Computer Methods in Engineering
Exercise on the conjugate gradient method

Problem 1
Consider the objective function given by the quadratic form

$$J(\vec{x}) = \vec{x}^\top \mathbf{A}\vec{x} = \vec{x}^\top \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \vec{x} \tag{1}$$

where $\vec{x} = [x_1, x_2]^\top$.

We now want to minimize $J(\vec{x})$ using the steepest descent method and the conjugate gradient method. For the steepest descent method, the gradient is given by

$$\nabla J(\vec{x}) = \nabla \left(\vec{x}^\top \mathbf{A}\vec{x} \right) = \frac{1}{2} \left(\mathbf{A} + \mathbf{A}^\top \right) \vec{x} = \mathbf{A}\vec{x} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \vec{x} \tag{2}$$

The steepest descent method is given by the iteration

$$\vec{x}_{k+1} = \vec{x}_k - \alpha_k \nabla J(\vec{x}_k) \tag{3}$$

where α_k is the step length at iteration k.

a) Write a Python code to minimize $J(\vec{x})$ using the steepest descent method. Start from the initial guess $\vec{x}_0 = [1, 1]^\top$. Use a fixed step length of $\alpha = 0.1$.

b) The Hessian is given by the matrix

$$\nabla^2 J(\vec{x}) = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2} & \frac{\partial^2 J}{\partial x_1 \partial x_2} \\ \frac{\partial^2 J}{\partial x_2 \partial x_1} & \frac{\partial^2 J}{\partial x_2^2} \end{bmatrix} \tag{4}$$

Show that the Hessian of a quadratic form $J(\vec{x}) = \vec{x}^\top \mathbf{A}\vec{x}$ is constant, and equal to $2\mathbf{A}$.

c) Use the Hessian to find the optimal step length at each iteration.

d) Write a Python code to minimize $J(\vec{x})$ using the steepest descent, when we use the step length $\alpha = \frac{\nabla J(\vec{x}_k)^\top \nabla J(\vec{x}_k)}{\nabla J(\vec{x}_k)^\top \mathbf{A} \nabla J(\vec{x}_k)}$ at each iteration k.

e) Write a Python code to minimize $J(\vec{x})$ using the conjugate gradient method. Start from the same initial guess $\vec{x}_0 = [1, 1]^\top$.

Problem 2
In this problem we will work on using the conjugate gradient method to solve an elliptic

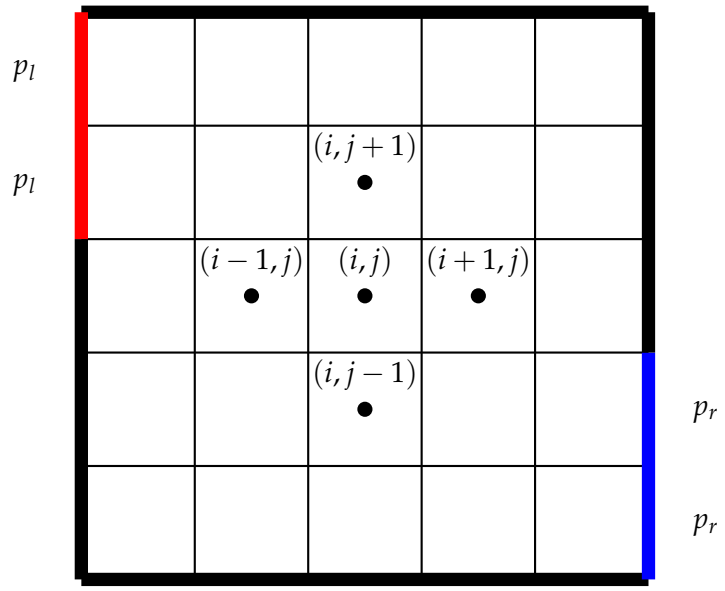

Figure 1: Grid representation of the sand-body connecting the two reservoirs. Thick black lines indicate no-flow boundaries, while red lines indicate boundaries connected to the left reservoir, and blue lines indicate boundaries connected to the right reservoir.

equation. More specifically, we will solve the Laplace equation we encountered in the exercise on elliptic equations. Recall that the exercise on elliptic equationswas considering flow in porous media as governed by the Darcy equation:

$$\vec{q} = -\frac{k}{\mu}\nabla p \quad ,$$

were q is the volumetric flow rate, k is the permeability (a measure for how well the porous medium allows for transport of fluids), μ is the viscosity of the fluid, and p is the fluid pressure. At steady state this gives the Laplace equation $\nabla^2 p = 0$. We considered a two-dimensional model, thus

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$$

Assume a sand-body connecting two fluid reservoirs at different pressure. The left reservoir has a pressure $p_l = 1 \times 10^5\,\mathrm{Pa}$, while the right reservoir has a pressure $p_r = 2 \times 10^5\,\mathrm{Pa}$. The sand-body has a shape between the two reservoirs as outlined in Fig. 1, where the grid cell size is $100\,\mathrm{m} \times 100\,\mathrm{m}$. Further, assume a viscosity of $1 \times 10^{-3}\,\mathrm{Pa\,s}$, a permeability of $1 \times 10^{-10}\,\mathrm{m}^2$, and assume a sand body thickness of $10\,\mathrm{m}$.

We saw in the exercise on elliptic equationsthat this gave the matrix representation

for the pressure field as $A\vec{P} = \vec{b}$, where the A matrix was given as

$$A = \begin{bmatrix}
-3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -3
\end{bmatrix}$$

and the vector $\vec{b}$ as

$$\vec{b} = \begin{bmatrix}
-100000 \\ 0 \\ 0 \\ 0 \\ 0 \\ -100000 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -200000 \\ 0 \\ 0 \\ 0 \\ 0 \\ -200000
\end{bmatrix}$$

a) Write a Python code to solve for the pressure $\vec{P}$ using the steepest decent method. If you want a residual smaller than 10^{-8}, how many iterations do you need.

b) Write a Python code to solve for the pressure $\vec{P}$ using the conjugate gradient method. How does your solution compare with the solution you obtained in the exercise on elliptic equations. Compare the time used by your own python code for the conjugate gradient method to the numpy library `linalg.inv`.