16.930 PS4

Sachith Dunatunga

May 21, 2015

1 Q1

Surprisingly, the answer is provided in the reference material[1].

We will consider the first order linear convection-reaction equation to keep things simple. We first write the original DG method as

$$(\sigma u, w) - (\mathbf{c}u, \nabla w) + \langle \widehat{\mathbf{c}u} \cdot \mathbf{n}, w \rangle = (f, w) \tag{1}$$

where the numerical flux is an upwinded flux given by

$$\widehat{\mathbf{c}u} = \begin{cases} \mathbf{c}u_D & \text{if } F \in \Gamma_{in} \\ \mathbf{c}u^+ & \text{if } F \in \mathcal{F} \setminus \partial\Omega \text{ and } \mathbf{c} \cdot \mathbf{n} < 0 \\ \mathbf{c}u^- & \text{if } F \in \mathcal{F} \setminus \partial\Omega \text{ and } \mathbf{c} \cdot \mathbf{n} > 0 \\ \mathbf{c}u & \text{if } F \in \Gamma_{out} \end{cases} \tag{2}$$

For this same problem, the HDG method reduces to

$$(\sigma u, w) - (u, \mathbf{c} \cdot \nabla w) + \langle \hat{u}(\mathbf{c} \cdot \mathbf{n}) + \tau(u - \hat{u}), w \rangle = (f, w) \tag{3}$$

where

$$\hat{u} = \begin{cases} u_D & \text{if } F \in \Gamma_{in} \\ \frac{\tau^+ u^+ + \tau^- u^-}{\tau^+ + \tau^-} & \text{if } F \in \mathcal{F} \setminus \partial\Omega \\ u & \text{if } F \in \Gamma_{out}. \end{cases} \tag{4}$$

From inspection, these are equivalent if

$$\widehat{\mathbf{c}u} = \hat{u}(\mathbf{c} \cdot \mathbf{n} - \tau) + \tau u \tag{5}$$

for all the cases.

The choice of τ which reduces to the regular DG method is given by

$$\tau = \frac{1}{2}\left(|\mathbf{c} \cdot \mathbf{n}| + \mathbf{c} \cdot \mathbf{n}\right). \tag{6}$$

Plugging in for the outlet, by definition we have $\tau = \mathbf{c} \cdot \mathbf{n}$. Likewise, at the inlet we have $\tau = 0$. This directly satisfies those cases as equalities. I have run out of time, but performing the substitutions for the interior terms shows you exactly recover the original DG formulation.

2 Q2

I ran out of time and I was not able to prove this.

3 Code

The code is available at `http://github.com/neocpp89/16.930-ps4` and should be attached as well. Most of the changes are contained to the hdg subdirectory.

A couple of problems I am aware of:

- Inhomogenous Dirichlet BCs don't seem to work (value is set on the nodes, but it doesn't propagate inside the domain). I just removed the boundary portions of the matrix (assuming gd = 0) to allow the simulation to proceed.
- I originally followed the notes and the reference material[1], which use a formulation where $\mathbf{q} = \kappa \nabla u$ instead of $\mathbf{q} = \nabla u$ as requested. The code has been hastily converted since I only recently realized this may be an issue, but my primary test case uses $\kappa = 1$ so I may not have gotten all of it.
- Distmesh seems to get stuck and can't create the structured mesh at times (keeps flipping a line across a quad). It succeeds most of the time, but if it fails I have to interrupt and try again.

4 Convergence plots for Structured Mesh

In figure 1 we see the expected convergence behavior of order p+1 for u when selecting $\tau = 1$ or $1/h$. However, the rate drops to order p for $\tau = h$.

Interestingly, we see the expected convergence behaviors for q_x and q_y when selecting $\tau = 1$ or $\tau = h$, but the rate drops to order p for $\tau = 1/h$.

Since the postprocessing step relies on the order of $\mathbf{q}$ and u, it then makes sense that $\tau = 1$ results in a p+2 convergence rate for u^*. I was surprised that $\tau = h$ also had this p+2 rate (since u was order p), but I was not surprised that $\tau = 1/h$ would result in staying at order p+1 for u^*.

I did not have time to convert this to a proper table, but doing a linear fit of the last couple points confirms our code is working for the diffusion case. I went out to 40 elements in addition to the specified 8, 16, and 32 sizes (convergence was higher than I expected with just the 3 meshes, but it appears to have been a transient effect, since the rate went back down by adding the even finer mesh). The series below includes this mesh with n=40.

```
u
h - Order 1 has rate -1.02711.
h - Order 2 has rate -2.0551.
h - Order 3 has rate -3.08317.
1 - Order 1 has rate -2.04695.
1 - Order 2 has rate -3.07596.
1 - Order 3 has rate -4.10484.
1/h - Order 1 has rate -2.06656.
1/h - Order 2 has rate -3.13457.
1/h - Order 3 has rate -4.25809.
qx
h - Order 1 has rate -2.05846.
h - Order 2 has rate -3.08746.
h - Order 3 has rate -4.11365.
1 - Order 1 has rate -2.06695.
1 - Order 2 has rate -3.09276.
1 - Order 3 has rate -4.12155.
1/h - Order 1 has rate -1.03401.
1/h - Order 2 has rate -2.05983.
1/h - Order 3 has rate -3.08649.
qy
h - Order 1 has rate -2.05846.
h - Order 2 has rate -3.08746.
```

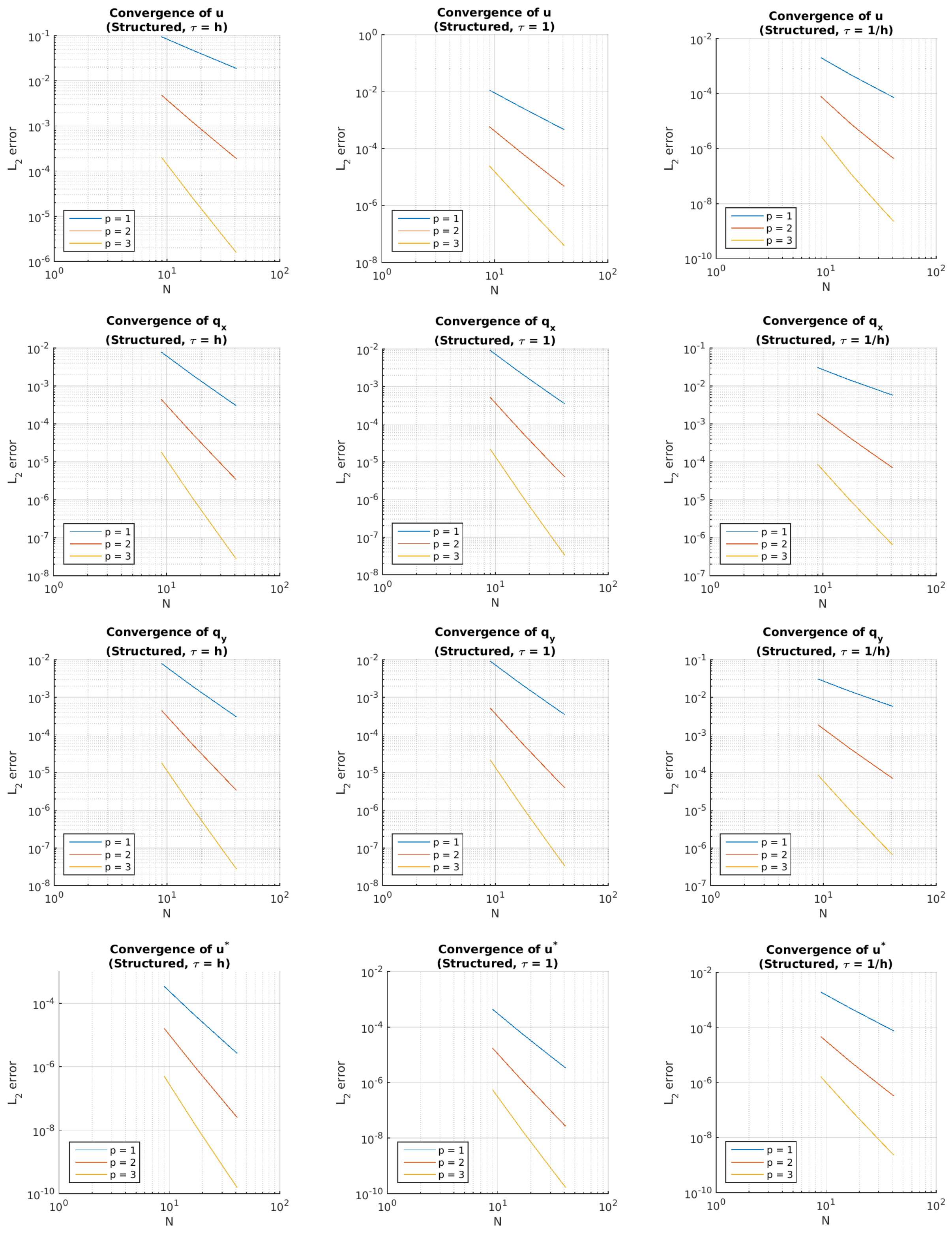

Figure 1: Convergence plots on the structured mesh. We have the stabilizations $\tau = h, 1, 1/h$ from left to right. The variables we compare are u, q_x, q_y, u^* from top to bottom. We only get good convergence for everything when $\tau = 1$.

```
h - Order 3 has rate -4.11365.
1 - Order 1 has rate -2.06695.
1 - Order 2 has rate -3.09276.
1 - Order 3 has rate -4.12156.
1/h - Order 1 has rate -1.03401.
1/h - Order 2 has rate -2.05983.
1/h - Order 3 has rate -3.08649.
u*
h - Order 1 has rate -3.08705.
h - Order 2 has rate -4.11205.
h - Order 3 has rate -5.14057.
1 - Order 1 has rate -3.08851.
1 - Order 2 has rate -4.11603.
1 - Order 3 has rate -5.1444.
1/h - Order 1 has rate -2.05619.
1/h - Order 2 has rate -3.09672.
1/h - Order 3 has rate -4.12233.
```

Table 1: Convergence rates for u, depending on order of elements and stabilization.

	p=1	p=2	p=3
h	1.02711	2.0551	3.08317
1	2.04695	3.07695	4.10484
1/h	2.06656	3.13457	4.25809

5 Unstructured Mesh

In figure 2 we see the meshes used. Since they are generated by distmesh, they may change from run to run.

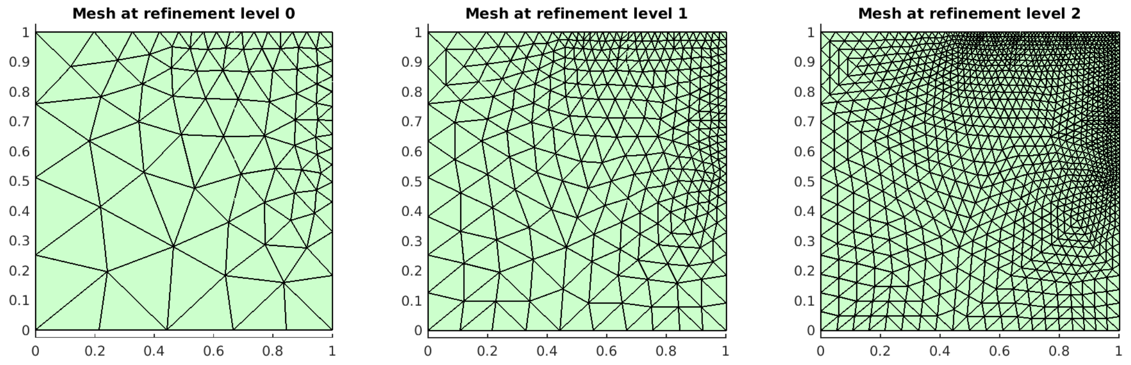
Mesh at refinement level 0
Mesh at refinement level 1
Mesh at refinement level 2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1

Figure 2: The mesh generated, and refinements thereof. Elements are more clustered towards boundaries where layers will begin to form. The order of elements in the mesh can be changed and is used for separate runs as well as postproceseing.

Before showing the results, we will explain our choice for parameter of τ. We followed the recommendation in the paper [1] and picked the centered scheme due to its simplicity. Specificially, it is single valued, with an expression given by

$$\tau = |\mathbf{c} \cdot \mathbf{n}| + \frac{\kappa}{0.2}. \tag{7}$$

The value of 0.2 is supposed to be a length-scale relating to diffusion, but we chose this essentially arbitrarily (a previous coarse mesh had this as a few element lengths).

Perhaps unsurprisingly, when diffusion dominates, the boundary layer is large and is easily captured even by the coarsest mesh. Purely relying on the visualization we can barely tell the difference between the postprocessed solutions (the unprocessed solutions are close but distinct), as shown in figure 3 and 4.

When convection becomes more dominant, we start to see the effects of a boundary layer in the solution (figures 5 and 6). While at $\kappa = 0.1$ ($\mathbf{c}$ unchanged) the boundary layer in the solution still looks reasonable, we are able to see more of a difference now between refinement levels and between element orders. However, once the postprocessing is applied, it again becomes difficult to discern many differences between the simulations.

In the convection-dominated case, we see a very clear boundary layer, which causes us to generate a poor solution when the mesh is coarse (see figures 7 and 8 and note the colorbar scale). Reducing the element size until the boundary layer fits results in dramatic improvement. We also noticed that the postprocessing does not seem to help as much in this case, although it does seem to make a small improvement. We suspect this is because the postprocessing step does not take into account any of the convection (only diffusion), and note that there is no dimensionless group relating the two in the postprocessing equations. This is okay when the convection is swamped out anyways by diffusion, but it is probably not as effective in a convection-dominated environment.

References

[1] N.C. Nguyen, J. Peraire, B. Cockburn *An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations* Journal of Computational Physics, 228 (2009) 3232-3254

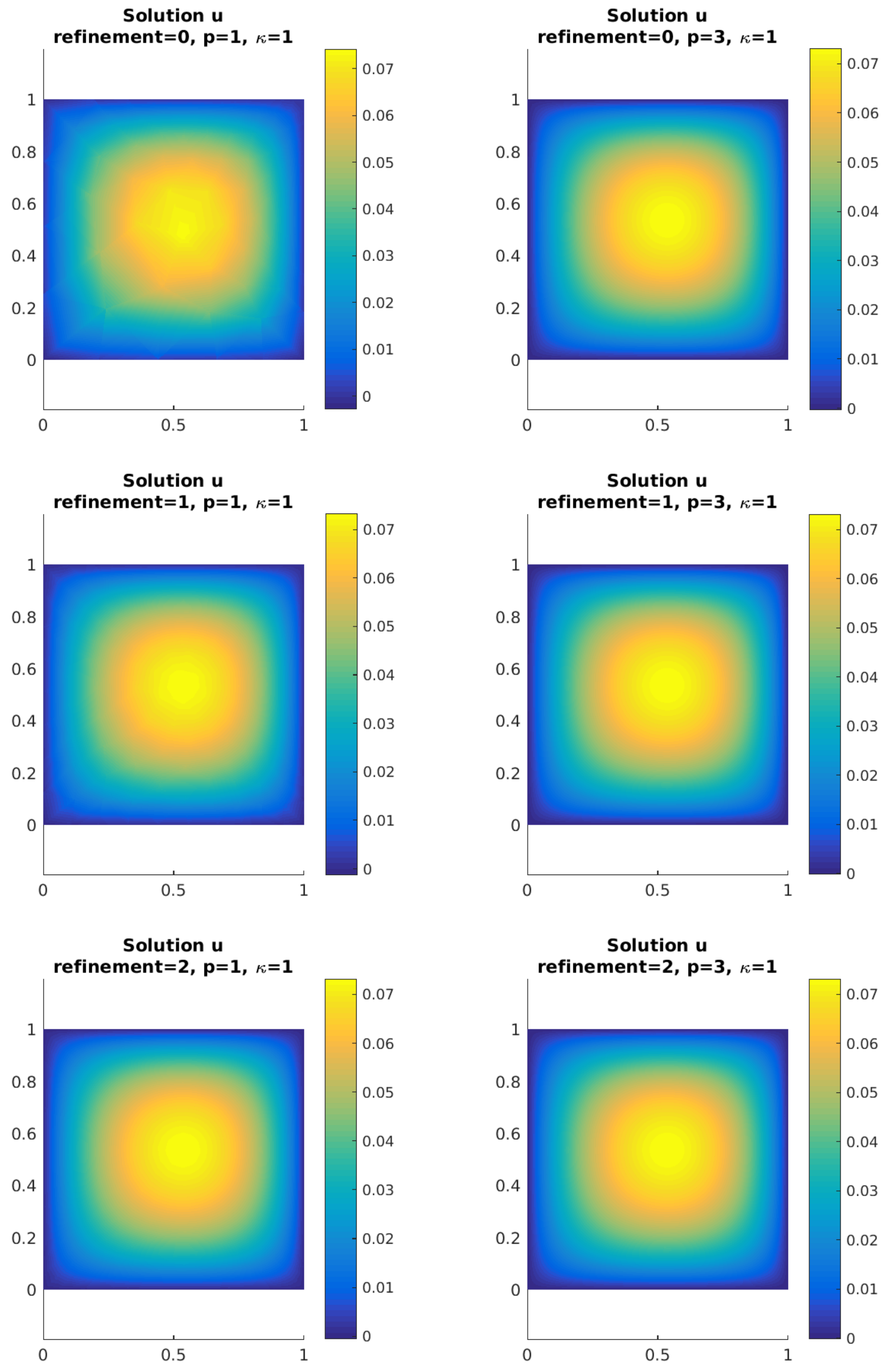

Figure 3: The diffusion-dominated case with $\kappa = 1$. Solutions from the p=1 elements are shown on the left while the p=3 elements are on the right.

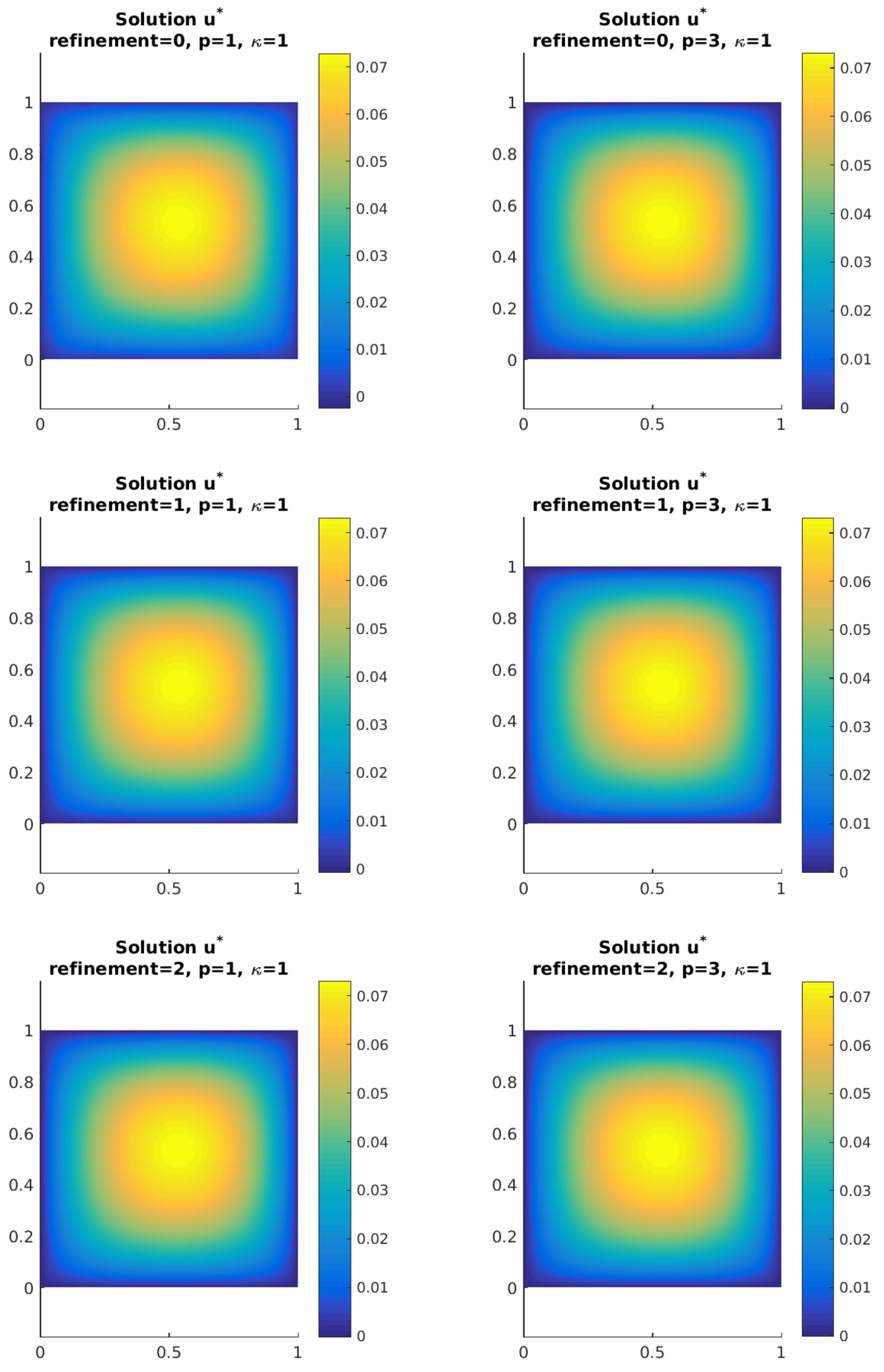

Figure 4: The diffusion-dominated case with $\kappa = 1$. Postprocessed solutions from the p=1 elements are shown on the left while the p=3 elements are on the right.

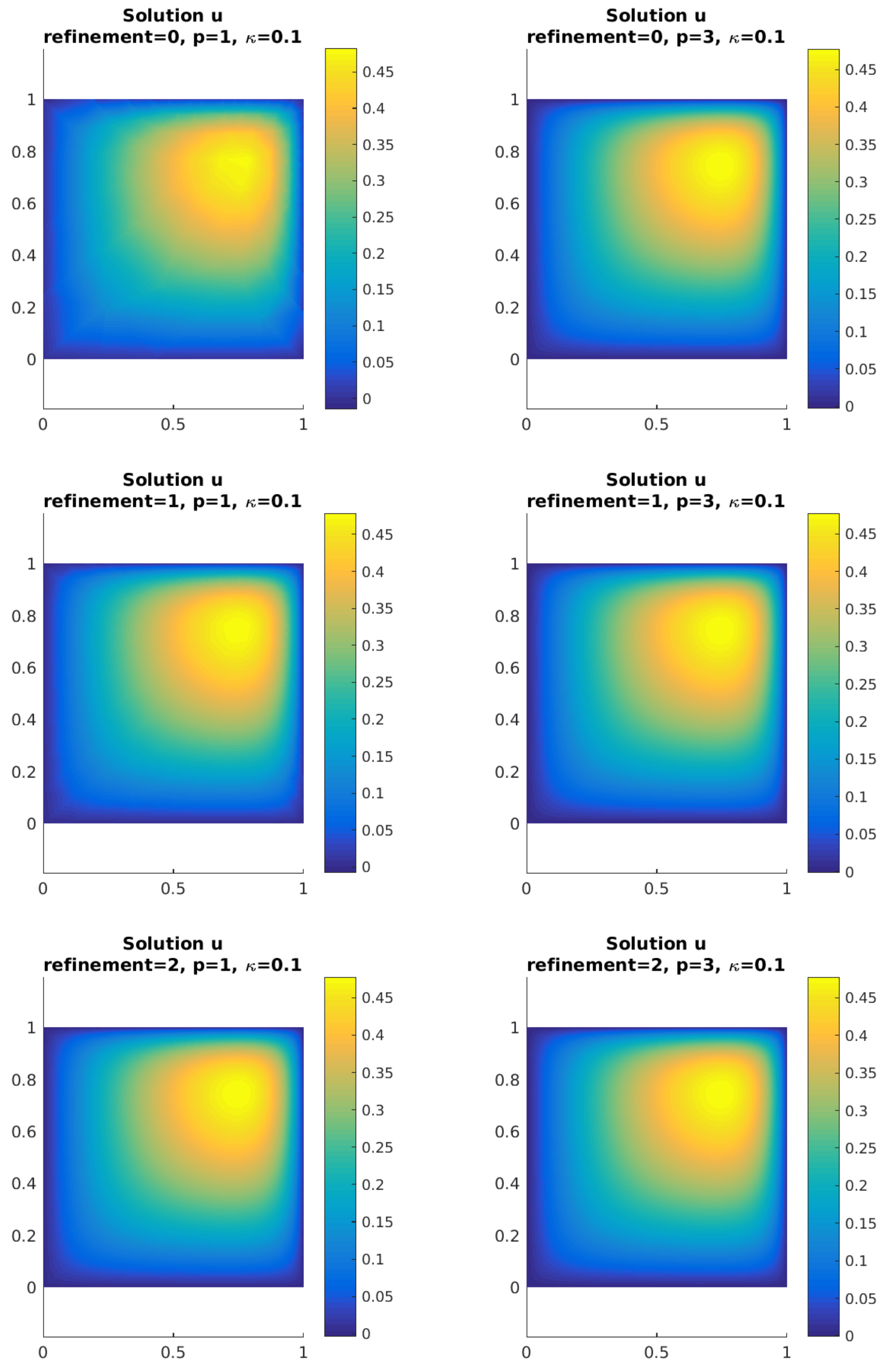

Figure 5: The mixed case with $\kappa = 0.1$. Solutions from the p=1 elements are shown on the left while the p=3 elements are on the right.

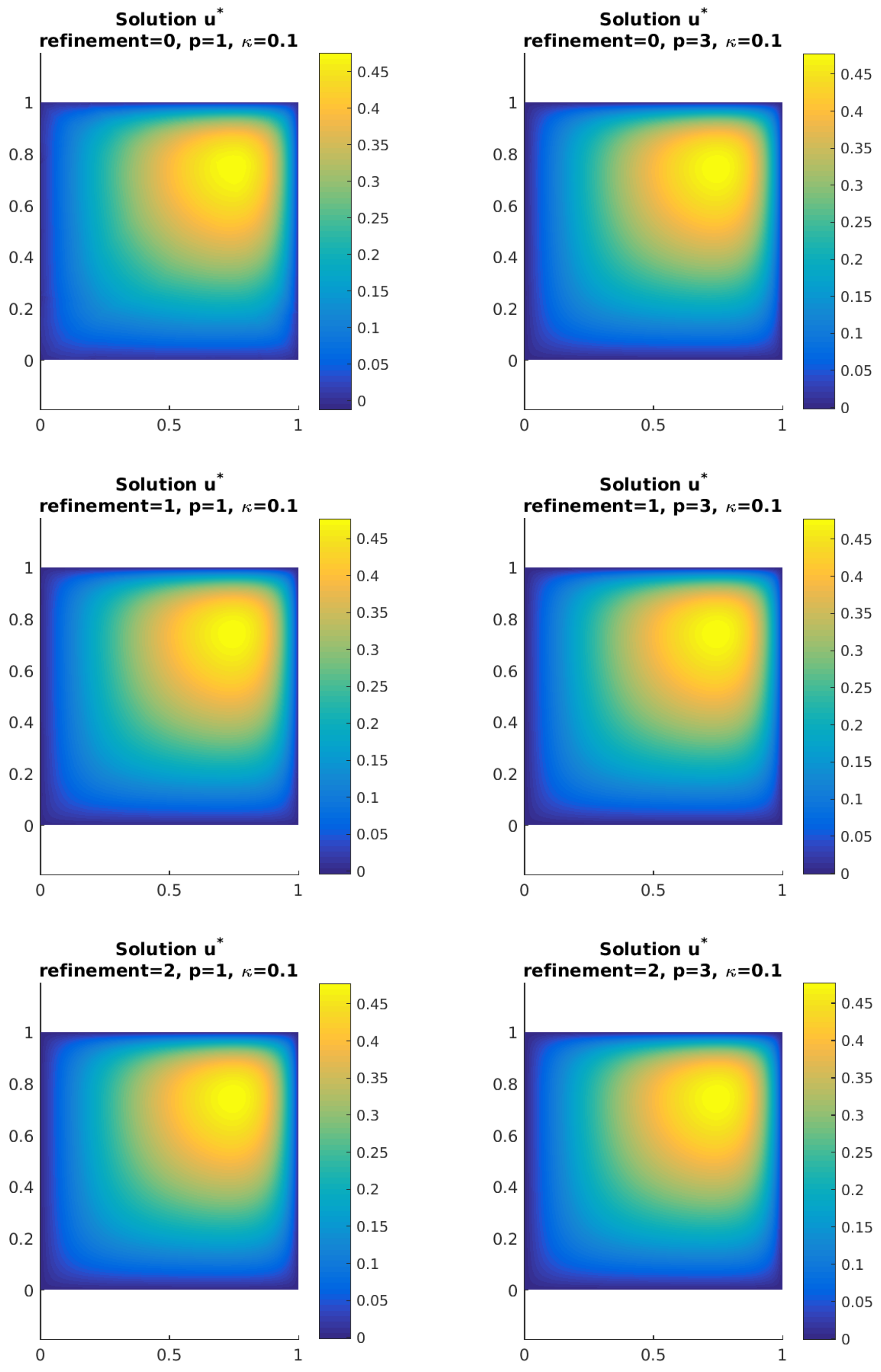

Figure 6: The mixed case with $\kappa = 0.1$. Postprocessed solutions from the p=1 elements are shown on the left while the p=3 elements are on the right.

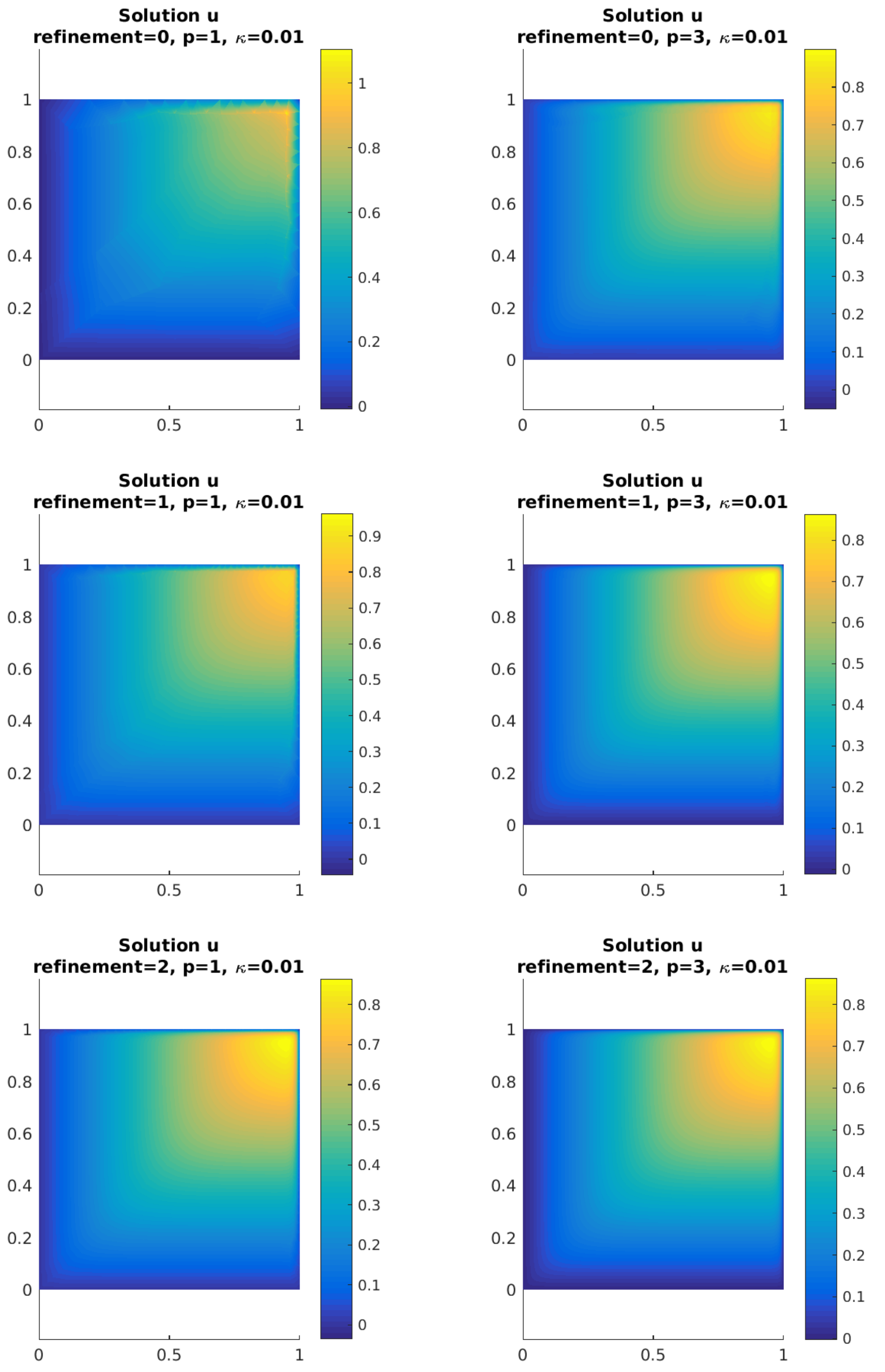

Figure 7: The convection-dominated case with $\kappa = 0.01$. Solutions from the p=1 elements are shown on the left while the p=3 elements are on the right.

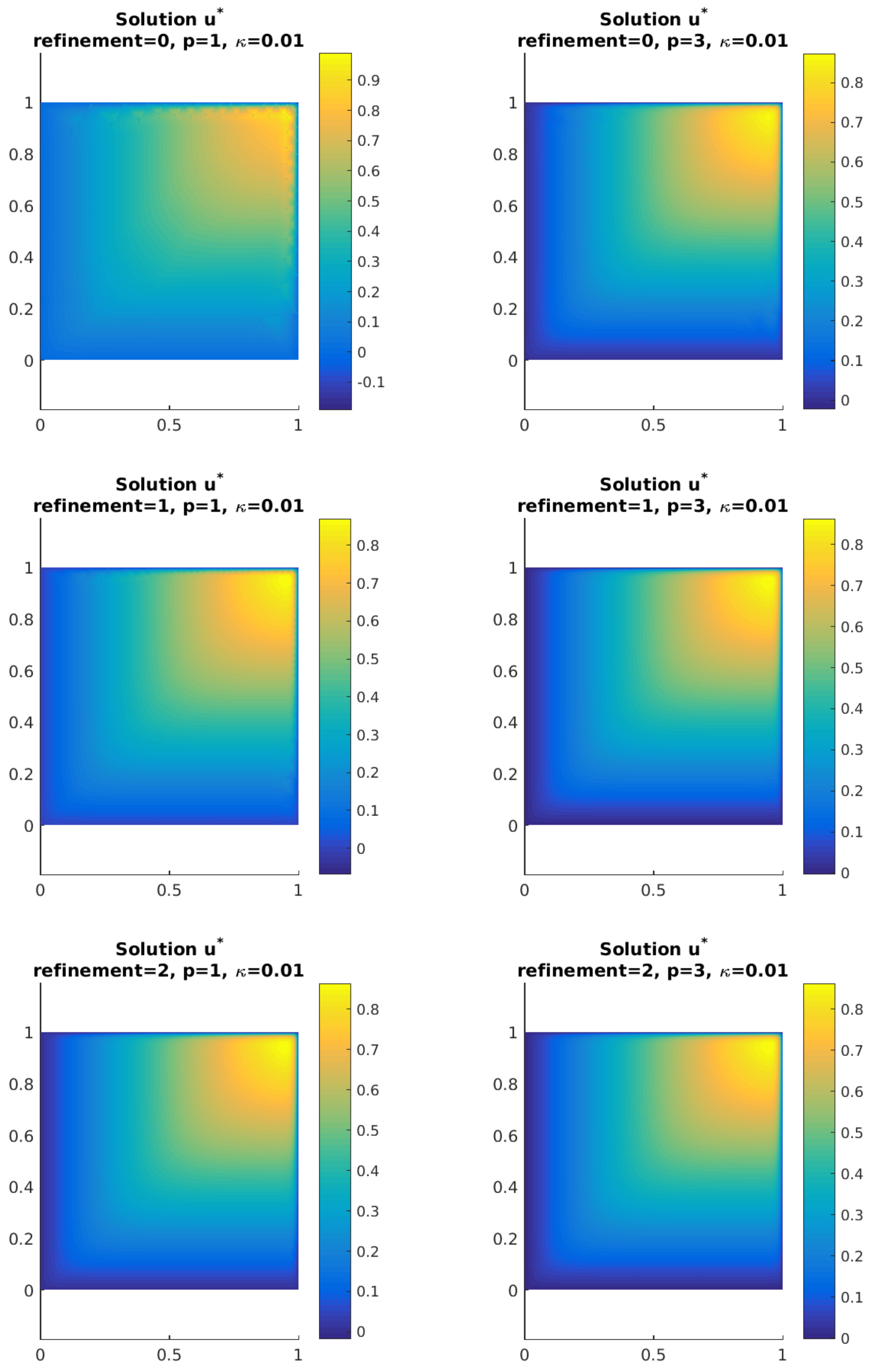

Figure 8: The convection-dominated case with $\kappa = 0.01$. Postprocessed solutions from the p=1 elements are shown on the left while the p=3 elements are on the right.