Università degli studi di Padova

Dipartimento di Ingegneria civile Edile e Ambientale

Corso di Laurea in Mathematical Engineering

Classe LM-44

Final Exam

Numerical Methods for Continuous System

Course given by:

Prof. ANTONIA LARESE

Prof. MARIO PUTTI

Student:

CIVIELLO SALVATORE

2124636

Academic Year 2024-2025

Contents

1 Project 1

The objective is to solve numerically the following problem:

Problem 1.1 *Find* $u(x,y) : \Omega := [0,1]\times[0,1] \to \mathbb{R}$ *of class* C^2 *such that*

$$\begin{cases} \frac{\partial}{\partial x}\left(K_x\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y\frac{\partial u}{\partial y}\right) - \frac{\partial \beta_x u}{\partial x} - \frac{\partial \beta_y u}{\partial y} = 0 & \text{in } \Omega, \\ u(x,y) = 1 & \text{on } \Gamma_1 := \{0\}\times[0,1] \cup [0,0.3]\times\{0\}, \\ u(x,y) = 0 & \text{on } \Gamma_2 := \partial\Omega/\Gamma_1. \end{cases} \tag{1}$$

with $K_x, K_y, \beta_x, \beta_y \in \mathbb{R}$ *and zero forcing.*

It should be solved with three different configurations for $K = (K_x, K_y)$ and $\beta = (\beta_x, \beta_y)$ and different algorithms:

- Case 1: $K = (8,4), \beta = (0,0)$ m/s using the Galerkin Finite Element Method (FEM).
- Case 2: $K_x = K_y$ varying in $[10^{-3}, 1]$, $\beta = (1,3)$ m/s with Galerkin FEM.
- Case 3: β as the previous case, $K_x = K_y = 0.01$ with Streamline Upwind Stabilization using $\tau = 0.01$ and $\tau = 1$.

The expression above is defined as the Strong form of the PDE problem. The useful form from which we get the matrices that we use in the code is the Weak form of the PDE:
we integrate the PDE over the domain and then we multiply both members of the equation by a test function v ($v | v \in H_0^1(\Omega)$); then we must apply the Green's lemma only to the first term, and so we get the following expression.

$$a(u,v) = 0 \quad \forall\, v \in H_0^1(\Omega) \tag{2}$$

where

$$a(u,v) = \int_\Omega K\nabla u \cdot \nabla v \; dx + \int_\Omega div(\beta u) v \; dx \tag{3}$$

1.1 Implementation overview

I decided to make 4 functions, similar to the ones showed during lectures, one for each important step:

- *input_var.m*: it reads the data present in the *mesh* folder
- *LocalB.m*: it builds the coefficients of the P_1 basis functions for each triangle and evaluates the area of each element
- *MatrixB.m*: it builds all the matrices used, the stiffness matrix H, the transport matrix B and the streamline upwind diffusion elemental matrix S, using the local coefficients computed by *LocalB.m*, and K, β and τ
- *lifting.m*: it is used to impose the boundary condition, by applying the "boundary lifting operator"

In the main script, depending on the case to solve, I recall the useful parameters before the *MatrixB.m* function. Furthermore, to use the function *MatrixB.m* for both stabilized and unstabilized FEM, I set the norm of the velocity vector $\left|\beta^{(e)}\right|$ to 1 if it is 0, because the elements s_{ij} of the stabilizing matrix S have this value in the denominator. Meanwhile, because the parameter τ appears in the numerator, if $\tau = 0$, the matrix S is omitted as required.
The global matrices H, B, and S are computed by summing the local matrices over all the elements, as usual. Local matrices are computed using the P_1 basis function on a triangular mesh. Since this kind of basis function are linear on x and y, their gradient are constant; in fact, we can use these constant coefficients to implement our local matrices:

$$\begin{aligned} b_i^{(e)} &= \frac{\partial \phi_i^{(e)}}{\partial x} \\ c_i^{(e)} &= \frac{\partial \phi_i^{(e)}}{\partial y} \end{aligned} \tag{4}$$

for each node i of the element e.
Hence, all terms involving the derivative of the basic functions, multiplied by some other constant parameter, can be taken out of the integrals.

Recalling that Δ is the area of the triangular element that we are considering, and that the integral over the triangular element is its area, we have:

$$\begin{aligned} h_{ij} &= (K_{xx}\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x} + K_{yy}\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y})\Delta \\ b_{ij} &= (\beta_x\frac{\partial\phi_j}{\partial x} + \beta_y\frac{\partial\phi_j}{\partial y})\Delta \\ s_{ij} &= \tau\frac{\left|h^{(e)}\right|}{K^e\left|\beta^{(e)}\right|}(\beta_x\frac{\partial\phi_i}{\partial x} + \beta_y\frac{\partial\phi_i}{\partial y})(\beta_x\frac{\partial\phi_j}{\partial x} + \beta_y\frac{\partial\phi_j}{\partial y})\Delta \end{aligned} \tag{5}$$

After this, there is the computation of the matrix $A = \mathsf{SYSMAT}$, summing all global matrices computed by the function $Matrix.m$, and then the right hand-side vector RHS is initialized by the vector of zeros.
By applying the boundary condition, the system SYSMAT u = RHS is solved by using the GMRES method, as required.

From the figure 1, we can see that the oscillations arise when the diffusion term $K \simeq 0.05$. In fact, by computing a pseudo-Peclet number:

$$\mathbb{P} = \frac{|\beta|\, h}{K} = 1.42\sqrt{10} \tag{6}$$

that is quite high; in fact, when we get a large Peclet number, we can say that our method is unstable. Using the SUD (Streamline Upwind Stabilization) we can see a better effect, in fact there is a reduction of the oscillations (figure 2).

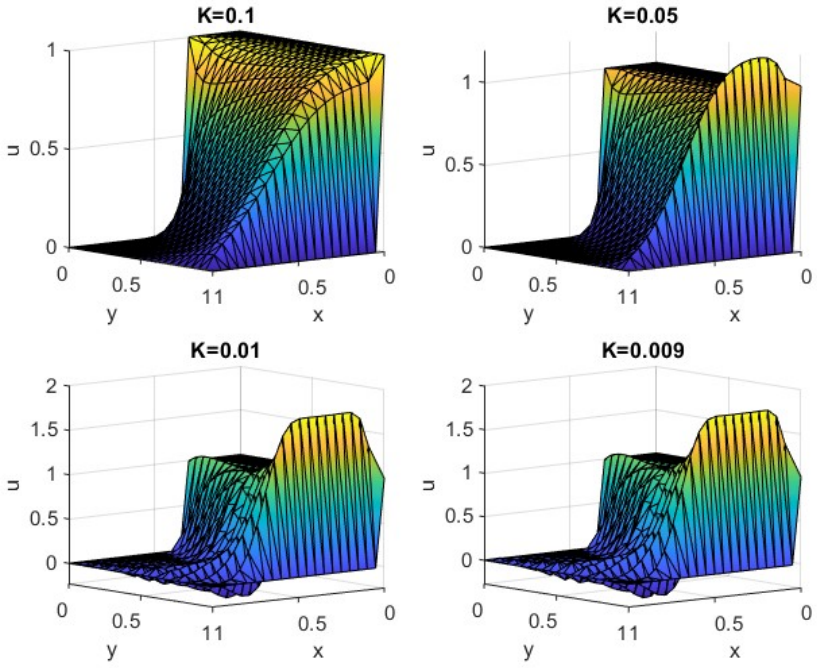

Figure 1: FEM with $\beta = (1, 3)^T$

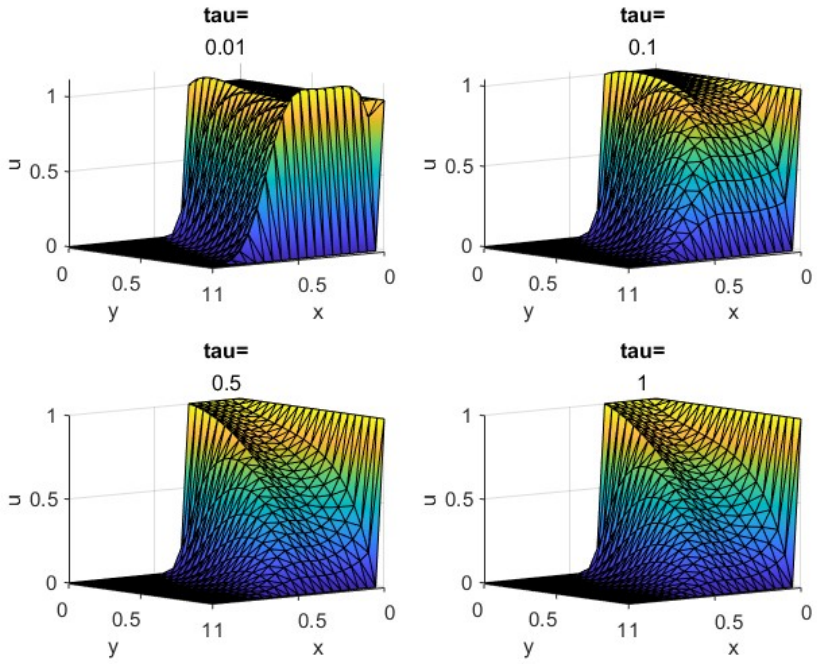

Figure 2: SUD-FEM with $K = (0.01, 0.01)^T$ and $\beta = (1, 3)^T$

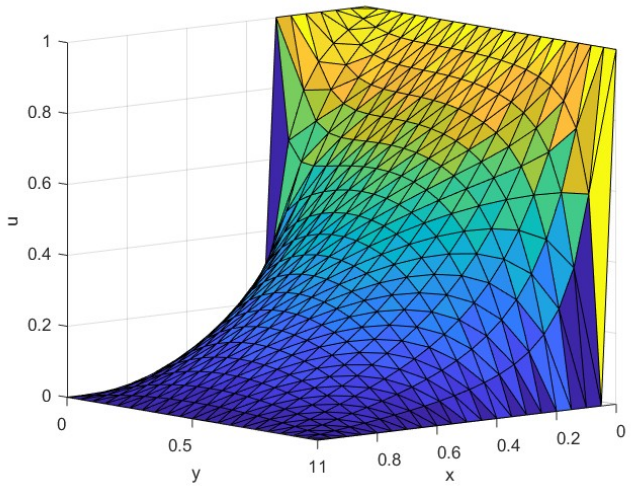

Figure 3: FEM with $K = (8, 4)^T$ and $\beta = (0, 0)^T$

2 Project 2

In this project, the task is to solve numerically the Stokes equations

Problem 2.2 *Find* $u(x, y); p(x, y)$ *such that*

$$\begin{cases} cu - \mu\Delta u + \nabla p = f & \text{in } \Omega, \\ \nabla \cdot u = 0 & \text{in } \Omega, \\ u(x, y) = u_D(x, y) & \text{on } \partial\Omega, \\ p(x, y) = p_D(x, y) & \text{on } \partial\Omega. \end{cases} \tag{7}$$

for a given value of c, μ, u_D, p_D *and* f.

The variational formulation is given by:

Problem 2.3 *Find* $(u, p) \in V \times Q$ *such that:*

$$\begin{cases} cm(u, v) + \mu a(u, v) + & b(v, p) = F(v) \quad \forall v \in V, \\ & b(u, q) = G(q) \quad \forall q \in Q \end{cases} \tag{8}$$

where

$$\begin{aligned} a(u, v) &= \mu \int_\Omega \nabla v : \nabla u \; dx \quad m(u, v) = \int_\Omega u \cdot v \; dx \\ b(v, p) &= -\int_\Omega p \; div(v) \quad F(v) = \int_\Omega f \cdot v \; dx \quad G(q) = -\int_\Omega gq \; dx \end{aligned} \tag{9}$$

Regarding the spaces we have:

$$\begin{aligned} V &= [H_0^1(\Omega)]^d \\ Q &= L_0^2(\Omega) = \{q \in L^2(\Omega) : \int_\Omega q \; dx = 0\} \end{aligned} \tag{10}$$

The numerical solution (u_h, p_h) can be expanded in terms of basis functions for the FEM spaces:

$$u_h = \sum_j u_j w_j(x) \quad p_h = \sum_j p_j \varphi_j(x) \tag{11}$$

And the linear system can be rewritten in matrix form

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \quad u = \begin{Bmatrix} u_{1,j} \\ u_{2,j} \end{Bmatrix}_{j=1,\dots} \quad p = \{p_k\}_{k=1,\dots}$$

at the end we will get

$$\begin{bmatrix} cM + \mu K & 0 & B1^T \\ 0 & cM + \mu K & B2^T \\ B1 & B2 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ 0 \end{bmatrix}$$

where $M, K, B1$ and $B2$ are obtained by the sum over the triangles of the mesh of their local matrices:

$$\begin{aligned} &M_{ij}^{(e)} = m(w_i, w_j), \quad K_{ij}^{(e)} = a(w_i, w_j), \\ &B1_{ij}^{(e)} = -\int_{\Omega^{(e)}} \varphi_i \frac{\partial \varphi_j}{\partial x}, \quad B2_{ij}^{(e)} = -\int_{\Omega^{(e)}} \varphi_i \frac{\partial \varphi_j}{\partial y} \end{aligned} \tag{12}$$

As I have written before, $\{w_i\}_i$ are the basis functions of the fluid velocity vector u while $\{\varphi_i\}_i$ are related to the fluid pressure p.

2.1 Implementation Overview

For this project, I decided to create a unified function that implements the three methods:

1. the difference between the *P1/P1* and the *P1/P1 GLS* is the presence of a different matrix A; in fact, in the second case, we have that

the zeros matrix in the southwest corner is replaced by the matrix $C = -\tau K$, with τ a positive parameter.
Moreover, the RHS includes the vector g, which sums over the nodes of the local vectors

$$g^{(e)} = (-\tau \int_{\Omega^{(e)}} f \cdot \nabla w_i)_i \tag{13}$$

2. Meanwhile, to implement the bubble scheme, the function requires a parameter (bu); if this parameter is different from zero, the function will compute the scheme using the bubble basis functions.
In this case, the function modifies the matrix A and the right-hand-side b, considering the additional node in the triangular mesh and increasing necessarily the local matrices from 3×3 to 4×4.
To compute the local matrices, we should add to the basis functions for $u_1,\ u_2,\ p$, a new one, the bubble function φ_b:

$$\varphi_b = 27\varphi_1\varphi_2\varphi_3 \quad \nabla\varphi_b = 27\sum_{l=1}^{3} \nabla\varphi_l\varphi_m\varphi_n \ \{l, m, n\} \equiv \{1, 2, 3\}. \tag{14}$$

For example, for a specific triangle $e^* = \{(x, y) : 0 \leq y \leq 1, 0 \leq x \leq 1 - y\}$, its basis function P_1 are:

$$\varphi_1^* = x \ \varphi_2^* = y \ \varphi_3^* = 1 - x - y. \tag{15}$$

Then, for a generic triangle e with area equal to Δ and for any set $I \subseteq \{1, 2, 3\}^n$, we get that

$$\int_{\Omega^{(e)}} \prod_{i \in I} \varphi_i = 2\Delta \int_{\Omega^{(e^*)}} \prod_{i \in I} \varphi_i^* \tag{16}$$

Using these integrals and the coefficients of the basis functions, it computes the elements of the local matrices.
The functions takes out the rows and columns in A related to the baricenters, and adjusts the boundary condition. Then, by applying the GMRES method, it computes the solution, and removes the baricenters value from it.
The last thing is the computation of the residual error with the exact solution, done only on the original mesh nodes.

In the end, I plotted the velocity components (u_1, u_2) and the pressure p for each algorithm, with the exact solution. As shown in the figures below (figures 7,8,9), in the case of spaces $P1/P1$, the pressure is more unstable with respect to the components of the velocity. I applied these algorithms to the Lid Driven Cavity problem, the stability of the methods.
In particular, the GLS method requires a small value of τ for the correctness (around 10^{-4}); for higher values the results become excessively smoothed.

	mesh0	mesh1	mesh2	mesh3	mesh4
P1/P1	1.4211	0.76995	0.39105	0.19614	0.097983
P1/P1 GLS	1.3667	0.73228	0.37248	0.18698	0.093582
P1-bubble/P1	1.3818	0.75416	0.38593	0.1945	0.097523

Figure 4: u1: residual errors for each mesh, with $\mu = 1$, $c = 1$ and $\tau = 0.1$

	mesh0	mesh1	mesh2	mesh3	mesh4
P1/P1	1.4206	0.76964	0.39088	0.19606	0.097941
P1/P1 GLS	1.3666	0.73222	0.37245	0.18697	0.093575
P1-bubble/P1	1.3813	0.75387	0.38577	0.19442	0.097482

Figure 5: u2: residual errors for each mesh, with $\mu = 1$, $c = 1$ and $\tau = 0.1$

	mesh0	mesh1	mesh2	mesh3	mesh4
P1/P1	15.336	21.542	22.705	35.484	60.1
P1/P1 GLS	1.6657	0.87263	0.43088	0.2142	0.10691
P1-bubble/P1	9.1573	6.6454	4.3189	3.0003	2.1332

Figure 6: p: residual errors for each mesh, with $\mu = 1$, $c = 1$ and $\tau = 0.1$

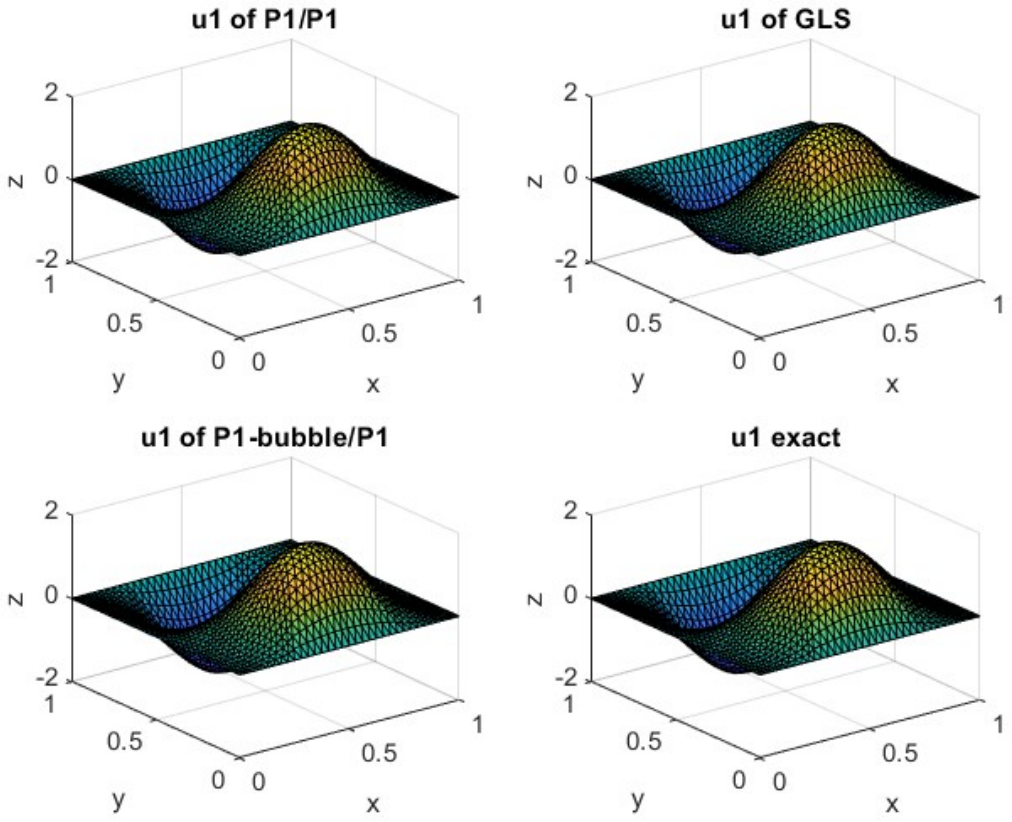

Figure 7: u1 with $\mu = 1$, $c = 1$ and $\tau = 0.1$ [mesh3]

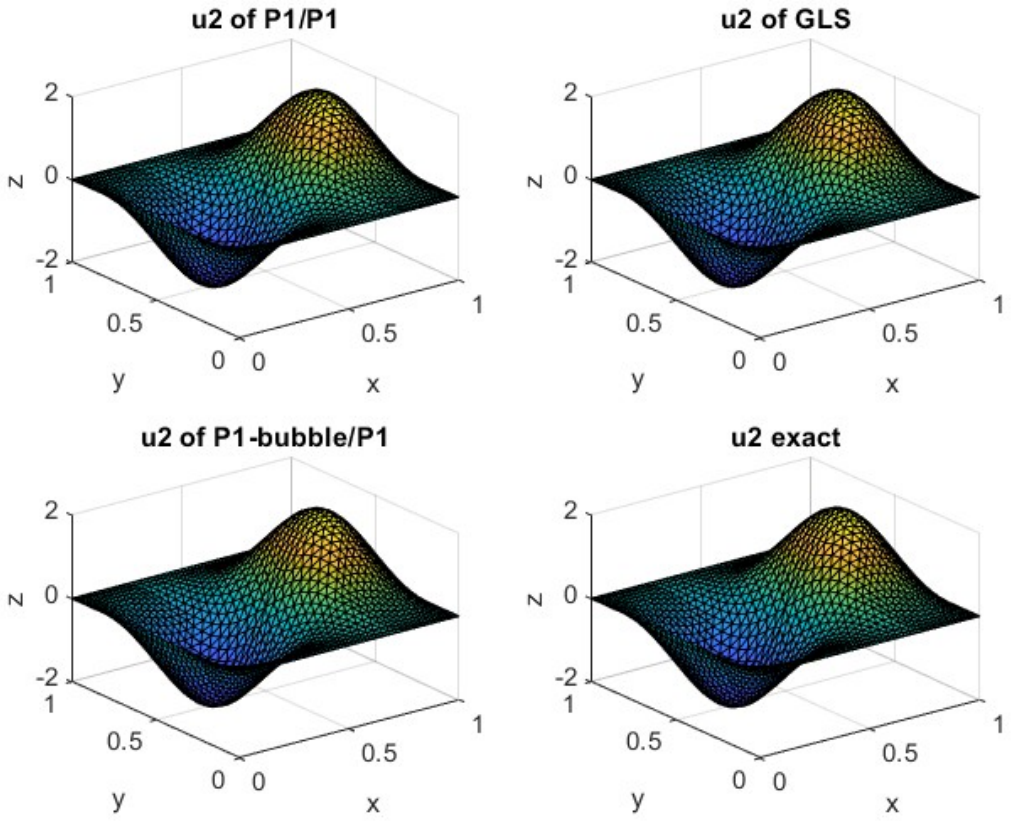

Figure 8: u2 with $\mu = 1$, $c = 1$ and $\tau = 0.1$ [mesh3]

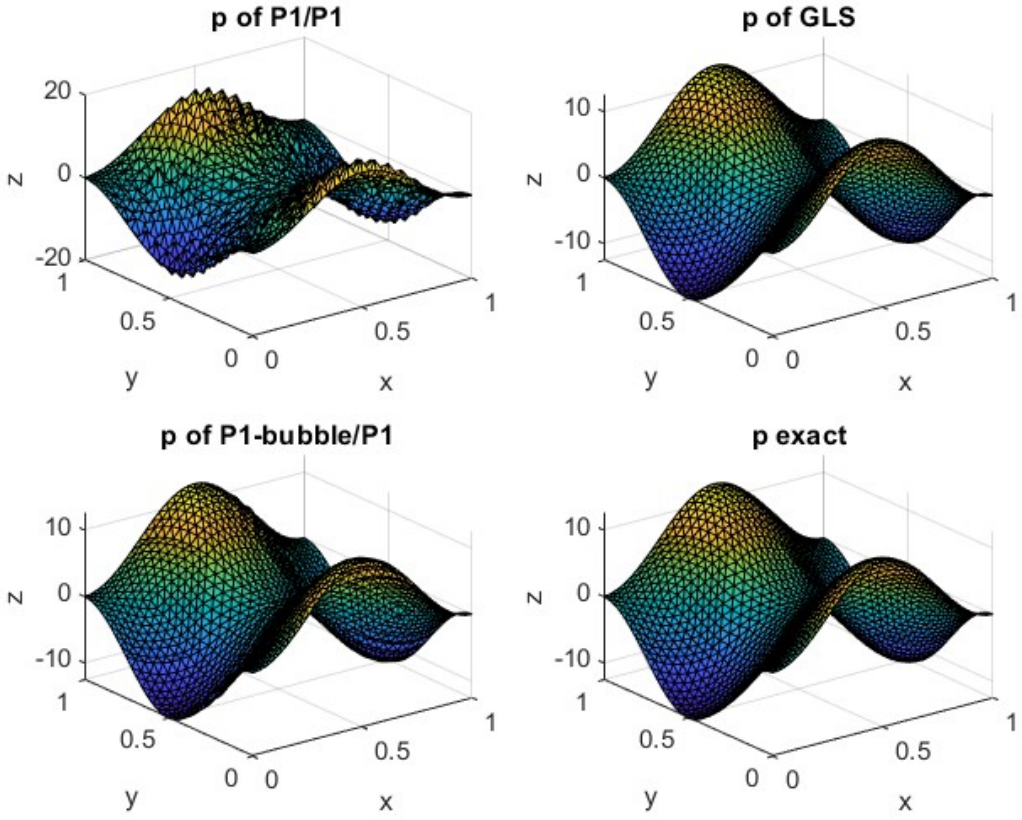

Figure 9: p with $\mu = 1$, $c = 1$ and $\tau = 0.1$ [mesh3]

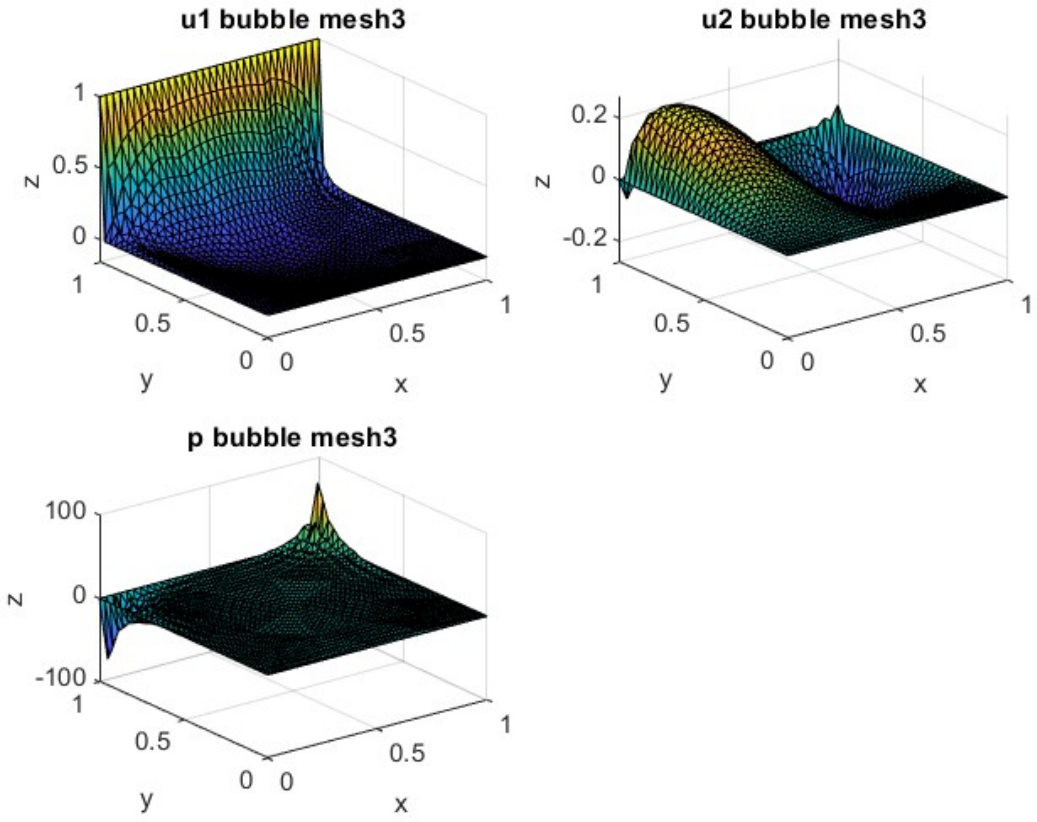

Figure 10: Bubble with $\mu = 1$ and $\tau = 10^{-4}$

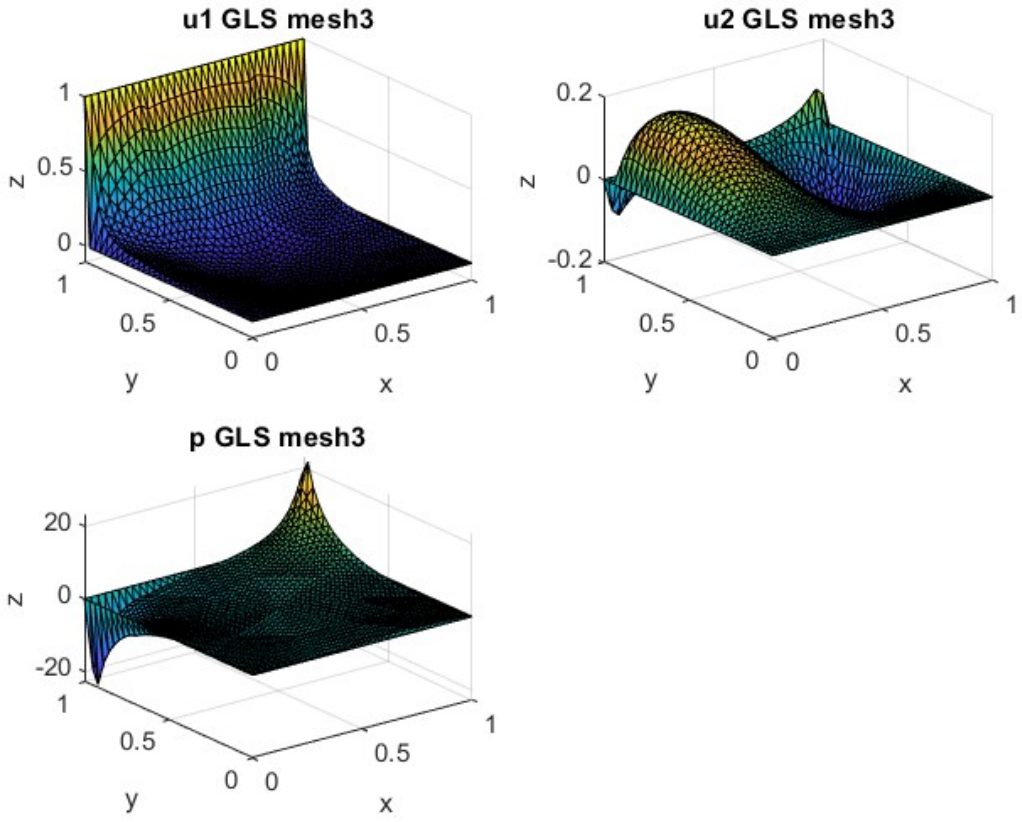

Figure 11: GLS with $\mu = 1$ and $\tau = 10^{-4}$