Laplace equation and flow in porous media

Pol Navarro Pérez

September 2024

Abstract

In this study, we solve the two-dimensional Laplace equation on a square domain. Given specific boundary conditions and a source term, triangular and quadrilateral elements of different degrees have been applied to the weak form of the equation. It is clearly observed that the error decreases when smaller element sizes are used and when the polynomial degree of the elements is increased.

On the other hand, we have analyzed the movement of water in a porous media. By applying Dirichlet and Neumann boundary conditions, we solve the resulting system of equations to compute the water flow across an excavation surface. The results show minimal differences between the Lagrange multiplier method and the system reduction approach. Additionally, we observe convergence as the mesh size is refined, and varying the distance of the artificial boundary wall significantly impacts the distribution of the piezometric level.

1. Laplace equation in 2D

Our starting point will be a two-dimensional domain $\Omega = [0,1]^2$ with given homogeneous Dirichlet boundary conditions:

$$-\Delta u = f \quad \text{in } \Omega,\ u = 0 \text{ on } \partial\Omega \tag{1}$$

with a source term f,

$$f(x,y) = -\frac{1}{2000}(g''(x)g(y) + g(x)g''(y)), \quad g(p) = p^2(1-p)^2e^{10p} \tag{2}$$

Developing the derivatives from g(p), using the chain rule:

$$g'(p) = e^{10p}[10x^4 - 16x^3 + 4x^2 + 2x]$$

$$g''(p) = e^{10p}[100x^4 - 120x^3 - 8x^2 + 28x + 2]$$

a) Weak form and discretization

First of all, we are required to obtain the weak form. The first step is to multiply by a test function v such that $v = 0$ in Γ_D.

$$-\int_\Omega v\nabla\cdot(\nabla u)d\Omega = \int_\Omega vfd\Omega \tag{3}$$

The next step is integrating by parts, considering we are in a 2 dimension space.

$$\int_\Omega \nabla u\nabla v d\Omega - \int_{\partial\Omega} v\nabla u\cdot\vec{n}dS = \int_\Omega vfd\Omega \tag{4}$$

where $v = 0$ on Γ_D and $\Gamma_N = \emptyset$. Therefore, the problem is now to find $u \in H^1$ such that $u = u_D$ on $\Omega_D = \partial\Omega$ and

$$\int_\Omega \nabla u\nabla v d\Omega = \int_\Omega vfd\Omega \tag{5}$$

$\forall\ v \in H^1$ such that $v(\Omega_N) = 0$.
Next part consists on the discretization of the weak form with $u \approx u^h(x) = \sum_{j=1}^N u_jN_j$, where $N_j(x_i) = \delta_{ij}$, and $v(x) = N_i$, $i \notin \Gamma_D$.
Our linear systems of equations is then $Ku = f$ with:

$$K_{ij} = \int_\Omega \nabla N_i\nabla N_j d\Omega \tag{6}$$

and

$$f_i = \int_\Omega N_i f d\Omega \tag{7}$$

Remark that the Dirichlet boundary conditions have not been applied yet.

b) Code Laplace in 2D

In this section we are going to work with a provided Python code called `main_Laplace2D`.
The code is structured in three main parts: preprocessing, computation and post-process.

Preprocessing

The first step is to define "where" are we going to work. The code defines the bidimensional array $\Omega = [0,1]^2$ and the reference element. In this case we use a `degree` $= 1$, which means we are gonna use a linear polynomial in a `elementType` $= 1$, triangles. This is implemented calling the function `defineRefElement`, which defines the main features of the elements, such as the nodal functions, the local nodal numbers, the integration functions and their weights.
Given our Ω domain we decompose this in a 8x8 mesh and implement the function `rectangleMesh`. The function generates a mesh grid of nodes for a rectangular domain and establishes how those nodes are connected to form finite elements based on the type specified. With this we get the connectivity matrix T and the nodes coordinates matrix X. Using these we can assembly the system of our FEM $Ku = f$.

Computation

In this part of the code we use the function `computeSystemLaplace`, in order to obtain the global system of matrices. This is done using a function `elementMatrices`, which calculates the elemental matrix for each element and the matrix of independent terms, using the Gauss-Legendre quadrature method.
Another key point of this section is to define which nodes are on the boundary of our domain. Then, using the exact solution obtained with `exactSol`, we assign the corresponding values to these boundary nodes. Eventually, the finite element method (FEM) solution for the non-boundary nodes is obtained using the function `findSolution_SystemReduction`, which incorporates the system and the Dirichlet boundary conditions. This functions reduce the systems by removing the corresponding rows and columns of the known values.

Post-processing

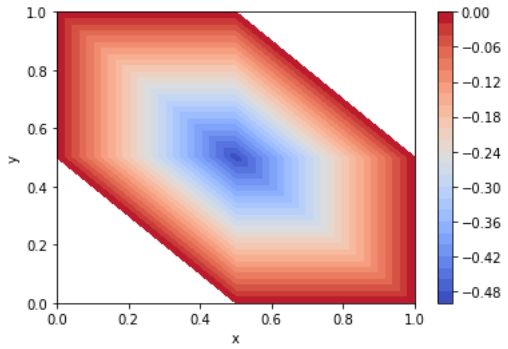
Once we have applied a method to solve a problem, we need to know how far we have deviated from the exact solution. The function `computeL2Error` takes as input the numerical solution u, the coordinates of the nodes X, the connectivity matrix T, and a reference element `refElement`, which includes information about the integration points and basis functions. For each element in the mesh, the Jacobian J is evaluated to transform coordinates from the reference space to the physical space. At each integration point, both the exact solution u_{ex} and the numerical solution u_{num} are evaluated, and the squared error $(u_{\text{num}} - u_{\text{ex}})^2$ is determined. The L^2 error is defined as the square root of the integral of these squared errors over the entire domain, formally given by:

$$L^2 = \sqrt{\int_\Omega |u_{\text{num}}(x) - u_{\text{ex}}(x)|^2 \, dx}$$

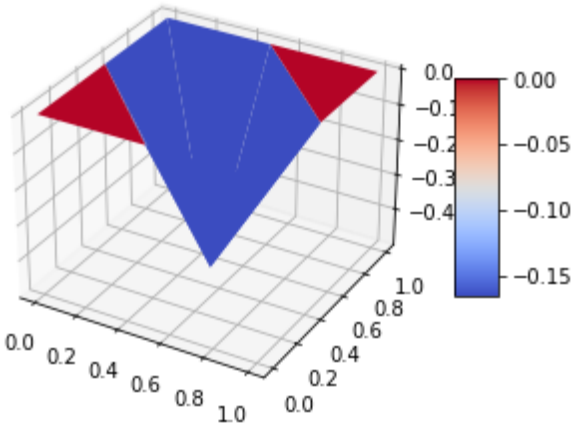
At this point we can use the function `plotSlopes` just to plot the value of a slope between two points. This is going to be very helpfully in the following sections.

c) Solution for P1 elements and size h = $\frac{1}{2}$

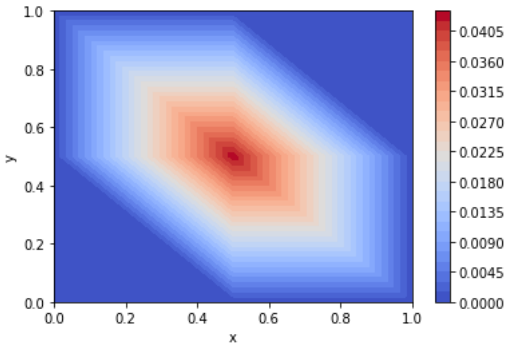
Now we implement the FEM for triangular elements with element size $h = \frac{1}{2}$.



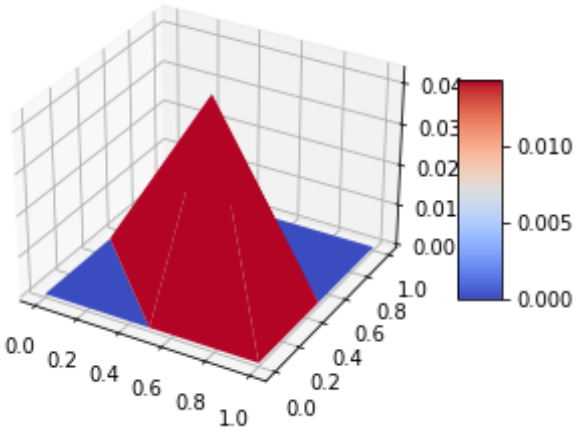
2D FEM



3D FEM



2D analytical



3D analytical

Figure 1: FEM method and analytical solutions for $h = \frac{1}{2}$ and P1 elements.

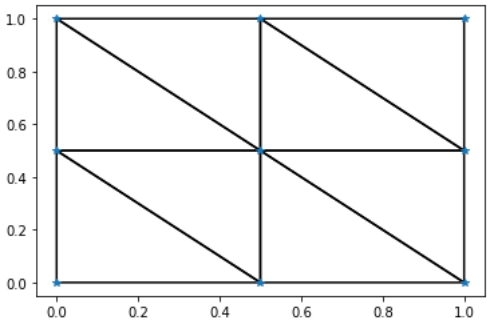
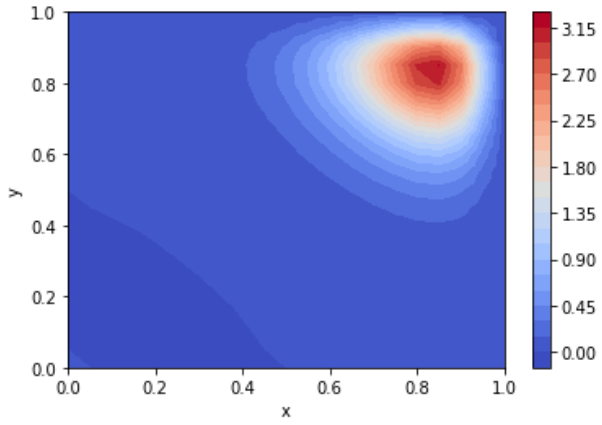


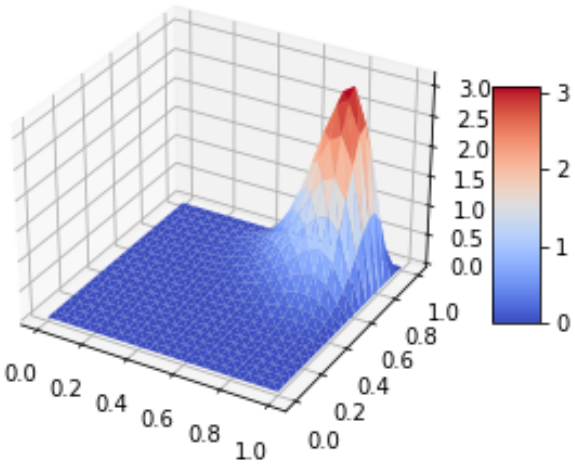
Figure 2: Mesh used for linear triangular elements of size h = $\frac{1}{2}$

.

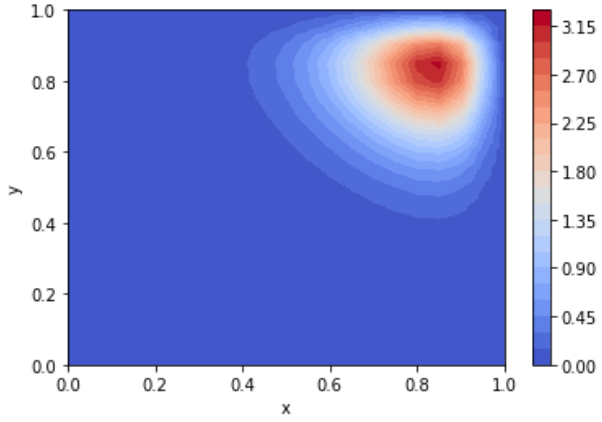
As we can observe in 1 the results are far from being accurate. This is because we have not enough nodes to adapt the discretization to the real behaviour of the system. If we want to show a more accurate FEM solution we can use a shorter element size.



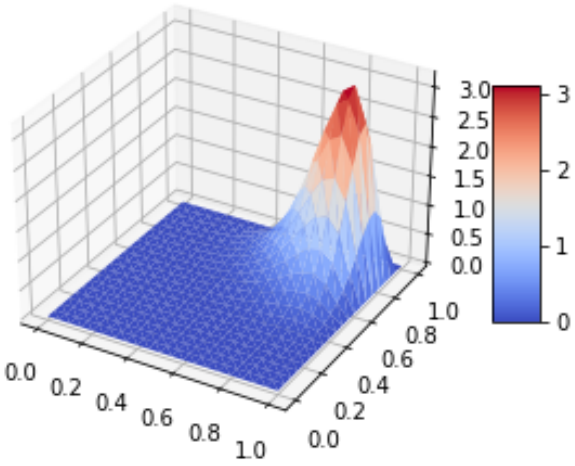
2D FEM



3D FEM



2D analytical



3D analytical

Figure 3: FEM method and analytical solutions for $h = \frac{1}{20}$.

As we can see in 3 the results are much more precise since our method captures the finer details of the local behaviour.

d) Sparsity pattern

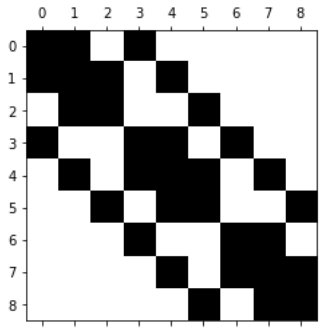


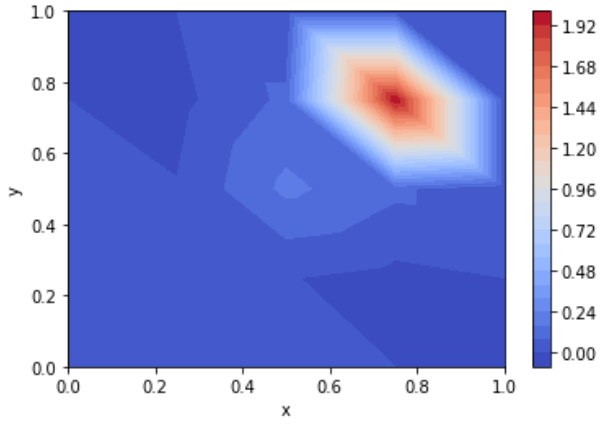
Figure 4: Sparsity pattern of the system matrix before applying boundary conditions, with element size h = $\frac{1}{2}$

.

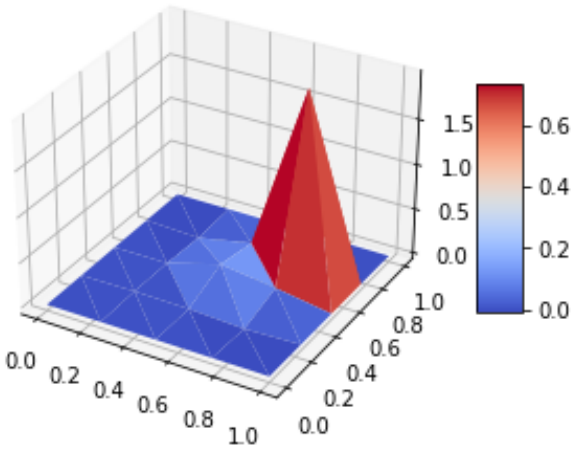
The Figure 4 shows the sparsity pattern. Therefore we can obtain the information about which nodes are connected. This can be one by one interpreted looking at the nodes of 2 and then considering which nodes are connected using the correct node orientation. For instance, the bottom-left node is connected with 2 nodes and itself, as we can see in the sparsity pattern.

e) Quadratic interpolation (Q2).

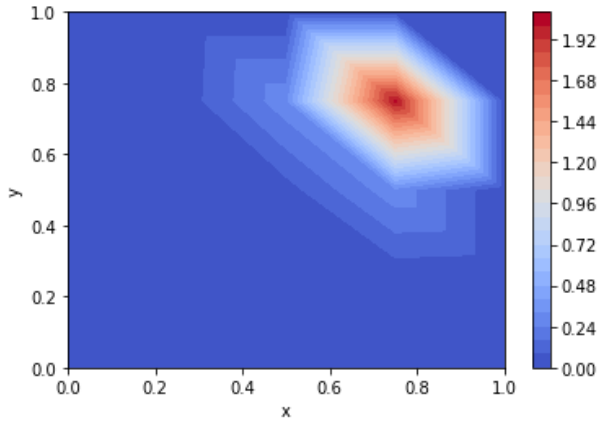
Now we have to evaluate the FEM using quadratic interpolation elements (Q2). Each element has 9 nodes and $5^2 = 25$ gauss-legendre integration points. The nodes are anticlockwise oriented, with the center as the last node. Remark that the element size is still $h = \frac{1}{2}$.
Modifying the function `defineRefElement` we have implemented the Q2 elements defining the shape functions and their corresponding derivatives.



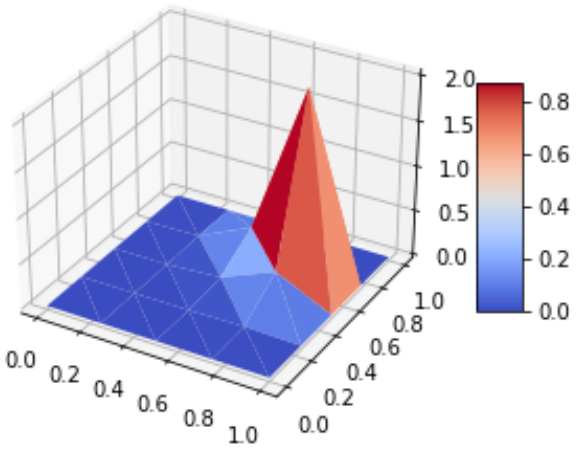
2D FEM



3D FEM



2D analytical



3D analytical

Figure 5: FEM method and analytical solutions for Q2 elements and element size $h = \frac{1}{2}$.

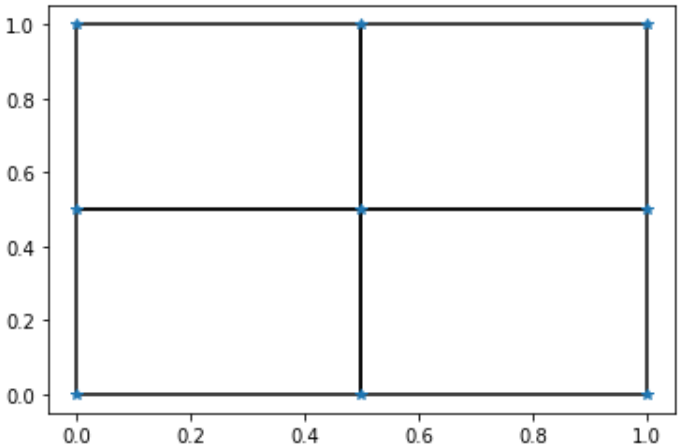


Figure 6: Mesh for quadratic elements.

.

As it is clearly observable in 5, the FEM approximation is more accurate than other linear elements, considering the same value of the element size h.

These Q2 shape functions are defined for each of the 9 nodes, including their corresponding derivatives with respect to the local coordinates ξ and η, which are crucial for computing the stiffness matrix.
The use of 25 Gauss-Legendre integration points allows for the precise evaluation of integrals over the elements, especially when dealing with quadratic terms in the shape functions. This guarantees the accuracy of the finite element formulation when assembling the global system matrix and force vector.

e) Errors for Q1 and Q2.

In this last section of the first exercise we are going to evaluate our method. In order to achieve that, we need to find the errors, the FEM compared to the analytical solutions. Specifically, we have worked with elements Q1 and Q2.
The errors have been computed according to:

$$L^2 = \int_\Omega |u - u^h|^2 d\Omega \tag{8}$$

$$H^1 = \int_\Omega |u - u^h|^2 d\Omega + \left(\int_\Omega ||\Delta u - \Delta u^h|| \right)^{\frac{1}{2}} \quad H^1 = \int_\Omega |u - u^h|^2 d\Omega + \left(\int_\Omega ||\nabla u - \nabla u^h|| \right)^{\frac{1}{2}} \tag{9}$$

As seen in theory sessions these errors should be boundered according to:

$$\|u_h - u\|_{L^2} \leq Ch^{p+1} \tag{10}$$

$$\|u_h - u\|_{H^1} \leq Ch^p \tag{11}$$

where C is an arbitrary constant, h is the element size and p the degree of the nodal functions.
As we have seen and already know, decreasing the size of the element size h should reduce the error computed. In addition, by definition of the errors, the error in H^1 should be higher than in L_2.

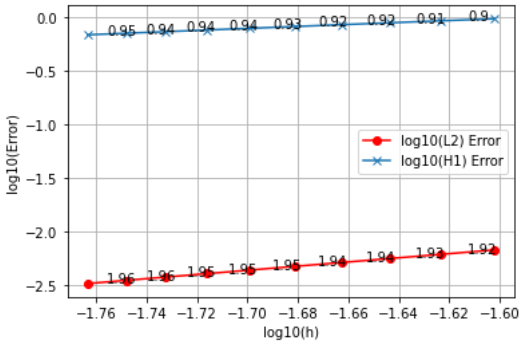


Figure 7: Errors in L_2 and H_1 for linear quadratic elements Q1.

.

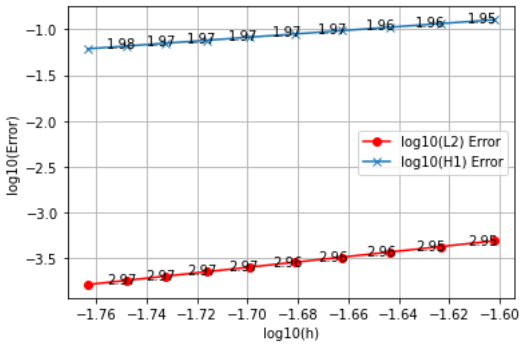


Figure 8: Errors in L_2 and H_1 for elements quadratic quadrilateral elements Q2.

.

As we can see in 7 and 8, the error converges as the element size decreases. We also observe, using the `plotSlopes` function, how the error, both for L_2 and H_1, is bounded according to the previous formula, with different values depending on the polynomial degree. In the previous graphs less values of element size have been taken in order to graphically see the convergence of the error. Straightaway we work with more values of h, working with smaller ones, to compute the regression of our error results.

Degree	Error	Slope
Q1	L_2	2.0077
	H_1	1.0055
Q2	L_2	2.9719
	H_1	1.9746

Table 1: Slopes for the regression of the logarithm errors as a function of the logarithm element size. Lengths ranging from $\frac{1}{40}$ to $\frac{1}{80}$ have been used.

As we can see on Table 1 the errors are getting quite close to the expected values of 10 and 11, considering it is a ln-ln plot.
We have used the previously mentioned function `plotSlopes` to clearly see how the error is converging as we decrease the element size. Additionally, linear regression has been performed, this time selecting much smaller values of h, to obtain the slope.

2. Porus media

Introduction

The objective of this exercise is to analyze the 2D movement of water in an orthotropic porous medium, governed by the equation that imposes an incompressible flow, $\nabla \cdot \mathbf{q} = 0$ in the domain Ω. For a laminar steady flow and fully saturated soil, the water flow, $\mathbf{q}$, is described by Darcy's law: $\mathbf{q} = -K\nabla u$, where u represents the piezometric level, which is the sum of the pressure head and elevation head, describing the potential energy of water at a given point in a porous medium.K is the hydraulic conductivity matrix of the orthotropic medium.

In this specific case, the conductivity matrix is given by:

$$K = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix}$$

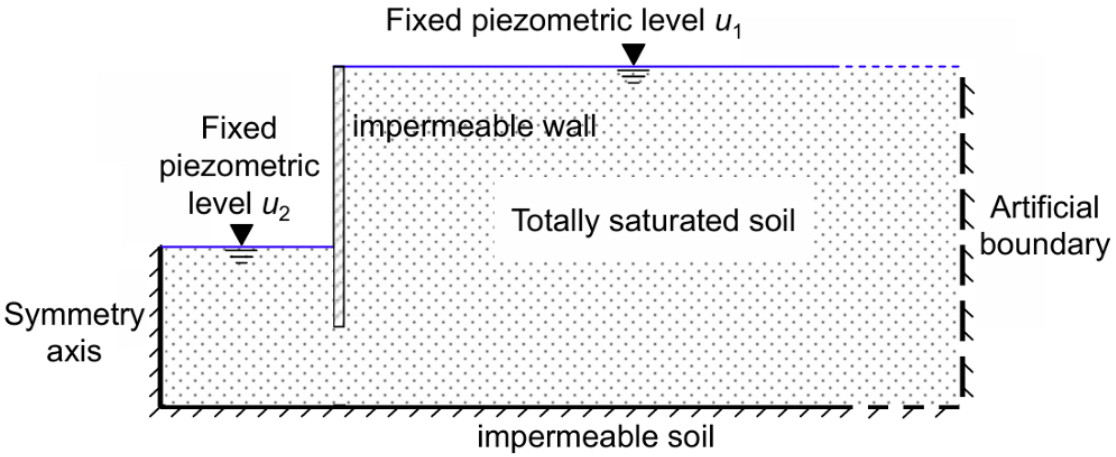


Figure 9: Excavation scheme.

.

The problem geometry is based on an excavation scenario, as shown in Figure 9, where symmetry and artificial boundaries are used to limit the computational domain. The goal of this exercise is to compute the water flow emerging from the excavation surface.

We consider a clay soil with constant hydraulic conductivities $K_1 = 10^{-6}$ m/s and $K_2 = 10^{-7}$ m/s, alongside specific boundary conditions. These include no flux on the symmetry, artificial, and impermeable boundaries, and a fixed piezometric level on the remaining boundaries. The problem involves the use of different structured meshes generated by the function `ExcavationMeshes.py`, with both Q1 and Q2 node distributions and varying element sizes.

a) Strong form and boundary conditions

In the first section we are going to use the equations known of our system to get:

$$\nabla \cdot (-K\nabla u) = 0 \tag{12}$$

And there is another information to use, the boundary condition of $q \cdot n = 0$ on the symmetry, artificial and impermeable boundaries. This means the flux is not going through theses boundaries. In addition, we know that there are boundaries where the piezometric level is fixed $u(x, y) = h(y)$, with the height $h(y) = y$. Hence, we can compute the strong form as:

$$\begin{cases} \nabla \cdot (-K\nabla u) = 0 & \text{on } \Omega, \\ -K\nabla \cdot u\vec{n} = 0 & \text{on } \Gamma_N, \\ u(x, y) = u_D = h(y) = y & \text{on } \Gamma_D \end{cases} \tag{13}$$

$$\tag{14}$$

where Γ_N are the symmetry, artificial and impermeable boundaries; and Γ_D are the points on the excavated part and on the surface not excavated. We also know that $\Gamma_N \cup \Gamma_D$ is all the boundary domain $\partial\Omega$.

b) Weak form and system matrix

The next step is the weak form and obtain the linear systems from the FEM approximation of u.
Starting from 3 we can get:

$$\int_\Omega v\nabla \cdot (-K\nabla u)d\Omega = 0 \tag{15}$$

and using the boundary conditions:

$$\int_\Omega \vec{\nabla} v \cdot (-K\vec{\nabla} u)d\Omega - \int_{\partial\Omega} v(-K\vec{\nabla} u)\vec{n}dS = 0 \tag{16}$$

But there are two boundary conditions, the Dirichlet and Newman, hence:

$$\int_{\partial\Omega} v(-K\vec{\nabla} u)\vec{n}dS = \int_{\Omega_D} v(-K\vec{\nabla} u)\vec{n}dS + \int_{\Omega_N} v\vec{n} \cdot \vec{q}dS = 0 \tag{17}$$

Therefore, the problem is now to find $u \in H^1$ such that $u = u_D$ on Ω_D and

$$\int_\Omega \vec{\nabla} v(-K\vec{\nabla} u)d\Omega = 0 \tag{18}$$

$\forall\ v \in H^1$ such that $v = 0$ on Ω_N.
Next part consists on the discretization of the weak form with $u \approx u^h(x) = \sum_{j=1}^{N} u_j N_j$, where $N_j(x_i) = \delta_{ij}$, and $v(x)$ $= N_i$, $i \notin \Gamma_D$.
Our linear systems of equations is then $Ku = f$ with:

$$K_{ij} = \int_\Omega -K\nabla N_i \cdot \nabla N_j d\Omega \tag{19}$$

$$f_i = 0 \tag{20}$$

Eventually we have to solve the system $\mathbf{K}\vec{u} = f_i = 0$. Remark that the Dirichlet boundary conditions have not been applied yet.

c) Solution of the weak form

The next step is to solve the above weak form using the function `main_flowPorousMedia` and to impose the Dirichlet boundary conditions.
To solve this new problem we are going to modify the code from problem 1. We have to readjust the weak form from the function `elementMatrices` adding the matrix K and removing the source term. Then the new Dirichlet boundary conditions have to be applied, in this case the two know values of the piezometric levels.

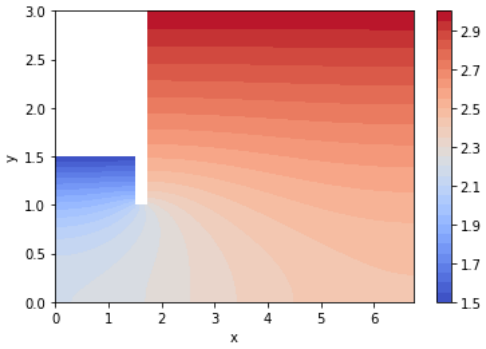


Figure 10: 2D solution of the weak form by systems reductions and FEM, using Q2 elements

.

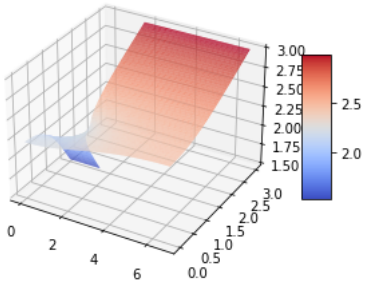


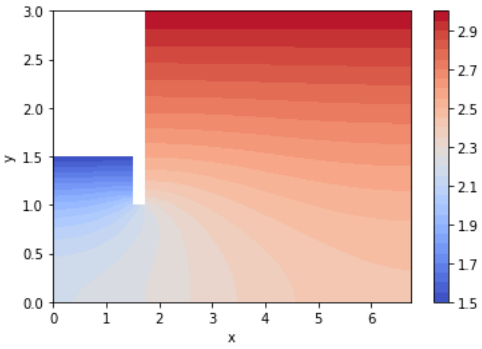
Figure 11: 3D solution of the weak form by systems reductions and FEM, using Q2 elements

.

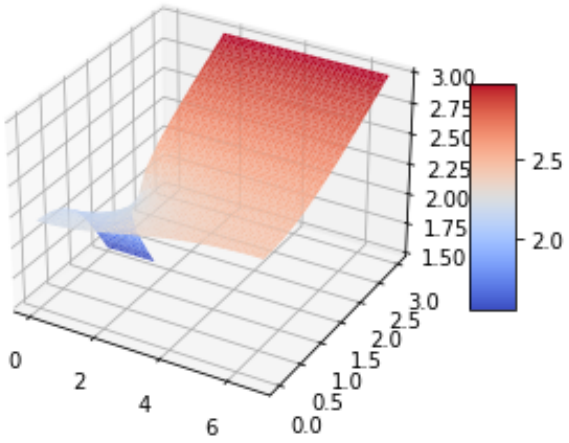
As we can observe, the Dirichlet boundary conditions appear in the image. In addition, if we observe how the piezometric level varies across the domain, it makes physically sense.

d) Piezometric level

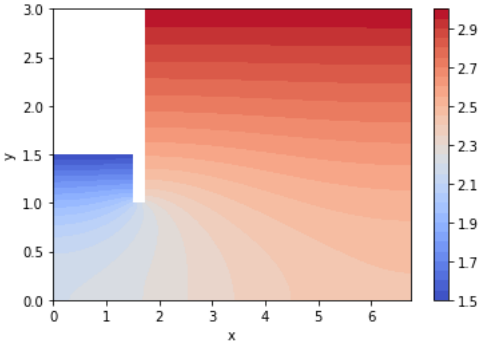
The next step is computing the previous results but using Q1 and Q2 elements, as previously asked in the problem 1. There is also state to try using an artificial boundary at 5m and 10m away from the excavation. This is simply done by changin the parameters from the `defineRefElement` function and the parameter `wallDistance`.



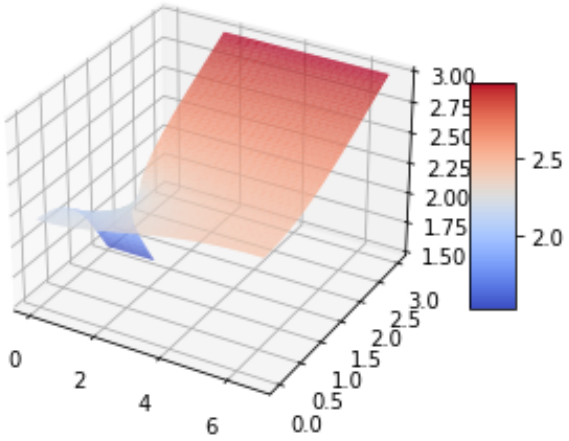
2D Q1



3D Q1



2D Q2



3D Q2

Figure 12: Wall distance 5m comparing Q1 and Q2 elements

As observed in 12 and 13 there is not much difference when comparing the use of Q2 and Q1 elements. The 10m wall distance makes the right part of the excavation more similar to the upper part, the one with piezometric level u_1. This makes the right part more homogeneous for bigger distances.
There is also a clear difference regarding the equipiezometric lines. For the 5m distance, the lines are completely straight, whereas for the 10m distance, they are quite curved.
Considering the curvature of the equipiezometric lines, the 10m boundary provides a more realistic representation of the flow behavior. Thus, I would opt for the 10m distance, as it better captures the natural variations in the piezometric level.

e) Use of Lagrange multipliers

Now we are solving the problem using Lagrange multipliers instead, for a wall distance of 10m.
We start by implementing the functions we need to minimize:

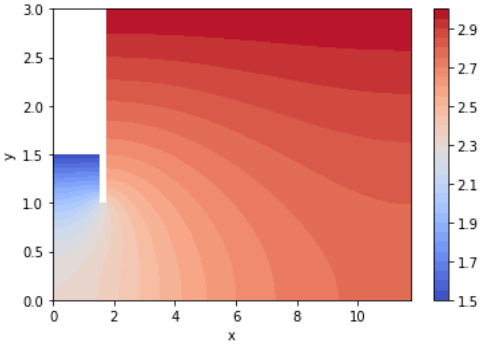
$$\phi(\hat{w}) = \frac{1}{2}\hat{w}^T \mathbf{K} \hat{w} \tag{21}$$

and the minimizing parameter

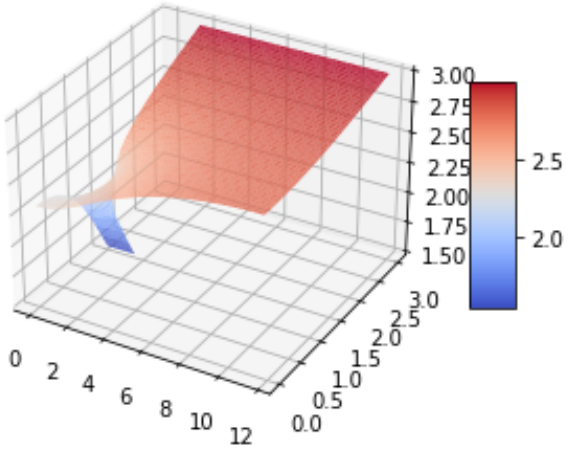
$$u = \underset{w \in H^1(\Omega)}{\operatorname{argmin}}\ \phi(w) \tag{22}$$

such that $w = u_D$ on Ω_D. We are going to use the previously calculated matrix $\mathbf{K}$. Then, using Lagrange multipliers we obtain:

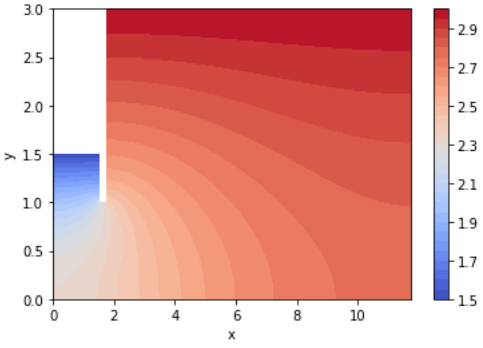
$$(u, \lambda) = \arg\min_{v \in H^1(\Omega)} \max_{\gamma \in H^{-1/2}(\Gamma_d)} \left(\phi(v) + \int_{\Gamma_d} \gamma(v - u_d)\, d\Gamma \right) \tag{23}$$



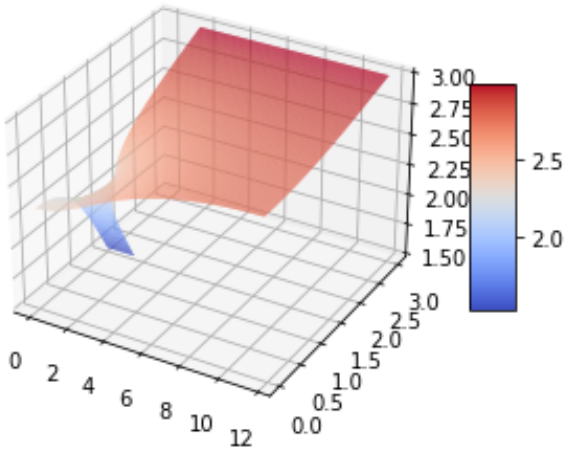
2D Q1



3D Q1



2D Q2



3D Q2

Figure 13: Wall distance 10m comparing Q1 and Q2 elements

Remark that the f term is zero in our case, so it does not appear in the previous equation. Physically, λ is the flux through the Dirichlet boundary. Discretized it is computed as:

$$\lambda \approx \sum_{l=1}^{n} \lambda_l \phi_l(x) \quad , \lambda = \phi_k, \quad \text{for} \quad k \in \Gamma_D,$$

we obtain the system to be solved:

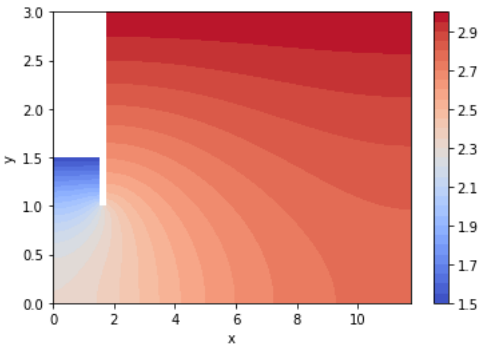
$$\begin{pmatrix} \mathbf{K} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix}$$

where

$$A_{i,j} = \int_{\Gamma_D} \phi_k N_j \, dS, \quad b_i = \int_{\Gamma_D} \phi_k u_D \, dS.$$

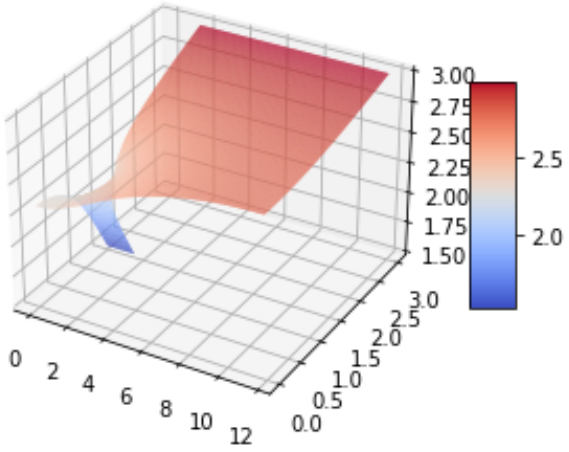
To impose the Dirichlet boundary conditions, we extracted the nodes along the two known boundaries of the domain. These nodes were assigned their corresponding Dirichlet values based on their y-coordinates. Then we constructed the matrix **A**, where each row corresponds to one Dirichlet node, and a 1.0 is placed in the appropriate column for each node.
Finally, the augmented system matrix **K_aug**, the left matrix of our system, was built by combining **K** and **A**, and solved using `np.linalg.solve`. The resulting solution, `sol_aug`, provided the field variable **u** and the Lagrange multipliers λ.

As we can see in 14, the results for a wall distance of 10m are quite similar to those in 13. We can therefore conclude that, for our boundary conditions and the system geometry, both methods; system reduction and Lagrange multipliers, give similar results.



2D

Lagrange multipliers



3D Lagrange multipliers

Figure 14: Comparison of 2D and 3D solutions using Lagrange multipliers.

f) Flow on the excavation surface

The final part is about computing the flow on the excavation surface. As seen in theory sessions the Lagrange multipliers have some physical values, in this case the desired water flow. Then, we just have to sum the Lagrange values for the nodes on the excavation surface. This is done by taking the vector of λ and getting the ones with the same indices than the nodes of the excavation surface. The flow obtained summing these values is $-1.5495119008008559 \cdot 10^{-7}$. We assume the negative sign physically means water is being extracted from the system. Otherwise, we would be introducing fluid trough the medium.

There is a final aspect to consider, the influence of the mesh size on the value of the flow. In order to study this dependence, we just iterate through different values of fineness and plot the graphic between the flow on the excavation surface and the mesh size.

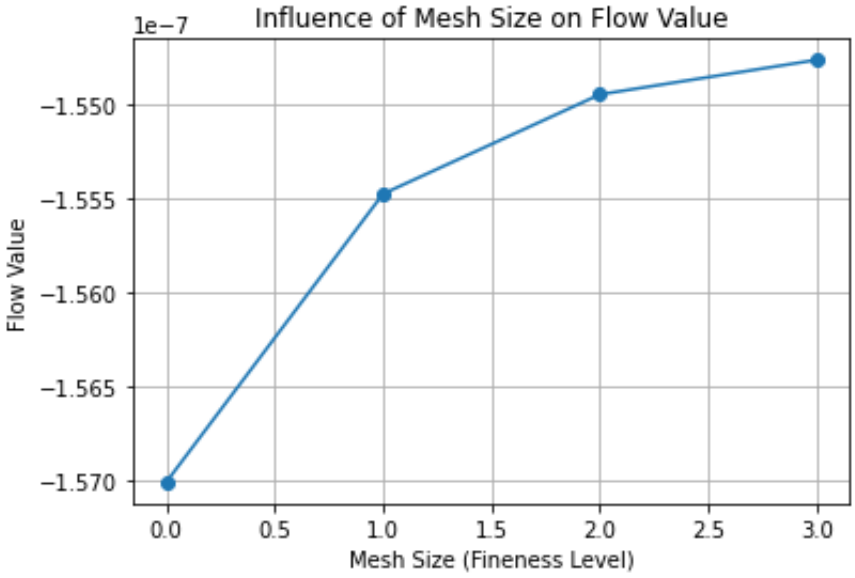


Figure 15: Dependence of the flow computed on the excavation surface and the mesh size

.

As shown in 15, the mesh size significantly influences the computed flow. The flow value appears to converge towards a supposed exact value, with the rate of change decreasing as the mesh becomes finer.