Numerical Methods for Continuous Systems
Stokes Project

Marangon Gaia
NM: 1242310

1 Description of the problem

1.1 Differential formulation

In this report, we analyze the Stokes problem in its extended version, expressed by the following differential formulation:

$$\begin{aligned}
&cu - \mu\triangle u + \nabla p = f && \text{in } \Omega\\
&div(u) = g && \text{in } \Omega\\
&u = u_D && \text{in } \Gamma\\
&\int_\Omega p\, d\Omega = 0
\end{aligned}$$

where c is a constant [1], μ is the dynamic viscosity, f is the forcing function, g is the divergence function. The unknowns of the problem are the fluid velocity vector u - with Dirichlet boundary conditions u_D - and the fluid pressure p - with zero mean over the domain. In this report we consider $\Omega = \{(x,y) \in \mathbb{R}^2 :\ 0 \le x \le 1,\ 0 \le y \le 1\}$ with boundary $\Gamma = \partial\Omega$.

1.2 Weak formulation

The weak formulation of the problem can be built as usual: we multiply both equations by a test function (a vectorial test function v for the first equation, a scalar test function q for the second equation), we apply the Green's Lemma and we change signs in the second equation, obtaining:

$$\text{Find } (u,p) \in \mathcal{V} \times \mathcal{Q} \text{ s.t.:} \qquad \begin{aligned} &a(u,v) + b(v,p) = F(v) && \forall v \in \mathcal{V}\\ &b(u,q) = G(u) && \forall q \in \mathcal{Q}\end{aligned}$$

where the bilinear forms $a(u,v)$, $b(v,p)$ and the functionals $F(v)$, $G(q)$ are defined as:

$$\begin{aligned}
&a(u,v) = \mu \int_\Omega \nabla u : \nabla v\, d\Omega + c\int_\Omega u\cdot v\, d\Omega && F(v) = \int_\Omega f\cdot v\, d\Omega\\
&b(v,p) = -\int_\Omega p\, div(v)\, d\Omega && G(q) = -\int_\Omega g\, q\, d\Omega
\end{aligned}$$

[1] The physical meaning of this constant depends on the problem we model. For example, it may appear when dealing with the time-dependent Stokes problem, and it may represent the term of an implicit Euler discretization (in this case c would be the inverse of the time step)

while the spaces $\mathcal{V}$, $\mathcal{Q}$ are defined as:

$$\mathcal{V} = [H_0^1(\Omega)]^2$$
$$\mathcal{Q} = L_0^2(\Omega) = \left\{ q \in L^2(\Omega) : \int_\Omega q \, d\Omega = 0 \right\}$$

1.3 Galerkin Finite Element formulation

Finally we pass to Galerkin Finite Element formulation. By choosing a regular triangulation $\mathcal{T}_h(\Omega)$ of the domain, we can express the discretized problem as:

$$\text{Find } (u_h, p_h) \in \mathcal{V}_h(\mathcal{T}_h) \times \mathcal{Q}_h(\mathcal{T}_h) \text{ s.t.:} \qquad \begin{aligned} &a(u_h, v_h) + b(v_h, p_h) = F(v_h) && \forall v_h \in \mathcal{V}_\langle \\ &b(u_h, q_h) = G(u_h) && \forall q_h \in \mathcal{Q}_\langle \end{aligned}$$

We choose a set of basis functions for both spaces:

$$\begin{aligned} &\text{for } \mathcal{V}_h(\mathcal{T}_h) : && \{\vec{\phi}_i(x,y) = (\phi_i^{(1)}(x,y),\, \phi_i^{(2)}(x,y)\,)\} && i = 1, \dots, N \\ &\text{for } \mathcal{Q}_h(\mathcal{T}_h) : && \{\varphi_k(x,y)\} && k = 1, \dots, M \end{aligned}$$

For simplicity, we assume $\phi_i(x,y) := \phi_i^{(1)}(x,y) = \phi_i^{(2)}(x,y)$, using the same quantity for both components of the vectorial basis function $\vec{\phi}_i$.
We can use these basis functions to expand u_h, v_h, p_h, q_h as follows:

$$u_h = \begin{pmatrix} u_h^{(1)} \\ u_h^{(2)} \end{pmatrix} = \sum_{j=1}^{N} \begin{pmatrix} u_j^{(1)} \phi_j(x,y) \\ u_j^{(2)} \phi_j(x,y) \end{pmatrix} \qquad v_h = \begin{pmatrix} v_h^{(1)} \\ v_h^{(2)} \end{pmatrix} = \sum_{i=1}^{N} \begin{pmatrix} v_i^{(1)} \phi_i(x,y) \\ v_i^{(2)} \phi_i(x,y) \end{pmatrix}$$
$$p_h = \sum_{l=1}^{M} p_l \varphi_l(x,y) \qquad q_h = \sum_{k=1}^{M} q_k \varphi_k(x,y)$$

Then, we gradually substitute these expressions in our equations. As a first step, we substitute the test functions v_h, q_h and, by recalling that the equations must hold for all values of the components $v_i^{(1)}$, $v_i^{(2)}$, q_k, we can split them into a system of $N + N + M$ equations:

$$\begin{aligned} c\int_\Omega u_h^{(1)} \phi_i \, d\Omega + \mu \int_\Omega \nabla u_h^{(1)} \cdot \nabla \phi_i \, d\Omega - \int_\Omega p_h \frac{\partial \phi_i}{\partial x} \, d\Omega &= \int_\Omega f^{(1)} \phi_i \, d\Omega && \forall i = 1, \dots, N \\ c\int_\Omega u_h^{(2)} \phi_i \, d\Omega + \mu \int_\Omega \nabla u_h^{(2)} \cdot \nabla \phi_i \, d\Omega - \int_\Omega p_h \frac{\partial \phi_i}{\partial y} \, d\Omega &= \int_\Omega f^{(2)} \phi_i \, d\Omega && \forall i = 1, \dots, N \\ -\int_\Omega \varphi_k \, div(u_h) \, d\Omega &= -\int_\Omega g \, \varphi_k \, d\Omega && \forall k = 1, \dots, M \end{aligned}$$

Secondly, we substitute the expressions for the unknowns u_h, p_h. By properly collecting the resulting terms, we can finally rewrite the system of equations in matrix form:

$$\begin{bmatrix} cM + \mu K & 0 & B_{(1)}^T \\ 0 & cM + \mu K & B_{(2)}^T \\ B_{(1)} & B_{(2)} & 0 \end{bmatrix} \begin{bmatrix} \bar{u}^{(1)} \\ \bar{u}^{(2)} \\ \bar{p} \end{bmatrix} = \begin{bmatrix} \bar{f}^{(1)} \\ \bar{f}^{(2)} \\ -\bar{g} \end{bmatrix}$$

where the matrix blocks are defined as follows:

$$
\begin{array}{llll}
M = \{m_{ij}\} & m_{ij} = \int_\Omega \phi_i \phi_j \, d\Omega & B_{(1)} = \{b^{(1)}_{kj}\} & b^{(1)}_{kj} = -\int_\Omega \varphi_k \frac{\partial \phi_j}{\partial x} \, d\Omega \\
K = \{k_{ij}\} & k_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j d\Omega & B_{(2)} = \{b^{(2)}_{kj}\} & b^{(2)}_{kj} = -\int_\Omega \varphi_k \frac{\partial \phi_j}{\partial y} \, d\Omega \\
\bar{f}^{(1)} = \{f^{(1)}_i\} & f^{(1)}_i = \int_\Omega f^{(1)}(x,y)\phi_i \, d\Omega & \bar{g} = \{g_k\} & g_k = \int_\Omega g(x,y)\varphi_k \, d\Omega \\
\bar{f}^{(2)} = \{f^{(2)}_i\} & f^{(2)}_i = \int_\Omega f^{(2)}(x,y)\phi_i \, d\Omega & &
\end{array}
$$

while the discretized unknowns are as usual the expansion's coefficients:

$$
\bar{u}^{(1)} = \{u^{(1)}_j\} \qquad \bar{u}^{(2)} = \{u^{(2)}_j\} \qquad \bar{p} = \{p_l\}
$$

2 Implementation

The specific way of implementing the Galerkin Finite Element method depends on the choice we make for the set of basis functions in each space. Different choices are available, belonging to two main categories: the inf-sup unstable spaces, which are easier to handle but present oscillations on the pressure and therefore need to be adjusted by mean of some stabilization term; and the inf-sup stable methods, involving an enriched space for the velocity and therefore fulfilling the conditions that guarantee stability. In this report we implement three different choices of basis functions:

- $\mathcal{P}_1/\mathcal{P}_1$ spaces: they do not satisfy inf-sup stability conditions, hence they result in an unstable method.
- GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$ spaces: with the same choice of basis, we add a residual-based stabilization term, obtaining a neat solution with no oscillations. In this way we commit a variational crime, but it becomes negligible as the stepsize of the discretization tends to zero, resulting in a consistent method.
- $\mathcal{P}_1$bubble$/\mathcal{P}_1$ spaces: they satisfy inf-sup conditions, therefore stability is guaranteed without recurring to any correction.

2.1 $\mathcal{P}_1/\mathcal{P}_1$

This method makes use of the same basis functions for both the velocity and the pressure space. All the functions $\{\phi_i^{(1)}(x,y)\}_{i=1,\dots,N}$, $\{\phi_i^{(2)}(x,y)\}_{i=1,\dots,N}$, $\{\varphi_k(x,y)\}_{k=1,\dots,M}$ are piecewise linear polynomial, in the same form:

$$\{\phi_i(x,y)\}_{i=1,\dots,N_{nodes}} \qquad\qquad \phi_i^{(T)}(x,y) = \frac{a_i^{(T)} + b_i^{(T)}x + c_i^{(T)}y}{2|T|}$$

where $|T|$ is the area of the triangle and the coefficients $a_i^{(T)}$, $b_i^{(T)}$, $c_i^{(T)}$ are defined by imposing the values of the function at the vertices. By calling i, j, k the local indexing of the vertices in the triangle, with anticlockwise order, we obtain:

$$a_i = x_j y_k - x_k y_j \qquad\qquad b_i = y_j - y_k \qquad\qquad c_i = x_k - x_j$$

while the elemental area can be computed using the Van Der Monde matrix, as:

$$VDM = \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} \qquad\qquad |T| = \frac{|det(VDM)|}{2}$$

We can also recall some simple relations which will be helpful in the following:

$$\frac{\partial \phi_i}{\partial x} = \frac{b_i}{2|T|} \qquad \frac{\partial \phi_i}{\partial y} = \frac{c_i}{2|T|} \qquad \int_T \phi_i \, d\Omega = \frac{|T|}{3} \qquad \int_T \phi_i^\alpha \phi_j^\beta \phi_k^\gamma \, d\Omega = \frac{2\alpha!\beta!\gamma!}{(\alpha+\beta+\gamma)!}|T|$$

$$\sum_{r=1}^{3} b_r = 0 \qquad \sum_{r=1}^{3} c_r = 0$$

Using all these properties, we can compute explicitly the elemental contributions for each block matrix of the system, obtaining:

$$m_{ij} = \begin{cases} \frac{|T|}{12} & \text{if } i \neq j \\ \frac{|T|}{6} & \text{if } i = j \end{cases} \qquad\qquad \begin{matrix} b_{kj}^{(1)} = -\frac{b_j}{6} \\ b_{kj}^{(2)} = -\frac{c_j}{6} \end{matrix} \qquad\qquad k_{ij} = \frac{b_i b_j + c_i c_j}{4|T|}$$

while for the right hand side we may apply the midpoint rule and obtain, by calling (x_G, y_G) the coordinates of the barycenter:

$$f_i^{(1)} = \sum_{T \ni i} f^{(1)}(x_G, y_G)\frac{|T|}{3} \qquad f_i^{(2)} = \sum_{T \ni i} f^{(2)}(x_G, y_G)\frac{|T|}{3} \qquad g_k = \sum_{T \ni k} g(x_G, y_G)\frac{|T|}{3}$$

2.2 GLS stabilized $\mathcal{P}_1/\mathcal{P}_1$

In order to solve the problem of oscillations, we introduce a residual-based stabilization term, in the form:

$$c\big((u,p),\,(v,q)\big) = \delta \sum_T h_T^2 \int_T (-\mu\triangle u + \nabla p - f)\,(-\mu\triangle v + \nabla q)\, d\Omega$$

where h_T is the length of the maximum edge in the triangle and δ is the constant coefficient used to calibrate the amount of stabilization. This term is included in the equations of the variational formulation, as follows:

$$\begin{array}{lll} \text{Find } (u,p) \in \mathcal{V} \times \mathcal{Q} \text{ s.t.:} & a(u,v) + b(v,p) = F(v) & \forall v \in \mathcal{V} \\ & b(u,q) + c\big((u,p),\,(v,q)\big) = G(u) & \forall q \in \mathcal{Q} \end{array}$$

When passing to the Galerkin Finite Element formulation, this term undergoes the same procedures described above for the other terms of the equation (functions u, v, p, q are discretized and expanded, the equations are split and the terms are collected, obtaining a matrix form). The resulting discretized system is now:

$$\begin{bmatrix} cM + \mu K & 0 & B_{(1)}^T \\ 0 & cm + \mu K & B_{(2)}^T \\ B_{(1)} & B_{(2)} & -J \end{bmatrix} \begin{bmatrix} \bar{u}^{(1)} \\ \bar{u}^{(2)} \\ \bar{p} \end{bmatrix} = \begin{bmatrix} \bar{f}^{(1)} \\ \bar{f}^{(2)} \\ -\bar{g} - \bar{g}_{stab} \end{bmatrix}$$

It includes two new contributions: a block $-J$ in the system matrix and a term $-\bar{g}_{stab}$ in the right hand side, whose components are respectively:

$$\begin{aligned} J = \{j_{kl}\} \qquad\qquad j_{kl} &= \delta \sum_T h_T^2 \int_T \nabla\varphi_k \cdot \nabla\varphi_l \, d\Omega \\ &= \delta \sum_T h_T^2 \frac{b_k b_l + c_k c_l}{4|T|} \\ \bar{g}_{stab} = \{g_{stab,k}\} \qquad\qquad g_{stab,k} &= \sum_{T \ni k} \delta h_T^2 \int_T \left[f^{(1)}(x,y)\frac{\partial\varphi_k}{\partial x} + f^{(2)}(x,y)\frac{\partial\varphi_k}{\partial y} \right] d\Omega \\ &= \sum_{T \ni k} \frac{\delta h_T^2}{2}\left[b_k f^{(1)}(x_G,y_G) + c_k f^{(2)}(x_G,y_G)\right] \end{aligned}$$

2.3 $\mathcal{P}_1$bubble/$\mathcal{P}_1$

This method makes use of different basis functions for the velocity and the pressure space. In particular, the pressure basis functions are still piecewise linear polynomial in the usual form:

$$\{\phi_i(x,y)\}_{i=1,\dots,N_{nodes}} \qquad\qquad \phi_i^{(T)}(x,y) = \frac{a_i^{(T)} + b_i^{(T)}x + c_i^{(T)}y}{2|T|}$$

with coefficients $a_i^{(T)}$, $b_i^{(T)}$, $c_i^{(T)}$ and area $|T|$ defined and computed as above. As for the velocity space, we use - for each component of the velocity - an enriched set of basis functions, including: N_{nodes} piecewise linear functions $\{\phi_i(x,y)\}_{i=1,\dots,N_{nodes}}$ (the same functions we use for the pressure space), plus N_{elem} new functions $\{\psi_m(x,y)\}_{m=1,\dots,N_{elem}}$, called bubble functions, each having non-zero value only on a specific element of the triangulation. The definition of these bubble functions involves the three piecewise linear functions which are non-zero on that specific element:

$$\psi_m(x,y) = 27\phi_1(x,y)\phi_2(x,y)\phi_3(x,y)$$

where the indices 1, 2, 3 refer to the local numbering of the vertices. With this definition, the bubble function has value zero on the vertices and value one on the barycenter of the triangle. It provides the sufficient enrichment needed to guarantee inf-sup stability.

With these basis, we can expand our unknown functions and test functions as we did in section 1.3:

$$u_h = \begin{pmatrix} u_h^{(1)} \\ u_h^{(2)} \end{pmatrix} = \sum_{j=1}^{N_{nodes}} \begin{pmatrix} u_j^{(1)}\phi_j(x,y) \\ u_j^{(2)}\phi_j(x,y) \end{pmatrix} + \sum_{n=1}^{N_{elem}} \begin{pmatrix} u_n^{(1)b}\psi_n(x,y) \\ u_n^{(2)b}\psi_n(x,y) \end{pmatrix}$$

$$v_h = \begin{pmatrix} v_h^{(1)} \\ v_h^{(2)} \end{pmatrix} = \sum_{i=1}^{N_{nodes}} \begin{pmatrix} v_i^{(1)}\phi_i(x,y) \\ v_i^{(2)}\phi_i(x,y) \end{pmatrix} + \sum_{m=1}^{N_{elem}} \begin{pmatrix} v_m^{(1)b}\psi_m(x,y) \\ v_m^{(2)b}\psi_m(x,y) \end{pmatrix}$$

$$p_h = \sum_{l=1}^{N_{nodes}} p_l\phi_l(x,y) \qquad\qquad q_h = \sum_{k=1}^{N_{nodes}} q_k\phi_k(x,y)$$

and then proceed in the same way, by substituting, splitting the equations (this time we'll get $2(N_{nodes}+N_{elem})+N_{nodes}$ equations), and then properly collecting the terms in such a way to obtain a discretized system. The result is:

$$\begin{bmatrix} c\{m_{ij}\}+\mu\{k_{ij}\} & c\{m_{in}\}+\mu\{k_{in}\} & 0 & 0 & \{b_{il}^{(1)}\} \\ c\{m_{mj}\}+\mu\{k_{mj}\} & c\{m_{mn}\}+\mu\{k_{mn}\} & 0 & 0 & \{b_{ml}^{(1)}\} \\ 0 & 0 & c\{m_{ij}\}+\mu\{k_{ij}\} & c\{m_{in}\}+\mu\{k_{in}\} & \{b_{il}^{(2)}\} \\ 0 & 0 & c\{m_{mj}\}+\mu\{k_{mj}\} & c\{m_{mn}\}+\mu\{k_{mn}\} & \{b_{ml}^{(2)}\} \\ \{b_{kj}^{(1)}\} & \{b_{kn}^{(1)}\} & \{b_{kj}^{(2)}\} & \{b_{kn}^{(2)}\} & 0 \end{bmatrix} \begin{bmatrix} \{u_j^{(1)}\} \\ \{u_n^{(1)b}\} \\ \{u_j^{(2)}\} \\ \{u_n^{(2)b}\} \\ \{p_l\} \end{bmatrix} = \begin{bmatrix} \{f_i^{(1)}\} \\ \{f_m^{(1)b}\} \\ \{f_i^{(2)}\} \\ \{f_m^{(2)b}\} \\ -\{g_k\} \end{bmatrix}$$

where, for simplicity, block matrices are already described by their components, with indices ranging as follows: $i,j=1,\dots,N_{nodes}$; $m,n=1,\dots,N_{elem}$; $k,l=1,\dots,N_{nodes}$ [2].

The list of all contributions is given by:

$$\begin{aligned} m_{ij} &= \int_\Omega \phi_i\phi_j\,d\Omega & k_{ij} &= \int_\Omega \nabla\phi_i\cdot\nabla\phi_j\,d\Omega & b_{kj}^{(1)} &= (b_{il}^{(1)})^T = -\int_\Omega \phi_k\cdot\frac{\partial\phi_j}{\partial x}\,d\Omega \\ m_{mn} &= \int_\Omega \psi_n\psi_m\,d\Omega & k_{mn} &= \int_\Omega \nabla\psi_m\cdot\nabla\psi_n\,d\Omega & b_{kn}^{(1)} &= (b_{ml}^{(1)})^T = -\int_\Omega \phi_k\cdot\frac{\partial\psi_n}{\partial x}\,d\Omega \\ m_{mj} &= \int_\Omega \psi_m\phi_j\,d\Omega & k_{mj} &= \int_\Omega \nabla\psi_m\cdot\nabla\phi_j\,d\Omega & b_{kj}^{(2)} &= (b_{il}^{(2)})^T = -\int_\Omega \phi_k\cdot\frac{\partial\phi_j}{\partial y}\,d\Omega \\ m_{in} &= (m_{mj})^T & k_{in} &= (k_{mj})^T & b_{kn}^{(2)} &= (b_{ml}^{(2)})^T = -\int_\Omega \phi_k\cdot\frac{\partial\psi_n}{\partial y}\,d\Omega \end{aligned}$$

[2] In each couple, the first index is used to run over the rows, the second one to run over the columns

while as for the right hand side:

$$f_i^{(1)} = \int_\Omega f^{(1)}(x,y)\phi_i \, d\Omega \qquad f_m^{(1)b} = \int_\Omega f^{(1)}(x,y)\psi_m \, d\Omega \qquad g_k = \int_\Omega g(x,y)\phi_k \, d\Omega$$
$$f_i^{(2)} = \int_\Omega f^{(2)}(x,y)\phi_i \, d\Omega \qquad f_m^{(2)b} = \int_\Omega f^{(2)}(x,y)\psi_m \, d\Omega$$

More explicit computations can be done, by recalling the definition of ϕ_i and ψ_m and the simple relations we already used in section 2.1. The results are the following:

$$m_{ij} = \begin{cases} \frac{|T|}{12} & \text{if } i \neq j \\ \frac{|T|}{6} & \text{if } i = j \end{cases} \qquad m_{mn} = \begin{cases} 0 & \text{if } m \neq n \\ \frac{81|T|}{280} & \text{if } m = n \end{cases} \qquad m_{mj} = \frac{3|T|}{20}$$

$$\begin{aligned} k_{ij} &= \tfrac{b_i b_j + c_i c_j}{4|T|} \\ k_{mj} &= 0 \end{aligned} \qquad k_{mn} = \begin{cases} 0 & \text{if } m \neq n \\ \frac{81}{40|T|}\left(b_1^2 + c_1^2 + b_2^2 + c_2^2 + b_1 b_2 + c_1 c_2\right) & \text{if } m = n \end{cases}$$

$$b_{kj}^{(1)} = -\frac{b_j}{6} \qquad b_{kn}^{(1)} = \frac{9b_k}{40} \qquad b_{kj}^{(2)} = -\frac{c_j}{6} \qquad b_{kn}^{(2)} = \frac{9c_k}{40}$$

while for the right hand side:

$$f_i^{(1)} = \sum_{T \ni i} f^{(1)}(x_G, y_G)\frac{|T|}{3} \qquad f_m^{(1)b} = f^{(1)}(x_G, y_G)\frac{9|T|}{20} \qquad g_k = \sum_{T \ni k} g(x_G, y_G)\frac{|T|}{3}$$
$$f_i^{(2)} = \sum_{T \ni i} f^{(2)}(x_G, y_G)\frac{|T|}{3} \qquad f_m^{(2)b} = f^{(2)}(x_G, y_G)\frac{9|T|}{20}$$

3 The Project

3.1 Manufactured Problem

The first goal of this project is to test the convergence properties of the three methods described above. To this aim, we make use of five different meshes, with progressively halved stepsize ($h_0 = 2^{-2}, \ldots, h_4 = 2^{-6}$), resulting in an increasing refinement. With this setting, we test the Matlab code implementing each method on a manufactured solution:

$$\begin{aligned} u^{(1)}(x,y) &= -\,cos(2\pi x)sin(2\pi y) + sin(2\pi y) \\ u^{(2)}(x,y) &= \quad sin(2\pi x)cos(2\pi y) - sin(2\pi x) \\ p(x,y) &= \quad 2\pi\big(cos(2\pi y) - cos(2\pi x)\big) \end{aligned} \tag{1}$$

satisfying the Stokes equations with the following forcing and divergence functions:

$$\begin{aligned} f^{(1)}(x,y) &= cu^{(1)}(x,y) - 4\mu\pi^2 sin(2\pi y)\big(2cos(2\pi x) - 1\big) + 4\pi^2 sin(2\pi x) \\ f^{(2)}(x,y) &= cu^{(2)}(x,y) + 4\mu\pi^2 sin(2\pi x)\big(2cos(2\pi y) - 1\big) - 4\pi^2 sin(2\pi y) \\ g(x,y) &= 0 \end{aligned}$$

and with proper Dirichlet boundary conditions.
The aim is to observe the behavior of each method as the meshes grow refined and verify the presence of oscillations for the unstable method and the experimental convergence in the stabilized and inf-sup stable method.

3.2 Lid-driven Cavity Problem

The second part of the project addresses the so called "Lid-Driven-Cavity" problem, which is a particular case of the Stokes problem, formulated as:

$$\begin{aligned} &-\mu\triangle u + \nabla p = f = 0 && \text{in } \Omega \\ &div(u) = g = 0 && \text{in } \Omega \\ &u = (1,0)^T && \text{in } \Gamma_1 = \{(x,y) \in \Gamma : y = 1\} \\ &u = (0,0)^T && \text{in } \Gamma \setminus \Gamma_1 \\ &\int_\Omega p\, d\Omega = 0 && \text{in } \Omega \end{aligned} \tag{2}$$

This problem is solved using both stable schemes (GLS stabilized $\mathcal{P}_1/\mathcal{P}_1$ and $\mathcal{P}_1$bubble$/\mathcal{P}_1$).

4 The Program

All methods described above have been implemented in Matlab, resulting in codes with the following structure:

- *(inputData.m)* reading of the data from file (coordinates, triangulation, Dirichlet nodes and values) and definition of given constants (const c for mass matrix, dynamic viscosity μ, stabilization coefficient δ when needed) and functions (forcing and divergence functions; exact solutions for the manufactured problem)

- *(localBasis.m)* computation of element by element information (basis coefficients, elemental area and nodal area, barycenter)

- computation of the system matrix by blocks. This has been done with distinct functions in the case of the GLS stabilized $\mathcal{P}_1/\mathcal{P}_1$ method (*massBuild.m, stiffBuild.m, diverBuild.m, stabBuild.m*) and with a unique function in the case of $\mathcal{P}_1$bubble/$\mathcal{P}_1$ (*globBuild.m*). These two strategies are equivalent in terms of numerical results, but the first is more logical and easier to handle in case of error, while the second is more efficient in terms of computational time, due to the fact that we run over the elements just once. For this reason, the second strategy should be preferred, but we tested both ways.

- *(rhsBuild.m, rhsBuildStab.m)* computation of the right hand side. This has usually been done using the midpoint rule, even though the trapezoidal rule has been tested as well in the case of the GLS stabilized $\mathcal{P}_1/\mathcal{P}_1$ method. Another possibility would be that of expanding the forcing and divergence functions using the same basis we use for the velocity and the pressure, respectively. In this way, the right hand side as well can be written as a matrix-vector product, with all entries explicitly known. The numerical results are good, even though this procedure is probably slower in terms of computational time.

- *(imposeBC.m)* imposition of Dirichlet boundary conditions. Two methods have been tested, yielding similar results: the penalty method and the standard method. As for the pressure, we also need to impose the mean-zero condition. This is done by the Lagrangian multipliers method. It can be proved that, starting from a discretized system:

$$\begin{bmatrix} cM+\mu K & 0 & B_{(1)}^T \\ 0 & cm+\mu K & B_{(2)}^T \\ B_{(1)} & B_{(2)} & 0 \end{bmatrix} \begin{bmatrix} \bar{u}^{(1)} \\ \bar{u}^{(2)} \\ \bar{p} \end{bmatrix} = \begin{bmatrix} \bar{f}^{(1)} \\ \bar{f}^{(2)} \\ -\bar{g} \end{bmatrix}$$

  the Lagrangian method consists in adding a new component to the system:

$$\begin{bmatrix} cM+\mu K & 0 & B_{(1)}^T & 0 \\ 0 & cm+\mu K & B_{(2)}^T & 0 \\ B_{(1)} & B_{(2)} & 0 & A^T \\ 0 & 0 & A & 0 \end{bmatrix} \begin{bmatrix} \bar{u}^{(1)} \\ \bar{u}^{(2)} \\ \bar{p} \\ \lambda \end{bmatrix} = \begin{bmatrix} \bar{f}^{(1)} \\ \bar{f}^{(2)} \\ -\bar{g} \\ 0 \end{bmatrix}$$

  where A is the vector of nodal areas and λ is the Lagrange multiplier. The procedure is the same in all the methods we implemented, even though the system matrix we start from in each case is different, as already described.

- *(solveSys.m)* the system is solved. We tested three different methods: the Matlab backslash operator, the GMRES and the GMRES with lifting boundary operator. All methods work fine, yielding approximately the same results.

- *(plotSol.m, testError.m)* extra functions to plot the numerical (and exact, when given) solution and to compute, when possible, the node-by-node residuals.

5 Numerical Results

5.1 Experimental Convergence

As a first issue, we use the manufactured solution to test the convergence of the three methods described above ($\mathcal{P}_1/\mathcal{P}_1$; GLS-stabilized $\mathcal{P}_1/\mathcal{P}_1$; $\mathcal{P}_1$ bubble/$\mathcal{P}_1$). For each method, we make use of five different meshes with progressively halved stepsizes, so that the refinement is gradually increased ($h_0 = 2^{-2}, \ldots, h_4 = 2^{-6}$). In each case we compute the numerical solution (see figures from 1 to 4). Then, by exploiting the fact that the numerical solution is known, we set some numerical check, in order to draw some experimental information on the convergence of the methods. In particular, we compute at every node the difference between numerical and exact solution, and then we take the euclidean norm, obtaining the residual: $res^2(u) = \sum_{i=1}^{N_{nodes}} (\, u_{exact}(x_i, y_i) - u_{num,i}\,)^2$. We apply this procedure to both the whole numerical solution (*total res*) and each physical component ($res(u_1)$, $res(u_2)$, $res(p)$). The results are shown in table 1 and commented in the following.

Table 1: Table of residuals (euclidean norm of the difference between exact and numerical solution). Stabilization coefficient $\delta = 0.02$. Comparing meshes with growing refinement.

mesh:		0	1	2	3	4
$\mathcal{P}_1/\mathcal{P}_1$:	$res(u_1)$	0.45	0.19	0.09	0.04	0.02
	$res(u_2)$	0.45	0.19	0.09	0.04	0.02
	$res(p)$	11.26	21.84	22.58	35.47	60.11
	total res	11.28	21.84	22.58	35.47	60.11
$GLS - \mathcal{P}_1/\mathcal{P}_1$:	$res(u_1)$	0.38	0.15	0.07	0.04	0.02
	$res(u_2)$	0.38	0.15	0.07	0.04	0.02
	$res(p)$	2.67	2.38	1.70	1.23	0.87
	total res	2.73	2.39	1.70	1.23	0.87
$\mathcal{P}_1 - bubble/\mathcal{P}_1$:	$res(u_1)$	0.39	0.16	0.08	0.04	0.02
	$res(u_2)$	0.39	0.16	0.08	0.04	0.02
	$res(p)$	4.15	2.95	2.46	1.87	1.37
	total res	4.19	2.95	2.46	1.87	1.37

As for the $\mathcal{P}_1/\mathcal{P}_1$ method, we notice that the components of the velocity are quite similar to the exact solution, with a residual of about 0.45 for the first mesh and then approximately halved at each refinement. On the contrary, the result for the pressure is quite rough, and worsens as the meshes get refined. This is due to the appearance of oscillations, whose effect becomes more and more evident as the stepsize is reduced. This behavior fully agrees with what expected from the theory: the method does not satisfy the inf-sup conditions and hence is not stable.

The problem of oscillations is expected to be solved in the case of the GLS$-\mathcal{P}_1/\mathcal{P}_1$ method, due to the stabilizing effect of the extra term introduced in the model. The numerical results confirm what expected: the residuals, including the one of the pressure, are all reduced as the meshes grow more refined.

Finally, the results for the $\mathcal{P}_1$ bubble /$\mathcal{P}_1$ method show the same behavior, with decreasing residuals as the meshes grow refined, confirming the stability property expected from the theory.

As a further comment, we may also notice that in all cases the total residual error is mainly due to the error on the pressure, which, even in the stable methods, is about forty times

(GLS$-\mathcal{P}_1/\mathcal{P}_1$) or seventy times ($\mathcal{P}_1$ bubble $/\mathcal{P}_1$) the error on the velocities. In this sense, by comparing the total residual errors, we observe that the GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$ seems preferable. This is due to the fact that, by modifying the stabilization coefficient δ we can adjust the amount of stabilization - and therefore obtain smaller residuals - according to our needs.

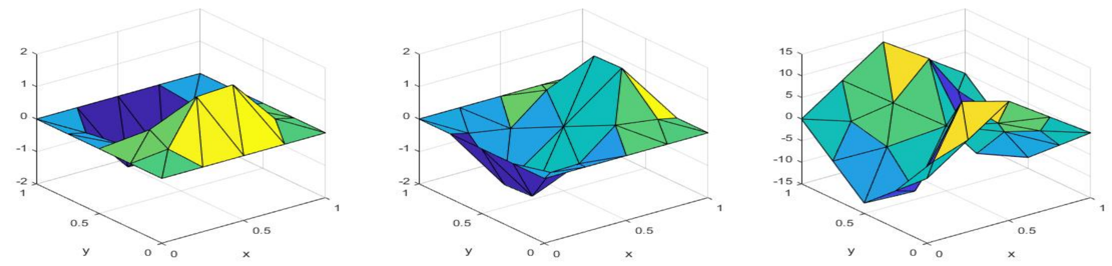
5.2 Lid-Driven Cavity Problem

We're now interested in solving the so called Lid-Driven Cavity problem. It is sufficient to choose a stable method and apply it, with proper adjustments on Dirichlet boundary conditions, forcing function and divergence function. In our case, we repeated the procedure using both the GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$ method and the $\mathcal{P}_1$ bubble $/\mathcal{P}_1$ method. The numerical results are plotted by components (u_1, u_2, p) in figure 5.

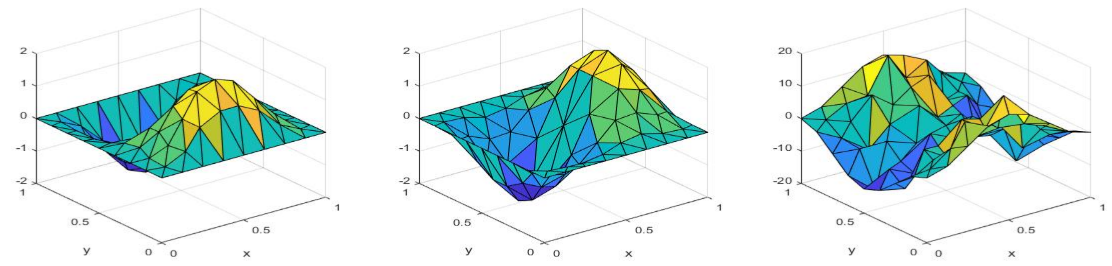
The results seem to be comparable, yielding approximately the same numerical solution. The main difference lies in the pressure, which in both cases consists in a large flat region with two peaks at the points $(0, 1)$ and $(1, 1)$. The height of these peaks is slightly bigger in absolute value in the case of the $\mathcal{P}_1$ bubble $/\mathcal{P}_1$ method. This difference is probably due to the fact that we're trying to represent a local behavior through a discretized system, whose precision is necessarily limited, especially when high gradients are involved. In particular, if we use more and more refined meshes we observe that the height of the peaks increases in absolute value, showing the tendency to diverge.

Figure 1: $\mathcal{P}_1/\mathcal{P}_1$ method - numerical solution by components, with progressively refined meshes.

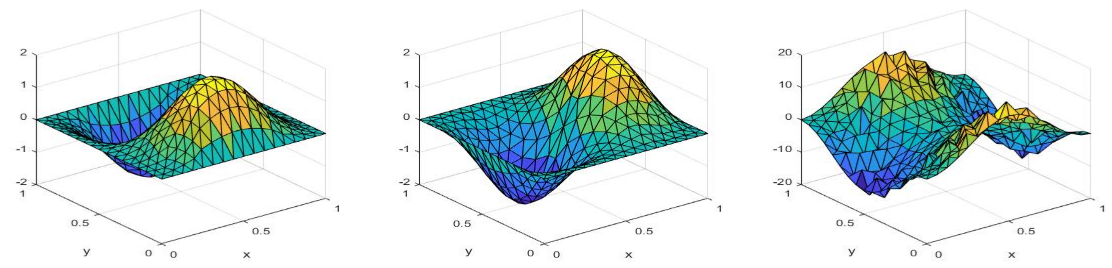
(a) Below: Mesh 0 - from left to right: u_1, u_2, p



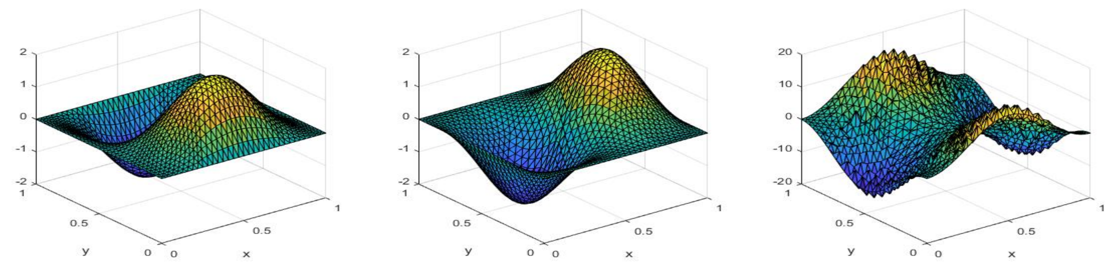
(b) Below: Mesh 1 - from left to right: u_1, u_2, p



(c) Below: Mesh 2 - from left to right: u_1, u_2, p



(d) Below: Mesh 3 - from left to right: u_1, u_2, p



(e) Below: Mesh 4 - from left to right: u_1, u_2, p

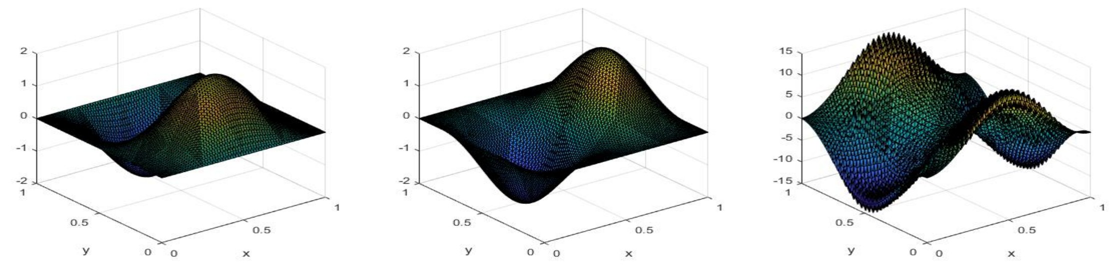
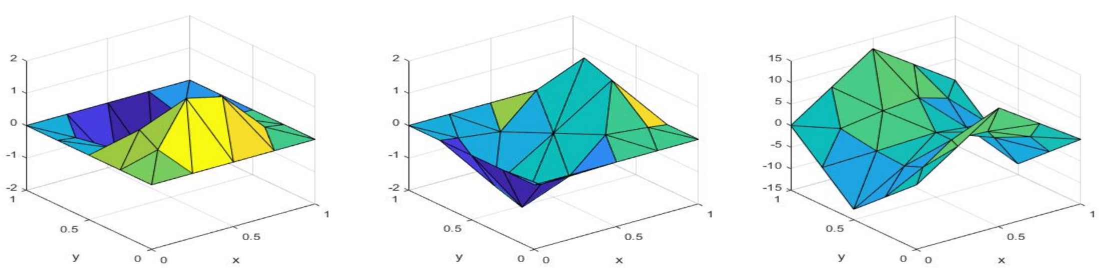
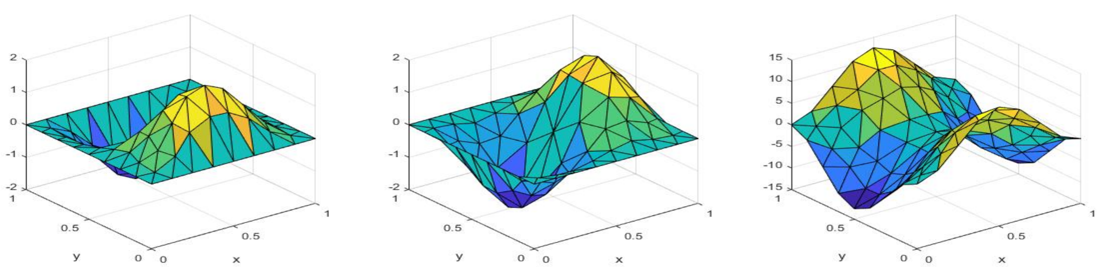


Figure 2: GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$ method - numerical solution by components, with progressively refined meshes.

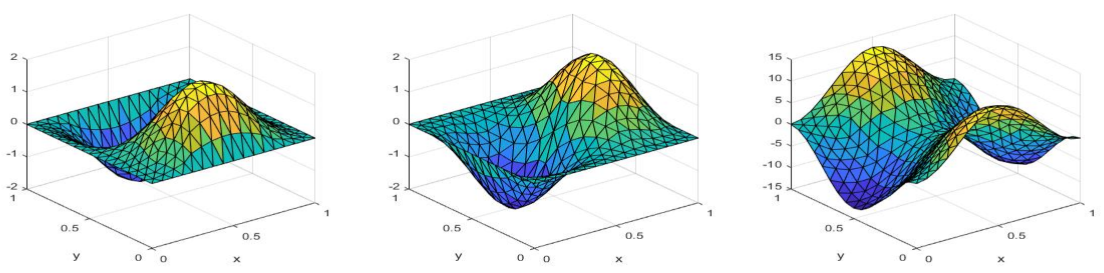
(a) Below: Mesh 0 - from left to right: u_1, u_2, p



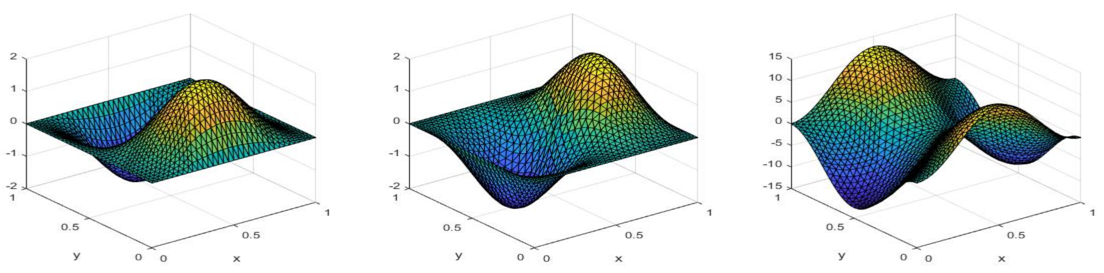
(b) Below: Mesh 1 - from left to right: u_1, u_2, p



(c) Below: Mesh 2 - from left to right: u_1, u_2, p



(d) Below: Mesh 3 - from left to right: u_1, u_2, p



(e) Below: Mesh 4 - from left to right: u_1, u_2, p

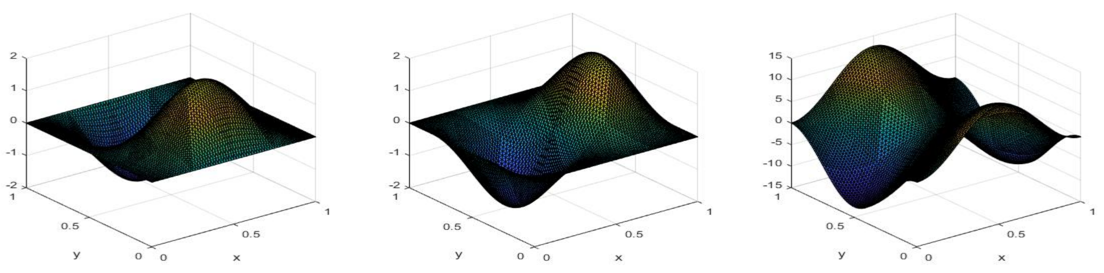
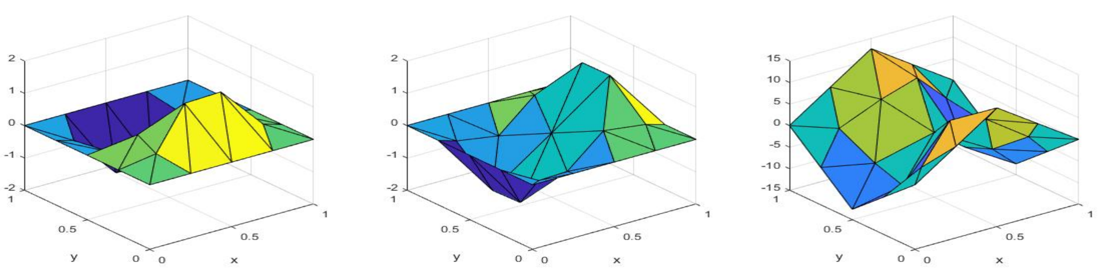
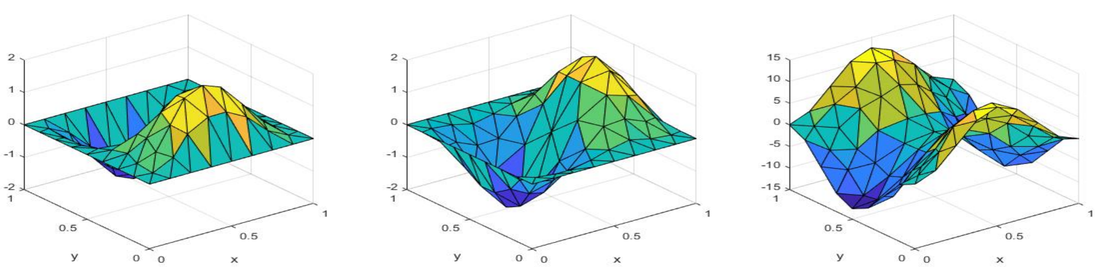


Figure 3: $\mathcal{P}_1$ bubble $/\mathcal{P}_1$ method - numerical solution by components, with progressively refined meshes.

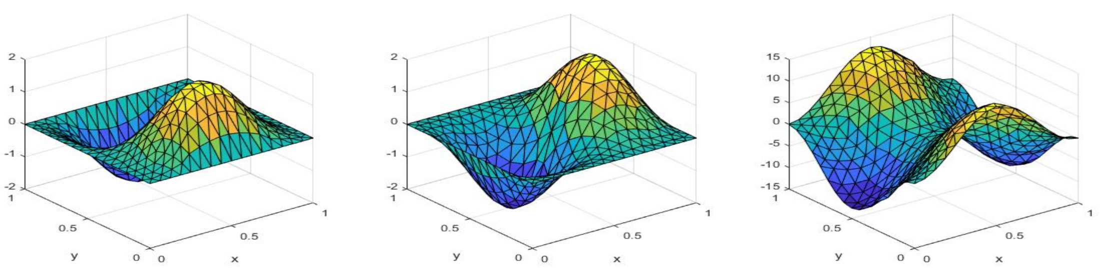
(a) Below: Mesh 0 - from left to right: u_1, u_2, p



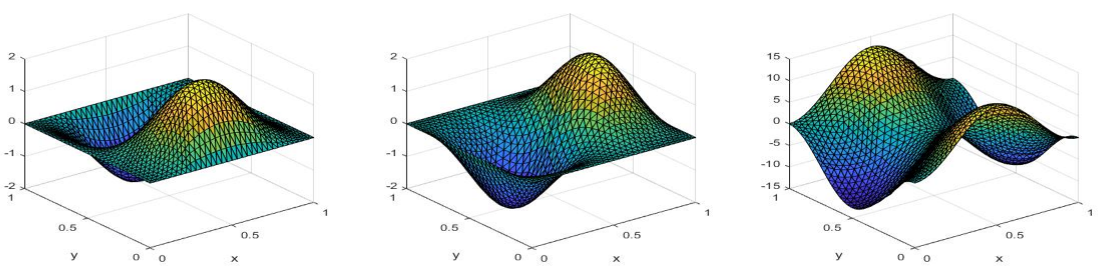
(b) Below: Mesh 1 - from left to right: u_1, u_2, p



(c) Below: Mesh 2 - from left to right: u_1, u_2, p



(d) Below: Mesh 3 - from left to right: u_1, u_2, p



(e) Below: Mesh 4 - from left to right: u_1, u_2, p

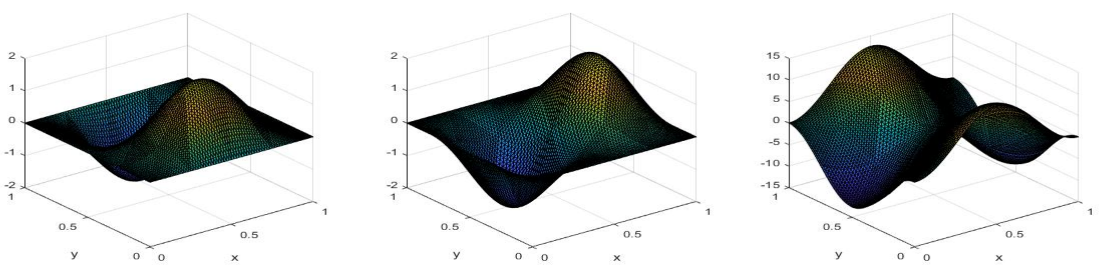
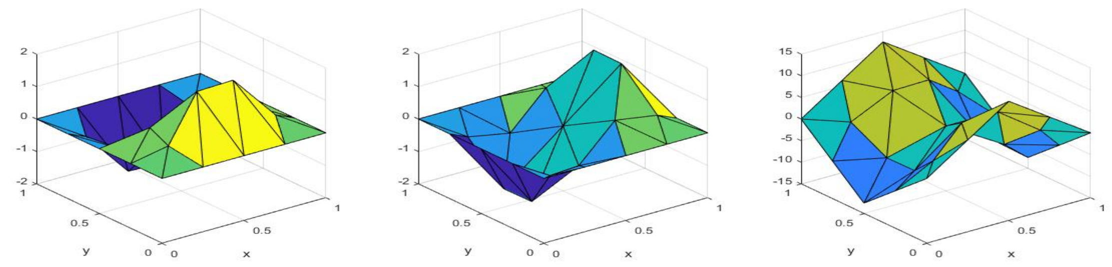
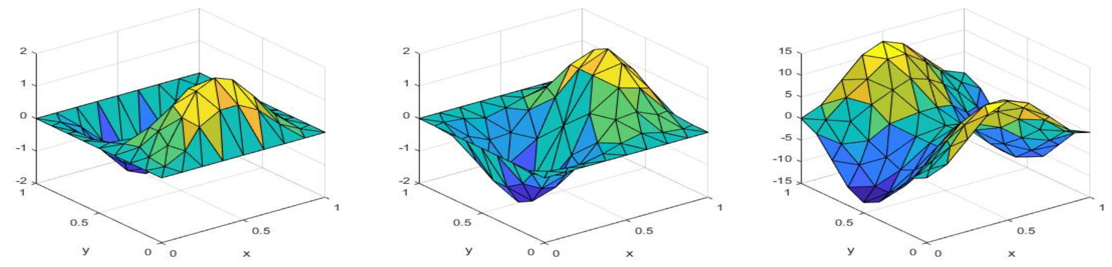


Figure 4: Exact solution by components, with progressively refined meshes.

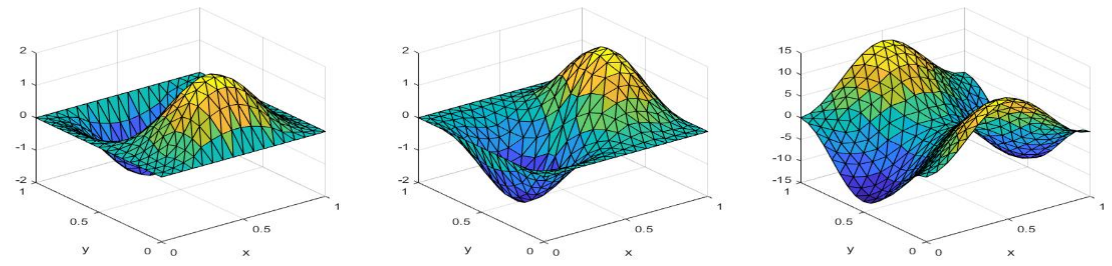
(a) Below: Mesh 0 - from left to right: u_1, u_2, p



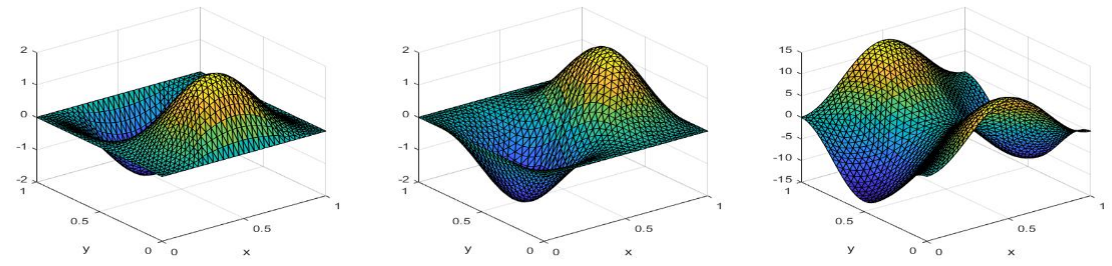
(b) Below: Mesh 1 - from left to right: u_1, u_2, p



(c) Below: Mesh 2 - from left to right: u_1, u_2, p



(d) Below: Mesh 3 - from left to right: u_1, u_2, p



(e) Below: Mesh 4 - from left to right: u_1, u_2, p

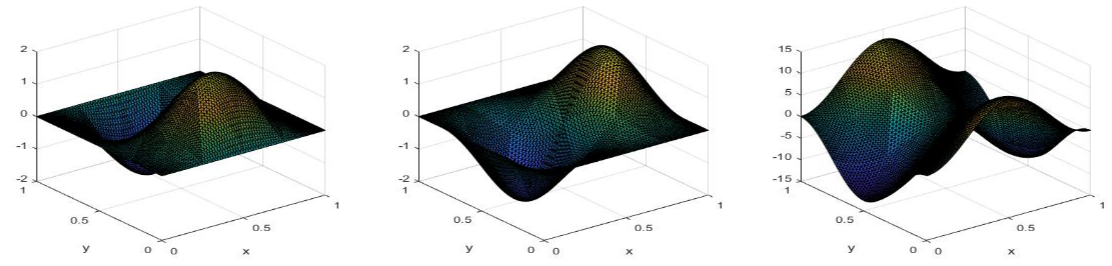
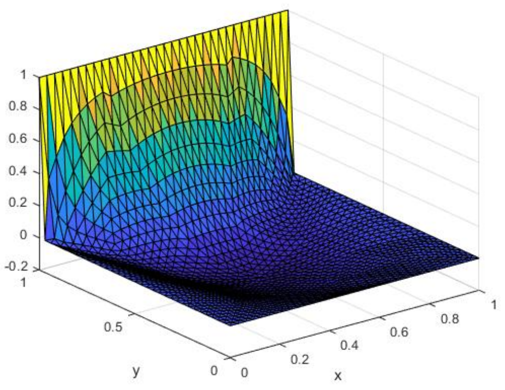
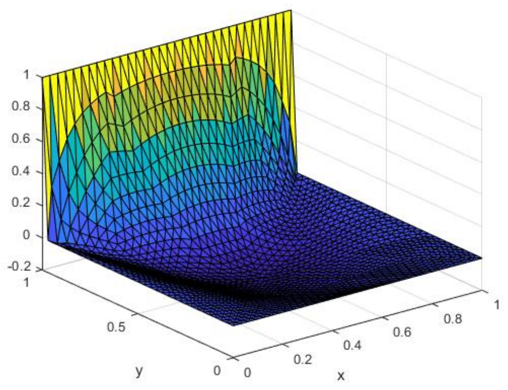


Figure 5: Lid-Driven Cavity Problem. On the left: GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$ method. On the right: $\mathcal{P}_1$ bubble $/\mathcal{P}_1$ method

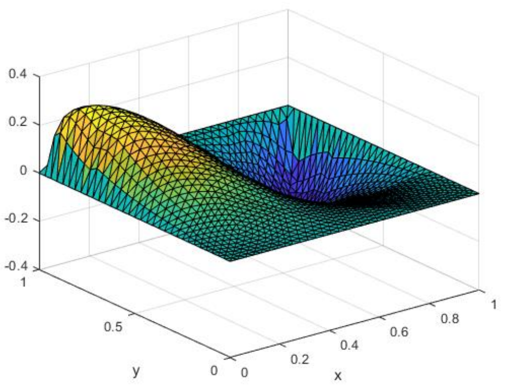
(a) GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$: component u_1



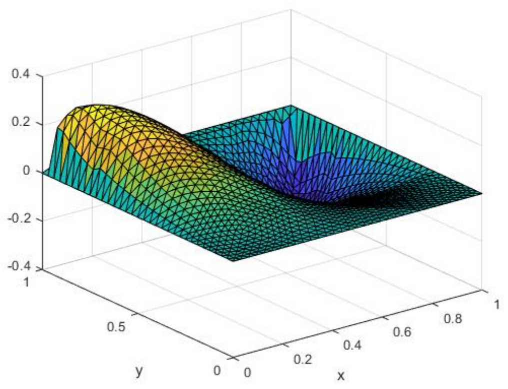
(b) $\mathcal{P}_1$ bubble $/\mathcal{P}_1$: component u_1



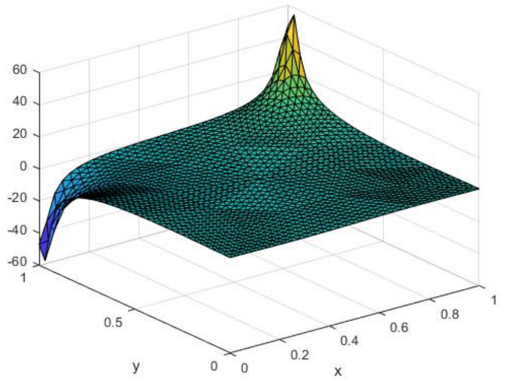
(c) GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$: component u_2



(d) $\mathcal{P}_1$ bubble $/\mathcal{P}_1$: component u_2



(e) GLS stabilized$-\mathcal{P}_1/\mathcal{P}_1$: component p



(f) $\mathcal{P}_1$ bubble $/\mathcal{P}_1$: component p