POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

PROJECT REPORT

Permeo

SCIENTIFIC COMPUTING TOOLS FOR ADVANCED MATHEMATICAL MODELLING

Authors: ARIANNA CAGALI, MATTIA GASTOLDI, ANNA P. IZZO AND ROBERTO GASTOLDI

Academic year: 2024-2025

The source code of the entire project is available at:
https://github.com/StefanoPagani/Permeo/tree/CagaliGastoldiIzzo

1. Introduction

A **porous medium** is a material containing pores within its structure. The pore space can be filled with one or more fluids.
In a multidimensional setting, **Darcy's law** describes the flux of a fluid through a porous medium as a linear relation between the flux vector $\boldsymbol{q}$ and the pressure gradient ∇p, modulated by the permeability tensor $\boldsymbol{K}$. Specifically, it reads:

$$\boldsymbol{q} = -\boldsymbol{K}\nabla p,$$

where $\boldsymbol{q} \in \mathbb{R}^d$ is the Darcy velocity, $\boldsymbol{K} \in \mathbb{R}^{d\times d}$ is the symmetric positive-definite permeability tensor, and p is the pressure field.
An important aspect of porous media is the size of pores. It typically spans from 2 to 50 nm, whereas industrial applications deal with scales ranging from centimeters to kilometers. It's impractical, and unnecessary, to seek solutions at the microscopic pore level, making a larger-scale depiction of physical phenomena more fitting. Thus, introducing the idea of a **Representative Elementary Volume (REV)** proves beneficial [1]. The REV method consists in assigning to one mathematical point in space the property (e.g. porosity or permeability) of a certain volume of material surrounding this point, the REV, which is used to define and possibly measure the average properties of the volume considered. Indeed, to describe the phenomena of our interest, we will rely on macroscopic equations that do not need an exact description of the microscopic configuration of the porous medium, but rather, an average of its properties. From now on, we do not distinguish anymore if a point is in a void space or in a matrix, because everything is averaged out over a REV. Thus the macroscopic flow equations will be valid everywhere in our computational domain.
For this problem, the **SPE10** dataset is used. This dataset is a benchmark and is widely used in reservoir simulation to evaluate the performance of numerical methods for subsurface flow. It provides highly heterogeneous and anisotropic permeability fields at fine resolution, derived from realistic geological models. The dataset consists of multiple layers, each representing a 2D horizontal slice of a 3D reservoir, and poses significant challenges for upscaling and numerical solvers due to its strong variability in permeability, which spans several orders of magnitude.
This project aims at developing innovative scientific computing tools for porous media.

2. Permeability upscaling for the Darcy problem

A fine-grid representation of the permeability tensor, while providing high accuracy in capturing local phenomena, proves to be computationally unfeasible for large-scale problems, making its use impractical. For this

reason, it is necessary to implement a function capable of generating an upscaled version of the permeability tensor on a **coarse grid**.

2.1. Mathematical formulation of the problem

When selecting a layer from the SPE10 dataset, the task is to upscale the corresponding anisotropic permeability tensor in $\Omega \subset \mathbb{R}^2$. The upscaling procedure aims to replace the fine-scale model with a coarse one. Combining the law of conservation of mass and the Darcy equation, given the dimensions of the coarse grid, for each subdomain $V \subset \Omega$, the following problems must be solved (in this case we take $V = (0, L_1) \times (0, L_2)$):

$$\begin{cases} \nabla \cdot (-\boldsymbol{K} \nabla p^H) = 0 \quad \text{in } V \\ p^H(0, y) = L_1 \\ p^H(L_1, y) = 0 \\ -\boldsymbol{K} \nabla p^H \cdot \boldsymbol{n} = 0 \quad \text{on } (x, 0) \cup (x, L_2) \end{cases} \qquad \begin{cases} \nabla \cdot (-\boldsymbol{K} \nabla p^V) = 0 \quad \text{in } V \\ p^V(x, 0) = L_2 \\ p^V(x, L_2) = 0 \\ -\boldsymbol{K} \nabla p^V \cdot \boldsymbol{n} = 0 \quad \text{on } (0, y) \cup (L_1, y) \end{cases} \tag{1}$$

Here, p denotes the pressure to which the porous medium is subjected, and $\boldsymbol{K}$ is the fine-scale permeability tensor. To approximate the anisotropy of the field, these systems enforce a horizontal and a vertical pressure gradient, respectively, and can be solved independently. Once the solutions p^H and p^V are computed, they are post-processed to obtain the Darcy velocities, i.e., the fluxes $\boldsymbol{q}^H$ and $\boldsymbol{q}^V$.
The goal of numerical upscaling is to obtain the effective (upscaled) permeability tensor:

$$\boldsymbol{K}^\star \qquad \text{such that} \qquad \langle \boldsymbol{q} \rangle = -\boldsymbol{K}^\star \langle \nabla p \rangle,$$

where $\langle \cdot \rangle$ denotes the average operator over the subdomain. In 2D, this is achieved by solving the following system:

$$\begin{bmatrix} \langle \nabla p^H \rangle_x & \langle \nabla p^H \rangle_y & 0 & 0 \\ 0 & 0 & \langle \nabla p^H \rangle_x & \langle \nabla p^H \rangle_y \\ \langle \nabla p^V \rangle_x & \langle \nabla p^V \rangle_y & 0 & 0 \\ 0 & 0 & \langle \nabla p^V \rangle_x & \langle \nabla p^V \rangle_y \\ 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} K^\star_{xx} \\ K^\star_{xy} \\ K^\star_{yx} \\ K^\star_{yy} \end{bmatrix} = - \begin{bmatrix} \langle q^H \rangle_x \\ \langle q^H \rangle_y \\ \langle q^V \rangle_x \\ \langle q^V \rangle_y \\ 0 \end{bmatrix} \tag{2}$$

The last equation in System (2) is added to ensure that the resulting permeability tensor $\boldsymbol{K}^\star$ is symmetric.
The solution yields the components of the upscaled tensor, while the right-hand side of the equation represents the average flux $\boldsymbol{q}$ within the porous medium.
The upscaled tensor $\boldsymbol{K}^\star$ is constant within each coarse cell and captures fine-scale effects in an averaged manner. As a result, it can be anisotropic, with the alignment of its principal axes reflecting the underlying distribution of high/low permeability in the fine-scale cells.

2.2. Methods

The upscaling procedure begins by reading a selected portion of the SPE10 dataset, focusing on a specific layer. The corresponding fine-scale permeability fields are loaded and the directional components (k_{xx}, k_{yy}) are separated. The fine-scale grid is then partitioned into a fixed number of rectangular subdomains, each of which is processed independently to compute an equivalent coarse permeability tensor. This partitioning sets the stage for a local, subdomain-wise upscaling approach.
Within each subdomain, the effective permeability tensor is computed by solving two flow problems: one with a pressure gradient applied in the horizontal direction (x-axis), and another in the vertical direction (y-axis). These problems are solved under steady-state conditions with Dirichlet boundary conditions on the inflow and outflow boundaries and Neumann conditions elsewhere. To discretize the equations, we opted for the **two-point flux approximation (TPFA)** method. While TPFA is typically limited to orthogonal grids and isotropic media, it was found to produce results in excellent agreement with those obtained via the more accurate **multi-point flux approximation (MPFA)** in our specific setting. Given this equivalence, TPFA was preferred due to its significantly lower computational cost and ease of implementation.
Once the pressure fields are obtained from the TPFA discretization, the mean pressure gradients and fluxes are post-processed. These quantities are used to construct and solve a small linear system that yields the components of the upscaled permeability tensor in **least square** sense.
Finally, the upscaled permeability tensors are assembled over the entire fine grid to form a coarse-scale representation of the medium. This field can be used in subsequent simulations or visualized directly. The process ensures that the coarse model retains the directional flow characteristics induced by the fine-scale heterogeneities while drastically reducing computational complexity.

To accelerate the upscaling process, the parallel computing capabilities of the **`joblib` library** were employed during the most computationally expensive step: the evaluation of the permeability tensors across all subdomains. This part of the workflow involves solving two independent flow problems (one per direction) for each subdomain, and since the subdomains do not interact with each other during this step, the computations are fully decoupled.
This structure makes the problem particularly well-suited for parallelization, as each subdomain can be processed independently without requiring synchronization or communication with the others. Leveraging `joblib`'s `Parallel` and `delayed` constructs allowed for straightforward and efficient distribution of the workload across multiple CPU cores, drastically reducing the overall runtime.
By identifying and isolating this bottleneck, the parallel execution yields a substantial speed-up without compromising the accuracy of the results, making it a practical enhancement in the context of large-scale upscaling procedures.

2.3. Numerical results

Results of the upscaling can be visualized with *ParaView*. The following is an example of fine scale of $\boldsymbol{K_{xx}}$ on the left, and its upscaled version $\boldsymbol{K^*_{xx}}$ on the right for Layer 18.

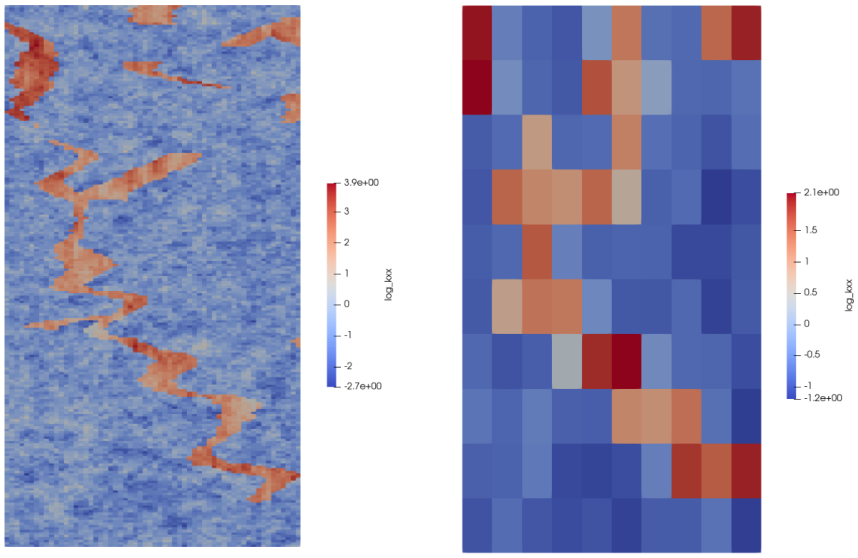


Figure 1: Fine-scale and coarse-scale permeability tensors.

The parallelization drastically speeds up the execution time of the function. An overview of the performances are reported in Table 1.

Average execution times – Sequential Upscaling	
Operation	**Execution Time [s]**
Grid loading and permeability reading	0.09
Grid partitioning	0.07
Upscaling (serial execution)	31.41
Postprocessing and export	0.12
Total	**31.69**

Average execution times – Parallel Upscaling	
Operation	**Execution Time [s]**
Grid loading and permeability reading	0.10
Grid partitioning	0.08
Upscaling (parallel with `joblib`)	16.86
Postprocessing and export	0.15
Total	**17.19**

Table 1: Average execution times of the main components of the upscaling pipeline. These results are obtained by upscaling Layer 18 to 100 subdomains.

The execution times were estimated by repeatedly running the function on the same laptop[1], showing a time saving of approximately 50% by parallelizing only the most expensive operation. This estimate strongly depends on several factors, such as CPU, RAM, and operating system, and may vary slightly from case to case.

3. Optimal well disposition for the Darcy problem

On each side of the boundary $\partial\Omega$ of the domain, a production (or extraction) well is located. These wells extract fluid from the porous medium, and their positions are fixed and known.
The aim of this checkpoint is to determine the optimal location for a single injection well inside the domain Ω. The injection well introduces fluid back into the system at a unit rate, balancing the flow extracted by the production wells.
Finding the **optimal position of the injector well** (x_i, y_i) is crucial for ensuring that the flow rates at the production wells are balanced, thereby improving the efficiency and stability of the reservoir management.
Due to the high computational cost of simulating fluid flow with full-resolution models, the solution approach should leverage numerical upscaling techniques developed in the previous checkpoint. This will enable faster approximations of single-phase flow in porous media, reducing the computational burden while preserving accuracy.

3.1. Mathematical formulation of the problem

The problem is formulated as follows: find the pressure field p such that

$$\begin{cases} \nabla \cdot (-\boldsymbol{K}\nabla p) = f + b & \text{in } \Omega, \\ \nabla p \cdot \nu = 0 & \text{on } \partial\Omega, \\ p(x_e^i, y_e^i) = 0 & i = 1, 2, 3, 4, \end{cases} \tag{3}$$

where:

- (x_e^i, y_e^i) denote the positions of the i-th production well on the boundary,
- $\boldsymbol{K}$ is the permeability matrix of the porous medium,
- Ω is the domain of interest with boundary $\partial\Omega$,
- $f = 0$ is the source term,

[1]Executed on a laptop equipped with an Intel Core i5-1135G7 CPU (4 cores), 8 GB RAM, Windows 11 Pro 64-bit, and Python 3.11.11.

- b is an additional source term representing a unitary injection rate located at the unknown position (x_i, y_i) of the injection well.

The flow $\mathbf{q}$ at each well is then obtained from Darcy's law.
The optimization problem consists in finding the injector location $(x_i, y_i) \in \Omega$ that minimizes the imbalance of flow among the four production wells, formulated as

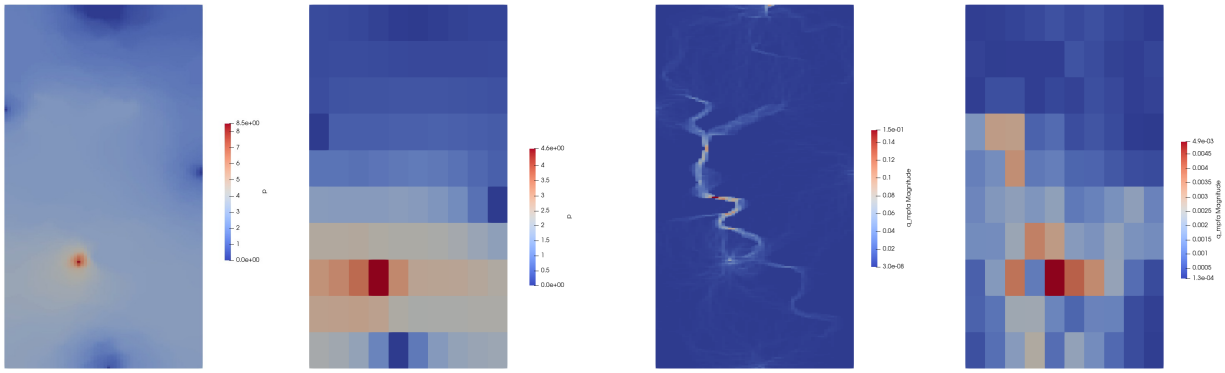
$$\arg \min_{(x_i,y_i)\in\Omega} \sum_{j=1}^{4} \left(\mathbf{q_j}(x_i, y_i) - \bar{\mathbf{q}}(x_i, y_i)\right)^2,$$

where $\mathbf{q_j}$ is the flow to the j-th production well and $\bar{\mathbf{q}}$ is the average production flow.

3.2. Methods

To address this problem, a coarse solver and an optimization routine need to be properly set up.

Coarse solver. We already have at our disposal a function, `solve_fine`, capable of computing the objective function in Section 3.1 given the selected layer and the positions of both production and injection wells. However, since `solve_fine` relies on a computationally expensive full-order model (FOM), using it repetitively during the optimization would be impractical.
To overcome this, we leverage the numerical upscaling technique developed in Checkpoint 1 to define a coarse solver, namely `solve_coarse`, that approximates the problem on coarser grids. This approach significantly reduces computational cost while maintaining sufficient accuracy to guide the optimization.



(a) Fine vs course p. (b) Fine vs course $\mathbf{q}$.

Figure 2: Comparison between fine-scale and coarse-scale solutions assuming the injection well is located at $(x_i, y_i) = (137,\ 197)$. Subfigure (a) shows the resulting pressure field, while subfigure (b) displays the corresponding Darcy velocity field.

Optimization. We use the function already implemented in Checkpoint 1 to upscale the layer's specific permeability tensor onto two coarse grids of resolutions 10×10 and 20×20, respectively. This step produces effective permeability fields at different scales, facilitating multi-resolution analysis.
Then, the optimization proceeds as follows:

Algorithm 1 Optimization Procedure for Injector Well Location

1st step:
  Evaluate `solve_coarse` at all cell centers of the 10×10 grid
  Identify the 20 lowest values of the cost functional
2nd step:
  For each of the 20 minima:
    Generate 15 sample points locally around the minimum
  Evaluate `solve_coarse` at these 300 points on the 20×20 grid
  Select the 20 points with lowest cost functional values
3rd step:
  Evaluate fine cost functional using `solve_fine` on the 20 selected points
  Choose the injector location corresponding to the minimum fine cost functional value
Return the optimal injector well location

During the execution of the above steps, the following additional assumptions were adopted to ensure robustness and consistency of the optimization process:

- **Step 1 – Exclusion of production well cells:** Cells containing a production well were excluded from the initial brute-force evaluation. This is motivated by the structure of the cost functional, which leads to an artificial global minimum when the injection well coincides with a production well. In such a configuration, all the injected flow is recovered by a single well, while the others register zero flow, resulting in a perfectly balanced—yet physically uninteresting—configuration.
- **Step 2 – Boundary filtering:** Among the 300 locally sampled points, those falling too close to the boundary of the computational domain were immediately discarded. This is based on the assumption that optimal locations for the injection well are unlikely to lie near the domain's edges. In particular, only points within the bounding box $(20, 345) \times (22, 648)$ were retained for further evaluation.
- **Step 3 – Sampling strategy:** Around each of the 20 selected coarse minima, 15 candidate points were sampled from a bivariate normal distribution. The mean of the distribution corresponds to the center of the coarse cell, and the covariance matrix was chosen as

$$\Sigma = \begin{pmatrix} 400 & 0 \\ 0 & 900 \end{pmatrix}.$$

  This choice reflects the rectangular shape of the subdomains, inducing an elliptical sampling region that accounts for anisotropy in the grid cell dimensions.

A sketch of the pipeline is summarized in Figure 3.

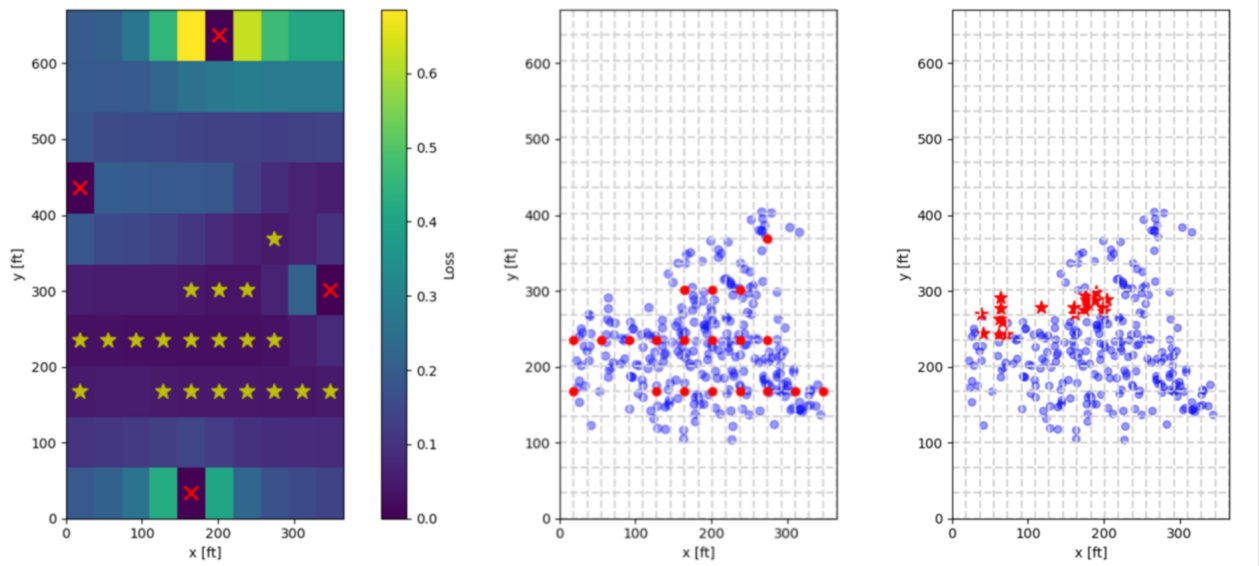


Figure 3: Layer 18, $\boldsymbol{\mu} = [0.545, 0.715, 0.603, 0.545]$. The figure illustrates the optimization pipeline from left to right: starting with the brute-force evaluation of the cost functional at the cell centers of a 10×10 coarse grid, followed by local sampling around the best candidates, and concluding with the selection of points for which the fine-scale solver `solve_fine` is applied.

This approach was preferred over standard local minimization techniques due to the highly non-convex nature of the objective function, which exhibits multiple local minima spread in the domain. Moreover, since it relies on independent evaluations, this technique can be easily parallelized, significantly reducing the overall execution time. This hierarchical strategy balances computational efficiency and solution accuracy by combining coarse-scale approximations with targeted fine-scale evaluations.

3.3. Numerical results

The adopted strategy proved to be highly effective: although it is unlikely to recover the exact coordinates of the optimal injection well location with sub-cell precision, it consistently identifies a sufficiently close candidate within just a few minutes of computation time. This represents a significant gain in efficiency when compared to traditional optimization algorithms such as Powell or Nelder–Mead, which are not only slower but also more susceptible to being trapped in one of the many local minima of the cost functional. As previously discussed, the non-convexity of the objective function makes global optimization particularly challenging, and in this context, our hierarchical and coarse-to-fine approach offers a more robust and computationally affordable alternative.

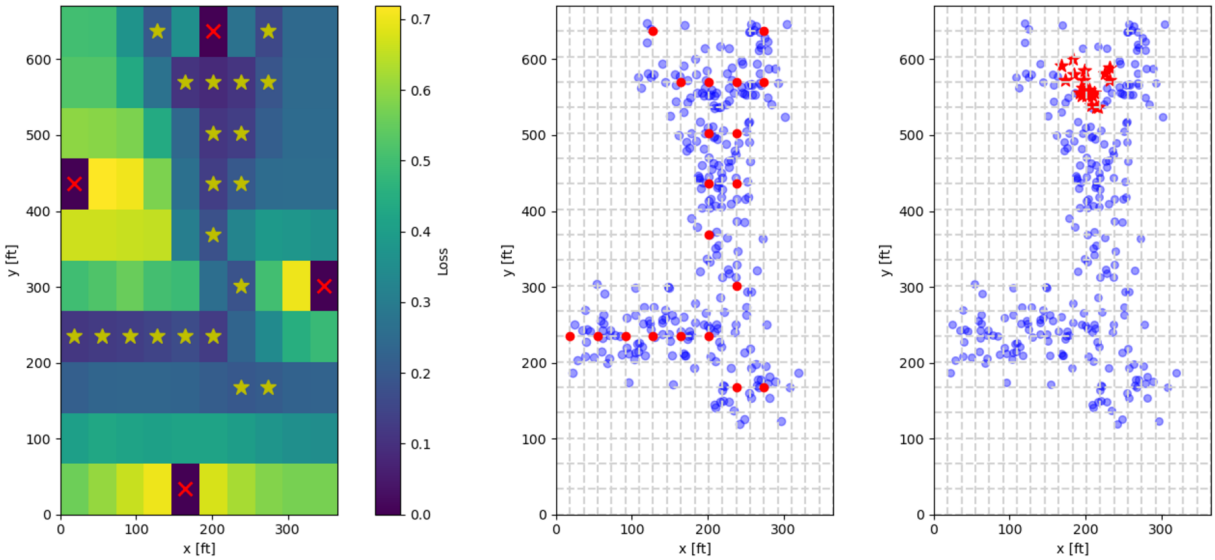


Figure 4: Layer 23, $\boldsymbol{\mu} = [0.545, 0.715, 0.603, 0.545]$.

Proceeding in this way, we are able to obtain a reasonably robust estimate of the optimal location, both when the final candidate points are clustered in a single region—as in Figure 4 (right)—and when they are spread across multiple distant regions of the grid—as shown in Figure 3 (right). In the latter case, classical minimization algorithms would have required an ad hoc strategy to detect and explore multiple distinct basins of attraction, significantly increasing the complexity of the optimization task.
Unfortunately, no ground truth solution or benchmark is available for direct comparison with the results obtained in this checkpoint. Nevertheless, for the test cases Layer 18 illustrated in Figure 3 and for Layer 23 in Figure 4, the outcomes of the implemented function are shown in Figure 5 and Figure 6, respectively.

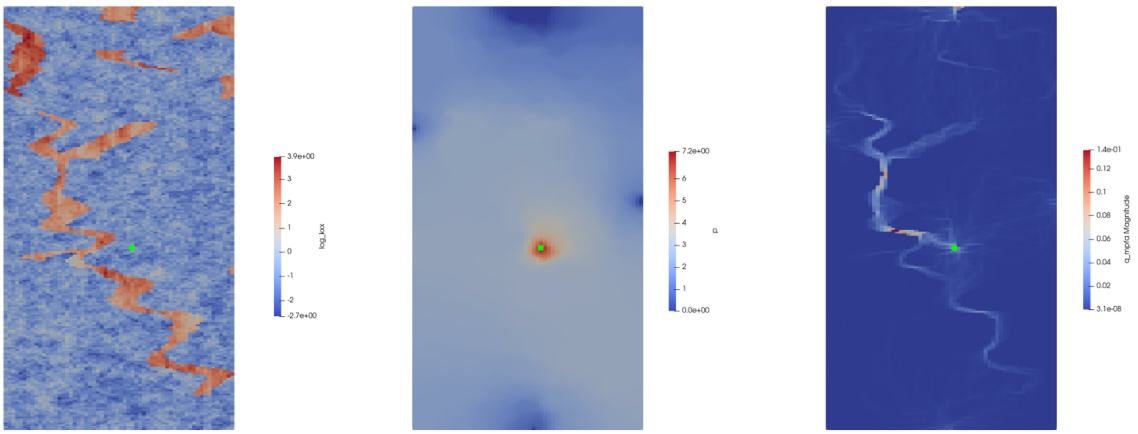


Figure 5: Layer 18, $\boldsymbol{\mu} = [0.545, 0.715, 0.603, 0.545]$. This Figure displays the Permeability of the layer (on the left), the pressure field (center) and the corresponding Darcy velocity field (right). The green point correspond to the optimal injection well position identified as $(x_i, y_i) = [203.78, 288.55]$.

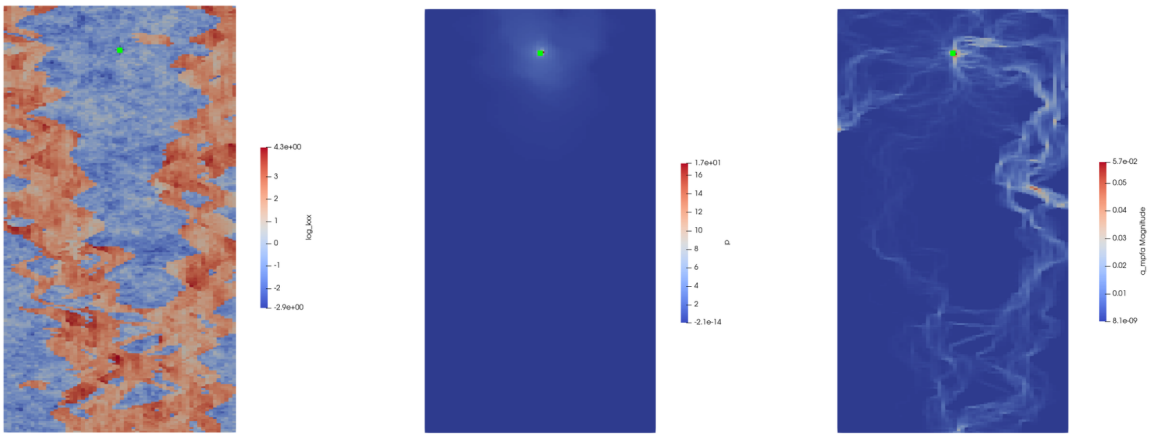


Figure 6: Layer 23, $\boldsymbol{\mu} = [0.545, 0.715, 0.603, 0.545]$. This Figure displays the Permeability of the layer (on the left), the pressure field (center) and the corresponding Darcy velocity field (right). The green point correspond to the optimal injection well position identified as $(x_i, y_i) = [183.22, 600.91]$.

4. Pollutant source identification

In this context, the extraction process at the production wells allow for the measurement of the concentration of a given pollutant over time. The objective is to infer **the initial location from which the pollutant** began its propagation.

4.1. Mathematical formulation of the problem

The diffusion of a pollutant concentration over time within a porous medium, such as a layer of the SPE10 dataset, can be described by coupling the transport equation driven by the advective flux with the steady-state Darcy's law for the velocity field:

$$\begin{cases} \dfrac{\partial c}{\partial t} + \nabla \cdot (\mathbf{q}\, c) = 0, \\ \mathbf{q} = -\mathbf{K}\nabla p, \end{cases} \tag{4}$$

where c denotes the concentration of the pollutant, $\mathbf{q}$ is the Darcy velocity, $\mathbf{K}$ is the permeability tensor, and p is the pressure field.
By solving this system of PDEs, it is possible to track the temporal evolution of the pollutant concentration, given its initial position in the domain and the underlying permeability distribution.
Based on the transport model described in (4), the following inverse problem is considered:

$$\arg \min_{(x_i, y_i) \in \Omega} J(x_i, y_i),$$

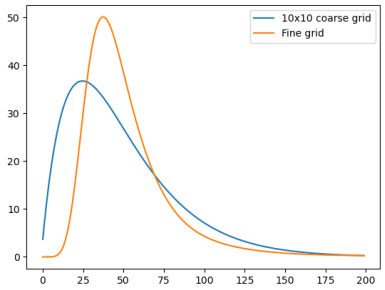
where the objective function $J(x_i, y_i)$ quantifies the discrepancy between the observed concentration profiles at the production wells and the model output simulated with an initial pollutant source located at (x_i, y_i):

$$J(x_i, y_i) = \frac{1}{T} \sum_{j=1}^{4} \sum_{t=1}^{T} \left(\text{outflow}_j(t) - \text{outflow}_{\text{model},j}(x_i, y_i, t) \right)^2. \tag{5}$$

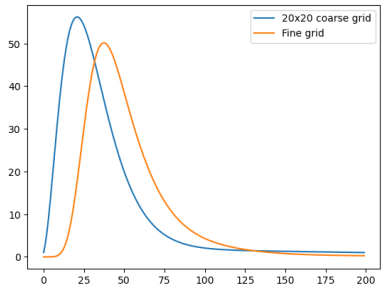
4.2. Methods

Similarly to the problem discussed in Section 3, addressing this task requires setting up a mathematical model for the transport of the pollutant, as described in equation (4), along with an appropriate optimization routine.

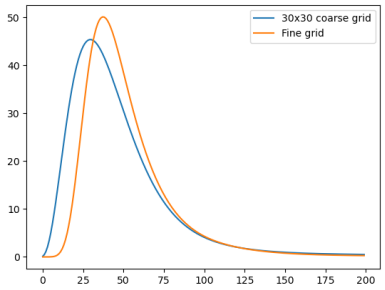
Surrogate Model. In this checkpoint, we were provided with a simulation code capable of modeling the transport of a pollutant within the domain, given the initial location of the concentration and all necessary features such as the positions of injection and production wells. The code outputs the pollutant breakthrough curves at the production wells over time, which are central to the analysis of the transport dynamics.
However, generating these curves through full-order simulations is computationally expensive. For this reason, we first aim to construct a surrogate model that approximates the pollutant transport on a coarse grid. While this model yields less accurate results, it enables significantly faster evaluations during the optimization phase. The implementation of this surrogate model did not pose major difficulties but required several modifications to the original codebase—most notably, in the way the initial mass is allocated. Depending on the pollutant's initial position relative to the coarse grid, discrepancies in mass allocation can range from 0% up to 3.8% in the worst cases.
The accuracy of the reconstructed concentration profiles at the production wells thus depends primarily on the resolution of the coarse grid: the finer the grid, the more accurate the approximation of the breakthrough curves. This trade-off between computational cost and accuracy is a central aspect of the surrogate modeling approach adopted in this checkpoint.



(a) Fine vs coarse 10×10 grid The surrogate model is 128 times faster.



(b) Fine vs coarse 20×20 grid The surrogate model is 64 times faster.



(c) Fine vs coarse 30×30 grid. The surrogate model is 43 times faster.

Figure 7: Layer 28, $\boldsymbol{\mu} = [0.601, 0.708, 0.021, 0.970]$, $(x_{init}, y_{init}) = (250, 100)$. Comparison of concentration breakthrough curves at the bottom production wells obtained using the full-order fine grid and surrogate models based on increasingly finer coarse grids (10×10, 20×20, and 30×30).

Optimization. In this section, we consider as loss function the one defined in Equation (5), introduced in Section 4.1. Building upon the performance of the optimization routine developed in Checkpoint 2, we opted for a similar strategy: instead of relying on sophisticated and computationally demanding minimization methods, we aim to solve the problem by exploiting different levels of grid resolution to balance accuracy and efficiency. The idea is to progressively restrict the search domain by combining coarse evaluations with selective refinement steps, allowing us to identify a near-optimal initial pollutant location in a fraction of the time required by conventional global optimization approaches. A schematic overview of the adopted algorithm is presented below:

Algorithm 2 Optimization Procedure for Initial Pollutant Concentration Location

Step 1:
  Retrieve the optimal injector and production well positions from Checkpoint 2
Step 2:
  Select 6 symbolic starting points on a 10×10 coarse grid
  Evaluate the surrogate transport model for each point
  Select the 2 points with lowest loss values to define a restricted search area
Step 3:
  Perform brute-force evaluation of the surrogate loss function on the selected subdomain (still on 10×10 grid resolution)
Step 4:
  Sample 15 candidate points around the best location in the subdomain
  Evaluate the loss on these points using a 30×30 coarse grid
  Choose the location minimizing the surrogate loss function
Return the estimated initial concentration location

As for Step 4, we adopted the same sampling strategy assumptions previously introduced in Section 3.2. A sketch of the pipeline is summarized in Figure 8.

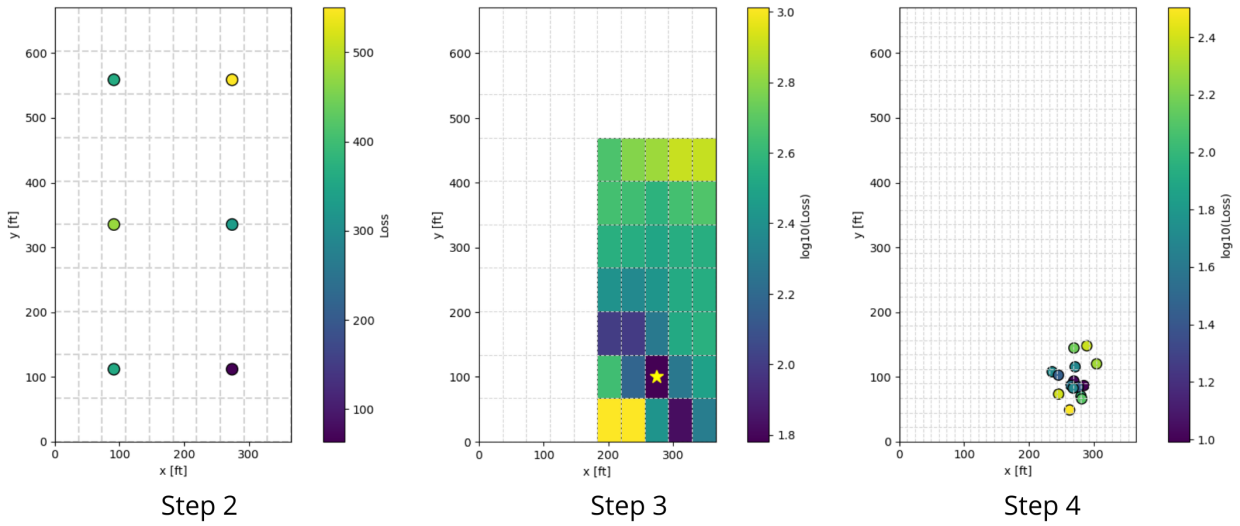


Figure 8: Layer 28, $\boldsymbol{\mu} = [0.601, 0.708, 0.021, 0.970]$. The figure illustrates the optimization pipeline from left to right: the process starts with the evaluation of six uniformly spaced points, which allows us to discard two-thirds of the domain. The second panel shows a brute-force search over the reduced region, leading to the identification of a promising location. Finally, 15 samples are drawn around this candidate to refine the estimation of the initial concentration position.

It is important to highlight that the entire optimization procedure relies exclusively on the surrogate transport model described above. This choice allows for significantly faster evaluations during the search phase, at the cost of a slight accuracy loss that is acceptable for locating promising candidate regions. Moreover, since the approach is based on independent evaluations of the loss functional at different spatial points, the entire pipeline is inherently parallelizable. This characteristic enables a substantial reduction in computational time when implemented on multi-core architectures.

4.3. Numerical results

An overview of the performance of this implementation is shown in Figure 9. Compared with the ground truth, the source of the pollutant is overall well identified, considering that there are two important sources of error: the location of the injection well and the discretization of the permeability field onto a coarse grid.

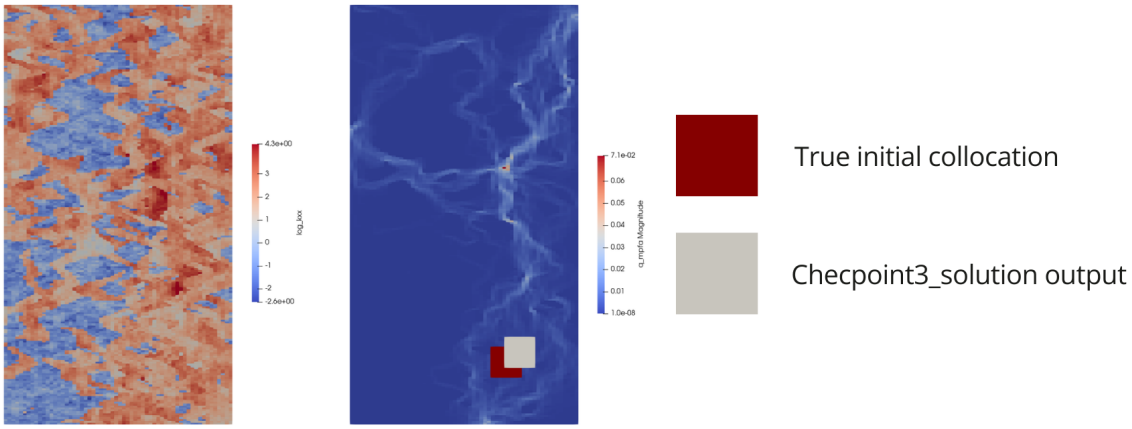


Figure 9: Layer 28, $\boldsymbol{\mu} = [0.601, 0.708, 0.021, 0.970]$. The figure illustrates the permeability $\boldsymbol{K_{xx}}$ of the layer (on the left) and and the corresponding Darcy velocity field (right). The two boxes identify the true inital concentration at $(x_{init}, y_{init}) = (250, 100)$ in red and the estimated one at $(x_{init}, y_{init}) = (271.55, 115.48)$ in gray.

Compared to the reference case shown in Figure 9, where the implemented function yields a localization error of approximately 26.5 ft, the results can be considered more than satisfactory, especially given that the entire optimization process relies exclusively on the surrogate version of the transport model. These outcomes provide a solid foundation for potential future improvements and refinements, discussed in Appendix A.

5. Conclusions

During this project, various mathematical tools have been implemented to address common problems related to the porous media setting.
First, the upscaling of the permeability field was addressed. Developing an upscaling procedure was essential to drastically reduce the number of degrees of freedom in the problem. While a fine grid provides accurate information, solving problems entirely on a fine grid is practically unfeasible, especially when the domain is large. Since the upscaling procedure serves as a baseline for all problems, its implementation was efficiently accelerated by parallelizing the most computationally expensive operation.
To locate the injection well, the non-convexity of the objective function makes global optimization particularly challenging. For this reason, instead of solving a complete minimization problem, a multifidelity approach was adopted. This method progressively refines the search in regions where the well is more likely to be located, proving to be an effective compromise between accuracy and efficiency.
Finally, the same idea was applied to identify the location where the pollutant begins its propagation. This implementation led to an accurate resolution of the problem while still leaving room for potential improvements, such as the use of an autoencoder.

References

[1] Anna Scotti and Alessio Fumagalli. Numerical methods for the geosciences, 2025. Lecture notes, Politecnico di Milano, A.Y. 2024–2025.

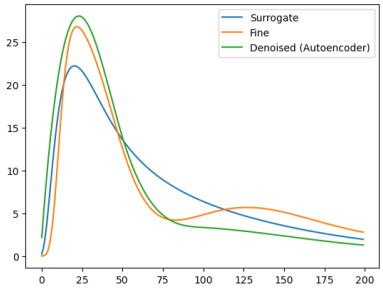
A. Appendix A

A possible improvement consists in exploring the identified region more effectively, by employing a suitable optimization routine within that subspace. However, due to the high computational cost of a fine-resolution solver for the transport problem, the idea is still to work with a coarse grid. To mitigate the errors introduced by this approach, a correction step must be performed at the end of the computation of the profile with the surrogate solver.
The idea is to adjust the concentration curves by defining a map $A : \mathbb{R}^n \to \mathbb{R}^n$ that performs this correction. In particular, this task can be tackled using a deep learning approach, such as an **autoencoder**. Given an informative training set, composed of pairs of concentration profiles obtained from the coarse and fine transport solvers respectively, the architecture presented in Table 2 is trained. Hyperparameter tuning is performed using the `optuna` library.

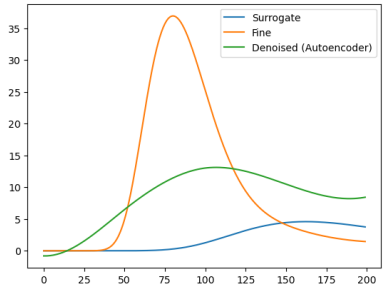
Component	Specification
Input dimension	200
Output dimension	200
Latent dimension	6
Hidden layers	1 layer: [57] neurons
Activation function (hidden layers)	`ReLU`
Activation function (output layer)	`Linear`
Dropout rate	0.0
Data type	`float64`
Learning rate	4.21×10^{-3}
Optimizer	Adam
Epochs	500
Loss function	MSE

Table 2: Summary of the autoencoder architecture

This approach proves to be promising; however, the performance of the network strongly depends on the quality and diversity of the data used for training. In fact, when using a relatively small dataset of only 144 curves, the model is exposed to a limited number of configurations, leading to both surprisingly good and notably poor results, as illustrated in Figure 10. It is likely that significantly increasing the dimensionality of the dataset (e.g., by including thousands of curves) would improve the generalization ability of the correction map, allowing for more accurate reconstruction.



(a) Autoencoder good reconstruction.



(b) Autoencoder bad reconstruction.

Figure 10: Comparison between the profile of the transport on a fine grid and that on a coarse grid, before and after the adjustment with the autoencoder.