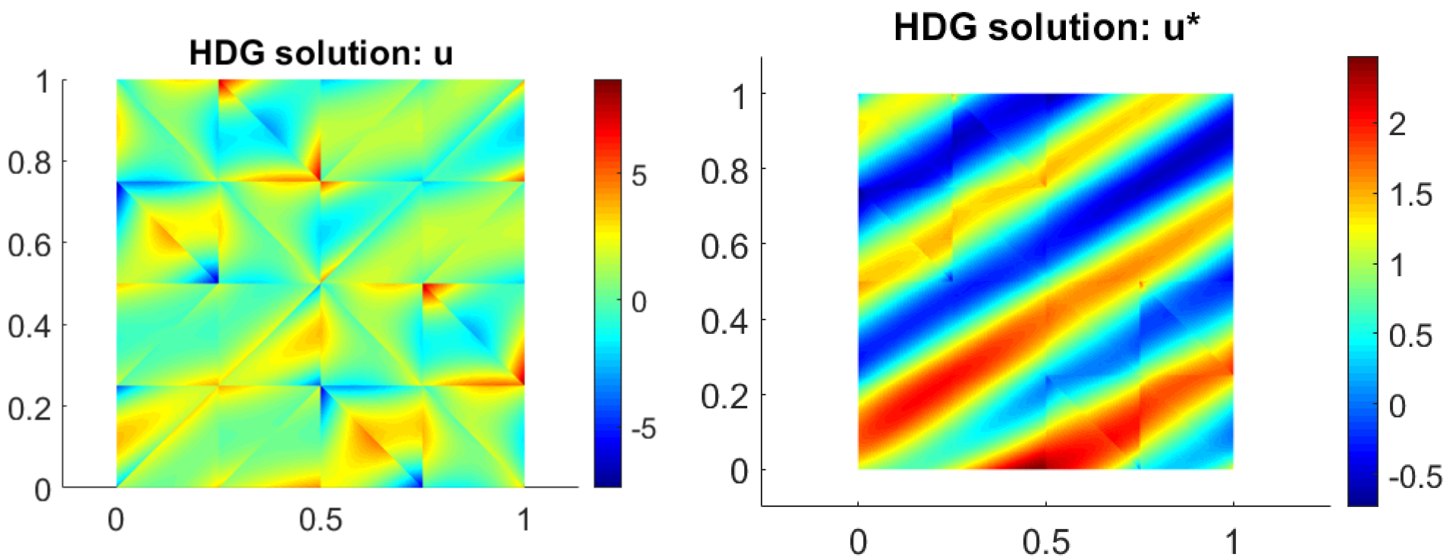


Figure 8. Solution of our problem ‘u’ and post-processed solution ‘u*’ for the Mesh 2 with polynomial degree p=2.

p=3

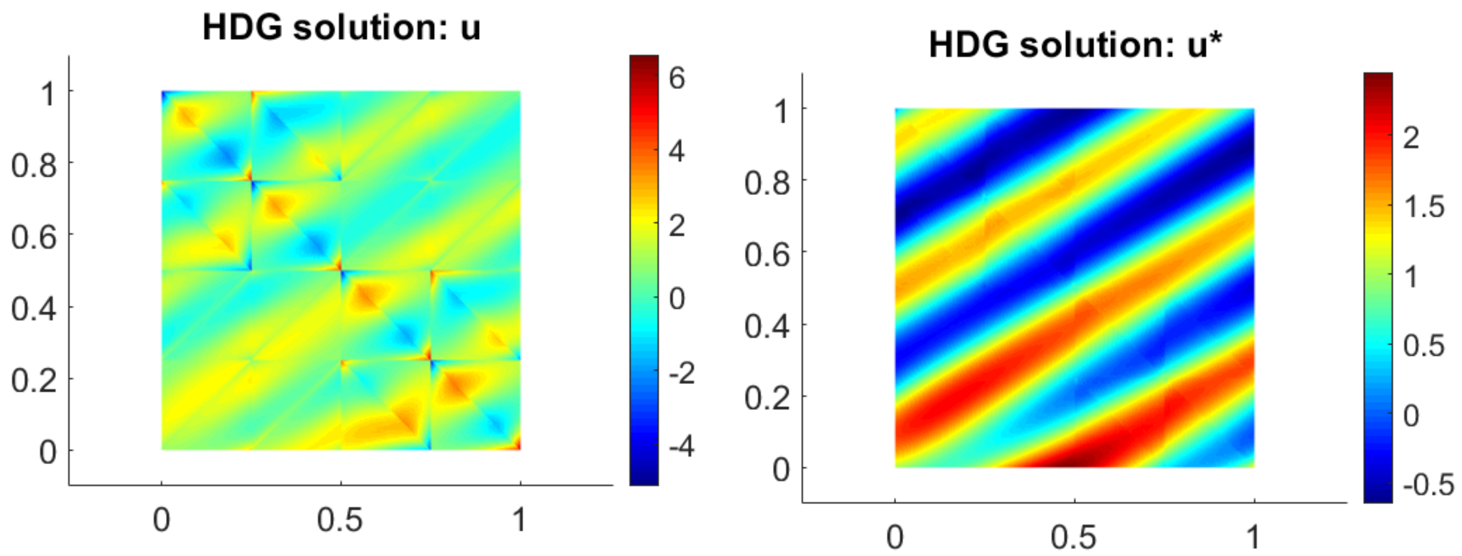


Figure 9. Solution of our problem ‘u’ and post-processed solution ‘u*’ for the Mesh 2 with polynomial degree p=3.

p=4

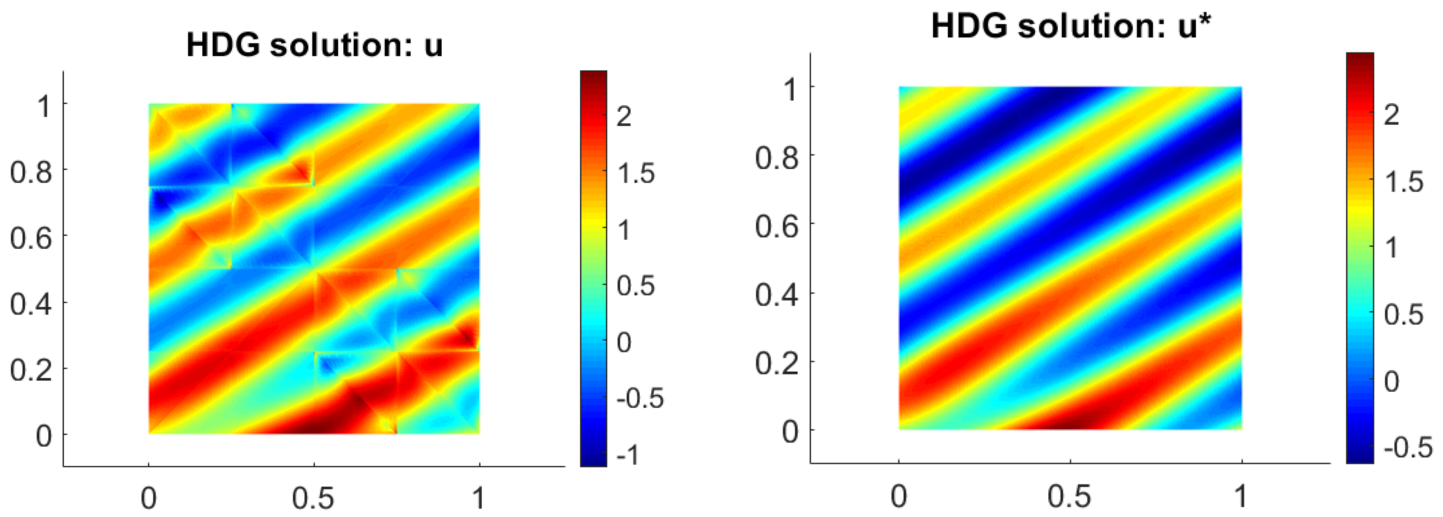


Figure 10. Solution of our problem ‘u’ and post-processed solution ‘u*’ for the Mesh 2 with polynomial degree p=4.

Again, the obvious and expected solution is that the solution quality increase when increasing the polynomial degree, achieving quite good results even for the solution and prost-processed solution.

Comparing all the previous results with the exact solution obtained from the analytical expression of the solution with WolframAlpha we clearly see that our solution is perfectly equivalent to the exact solution as seen in *Figure 11*.

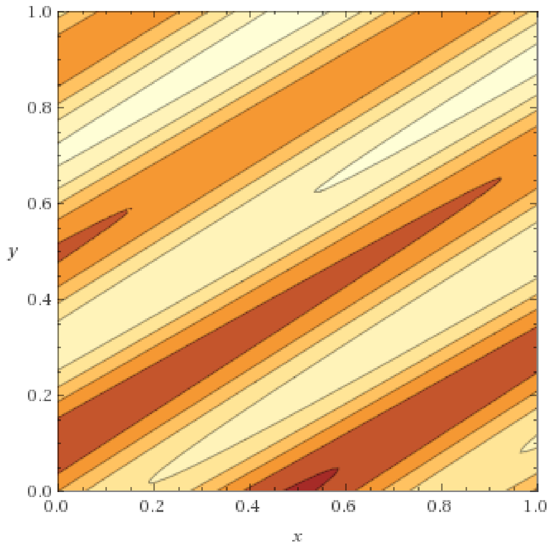


Figure 11. Exact solution from the analytical expression obtained with WolframAlpha.

To end this work, a convergence study is presented when increasing the polynomial degree by keeping the same mesh (Mesh 2). The magnitudes used to make the convergency study are the errors for u, **q** and u* in the L_2-norm defined in the domain Ω.

In the following table the errors obtained are presented:

	Error		
p	u	**q**	u*
1	3.1287	11.3127	0.3196
2	0.9785	11.5084	0.1860
3	0.5243	11.6609	0.1668
4	0.1894	11.6744	0.1640

Table 1. L_2 error for different magnitudes.

Plotting the errors in terms of the polynomial degree:

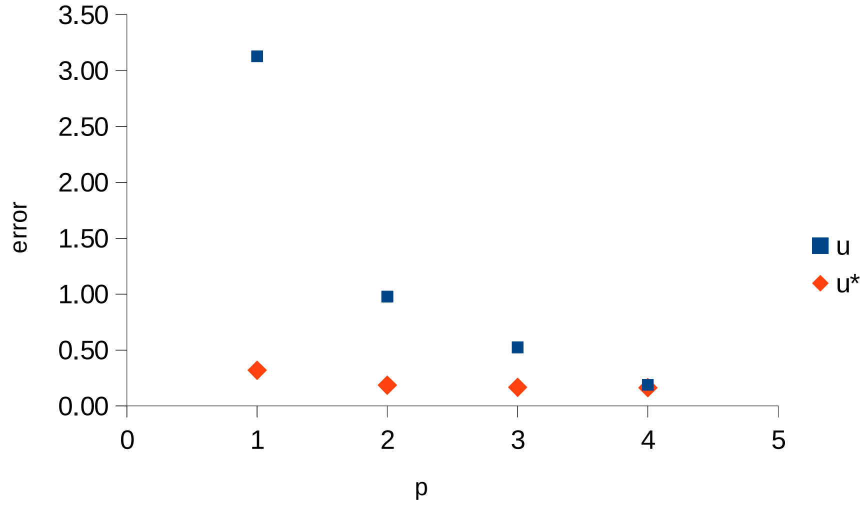


Figure 12. Error in terms of the polynomial degree for the solution and post-processed solution.

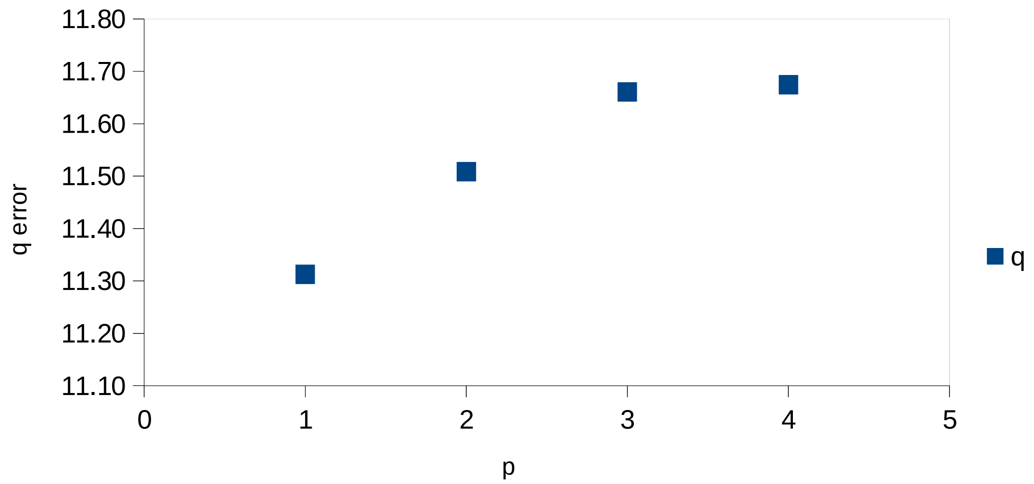


Figure 13. Error in terms of the polynomial degree for the flux.

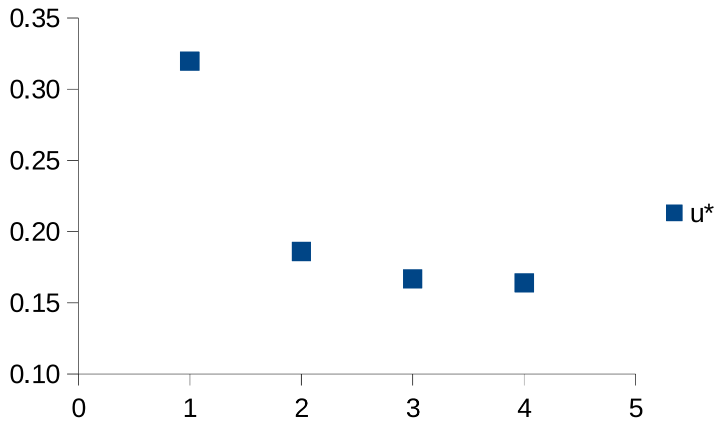


Figure 14. Error in terms of the polynomial degree for the flux.