Computer Methods in Enigineering
Exercise on parabolic equations

In this exercise we will continue the work from the earlier exercise on solutions to Elliptic equations. Then we solved the Laplace equation $\nabla^2 f = 0$. Now, we will solve the diffusivity equation $\eta \nabla^2 p = \partial p / \partial t$.

The solution of the diffusivity equations can be obtained by different methods. In this exercise we will use the two methods we learned in the lecture, namely explicit and implicit solution. Both are based on the finite difference approximation. The centered difference for the second order derivative is given as

$$\frac{\partial^2 p}{\partial x^2} \simeq \frac{p(x+\Delta x, y, t) - 2p(x,y,t) + p(x-\Delta x, y, t)}{(\Delta x)^2}$$

Equivalently, in the y-direction, we have

$$\frac{\partial^2 p}{\partial y^2} \simeq \frac{p(x, y+\Delta y, t) - 2p(x,y,t) + p(x, y-\Delta y, t)}{(\Delta y)^2}$$

For the explicit solution, we use a forward difference for the time derivative

$$\frac{\partial p}{\partial t} \simeq \frac{p(x,y,t+\Delta t) - p(x,y,t)}{\Delta t}$$

For the two-dimensional system, our diffusivity equation $\eta \nabla^2 p = \partial p / \partial t$ is then given as

$$\eta \nabla^2 p = 0 \tag{1}$$

$$\eta \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) p = \frac{\partial p}{\partial t} \tag{2}$$

$$\eta \frac{\partial^2 p}{\partial x^2} + \eta \frac{\partial^2 p}{\partial y^2} = \frac{\partial p}{\partial t} \tag{3}$$

Assuming the same spatial step-length $\Delta x = \Delta y$, these second order partial derivatives can then be approximated using finite differences as

$$P^t_{i+1,j} - 2P^t_{i,j} + P^t_{i-1,j} + P^t_{i,j+1} - 2P^t_{i,j} + P^t_{i,j-1} = \frac{(\Delta x)^2}{\eta \Delta t} \left(P^{t+\Delta t}_{i,j} - P^t_{i,j} \right) \tag{4}$$

$$P^t_{i+1,j} + P^t_{i,j+1} - 4P^t_{i,j} + P^t_{i-1,j} + P^t_{i,j-1} = \frac{(\Delta x)^2}{\eta \Delta t} \left(P^{t+\Delta t}_{i,j} - P^t_{i,j} \right) \tag{5}$$

where we have used the index notation $P^t_{i,j} = P(i\Delta x, j\Delta y, t)$.

Just like in the previous exercise, assume a sand-body connecting two fluid reservoirs at different pressure. The left reservoir has a pressure $p_l = 1 \times 10^5$ Pa, while the

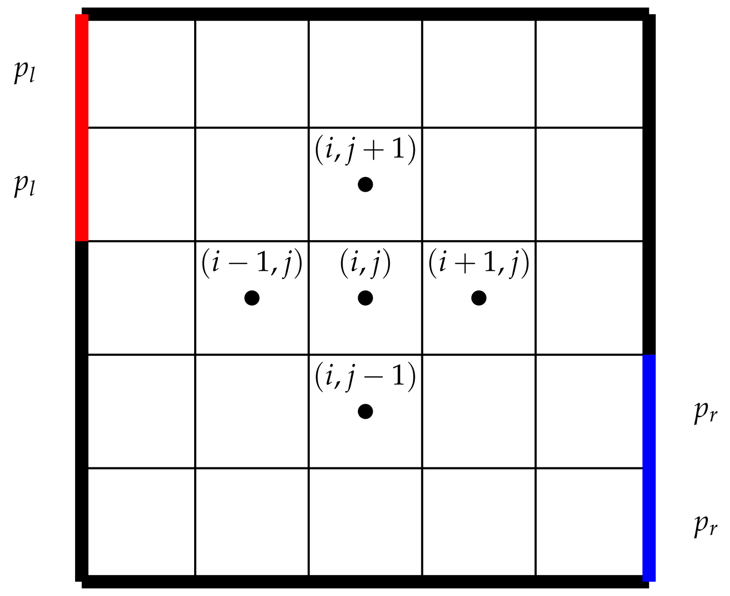


Figure 1: Grid representation of the sand-body connecting the two reservoirs. Thick black lines indicate no-flow boundaries, while red lines indicate boundaries connected to the left reservoir, and blue lines indicate boundaries connected to the right reservoir.

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

Problem 1

Assume the porous medium is filled with water at an initial pressure of $p = p_l = 1 \times 10^5$ Pa, and fully connected to the left and right boundary at the indicated boundaries. To simplify, we can assume that the porous medium stretches a half grid block into the two reservoirs where the connections are indicated.

a) Write out the explicit equation for this system i two dimensions.

Solution:

We first define

$$\alpha = \frac{\eta \Delta t}{(\Delta x)^2}$$

to ease our notation. Then the explicit equation for the next time-step is given by

$$P_{i,j}^{t+\Delta t} = \alpha P_{i+1,j}^{t} + \alpha P_{i,j+1}^{t} + (1-4\alpha) P_{i,j}^{t} + \alpha P_{i-1,j}^{t} + \alpha P_{i,j-1}^{t}$$

b) Create a code to solve the pressure field using an explicit scheme, when we define $\eta = 1, \Delta x = \Delta y = 1$.

Solution:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
#Grid size
innSide=5
#Properties
fEta=1.0
fDeltaX=1.0
fDeltat=0.01
# Define the alpha parameter
fAlpha=fEta*fDeltat/fDeltaX**2

#Boundary conditions
pl=1E5
pr=2E5

# Define function for explicit step
def solveExplicit(innSide,fAlpha,pl,pr,fTime):
  #Create an extended matrix, including boundaries
  E=np.zeros((innSide+2,innSide+2))
  E[:]=(pl+pr)/2

  #Add boundary conditions
  E[1,0]=pl
  E[2,0]=pl
  E[-2,-1]=pr
  E[-3,-1]=pr

  fCurTime=0.0
  while fCurTime < fTime:
    fCurTime += fDeltat
    #Update boundaries
    E[0,1:-1]=E[1,1:-1]
    E[-1,1:-1]=E[-2,1:-1]
    E[3:-1,0]=E[3:-1,1]
    E[1:-3,-1]=E[1:-3,-2]
    for jj in range(1,innSide+1):
      for ii in range(1,innSide+1):
        E[ii,jj]=(1.0-4*fAlpha)*E[ii,jj]+fAlpha*(E[ii-1,jj]+E[ii+1,jj]+E[ii,jj-1]+E[ii,jj+1])
  return E[1:-1,1:-1]
```

c) Solve the explicit scheme at time $t = 0.1, 1.0, 10.0$. How does the longer times compare with the elliptic solution in the previous exercise?

Solution:

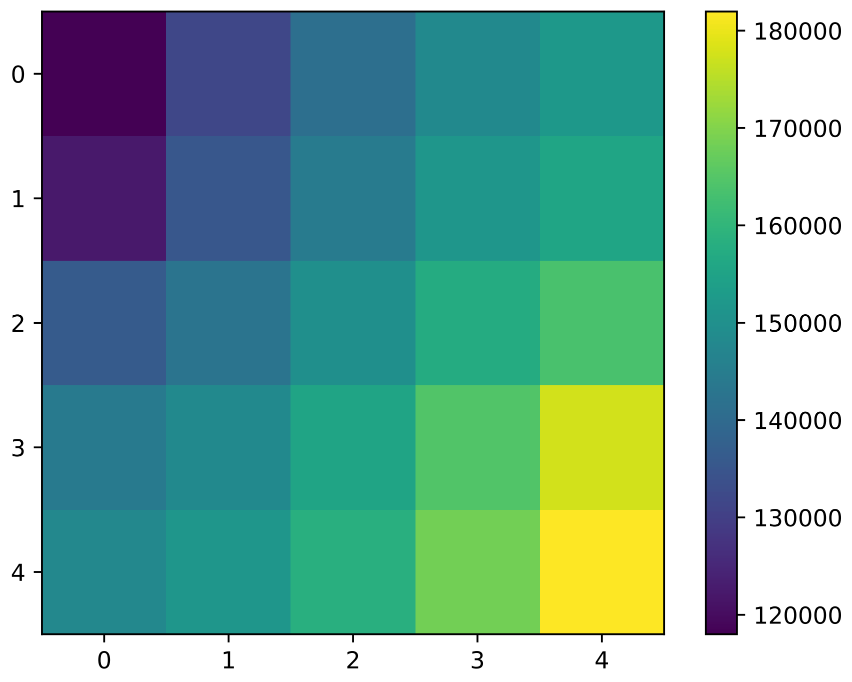
Running the code above, and plotting it, gives the output shown in the figure below for time $t = 1.0$. We see that already at $t = 1.0$, the solution is starting to be comparable to the steady-state solution given by the elliptic equation in the previous exercise.

Problem 2

We will now solve the same system using an implicit solution.

a) Create a code to solve the pressure field using an implicit scheme, using the same input as in the previous explicit scheme.

Solution:



```
#Implicit solution

# Define function for implicit step
def solveImplicit(innSide,fAlpha,pl,pr,fTime):
  A=np.zeros((innSide**2,innSide**2))
  b=np.zeros(innSide**2)
  p=np.zeros(innSide**2)
  p[:]=(pl+pr)/2

  #Set up matrix A for all internal cells
  for jj in range(0,innSide):
    for ii in range(0,innSide):
      cellNum=ii+innSide*jj
      A[cellNum,cellNum]+=1
      if ii>0:
        A[cellNum,cellNum]+=fAlpha
        A[cellNum,cellNum-1]-=fAlpha
      if ii<innSide-1:
        A[cellNum,cellNum]+=fAlpha
        A[cellNum,cellNum+1]-=fAlpha
      if jj>0:
        A[cellNum,cellNum]+=fAlpha
        A[cellNum,ii+innSide*(jj-1)]-=fAlpha
      if jj<innSide-1:
        A[cellNum,cellNum]+=fAlpha
        A[cellNum,ii+innSide*(jj+1)]-=fAlpha

  #Add boundaries to matrix A
  #and to vector b
  A[0,0]+=fAlpha
```

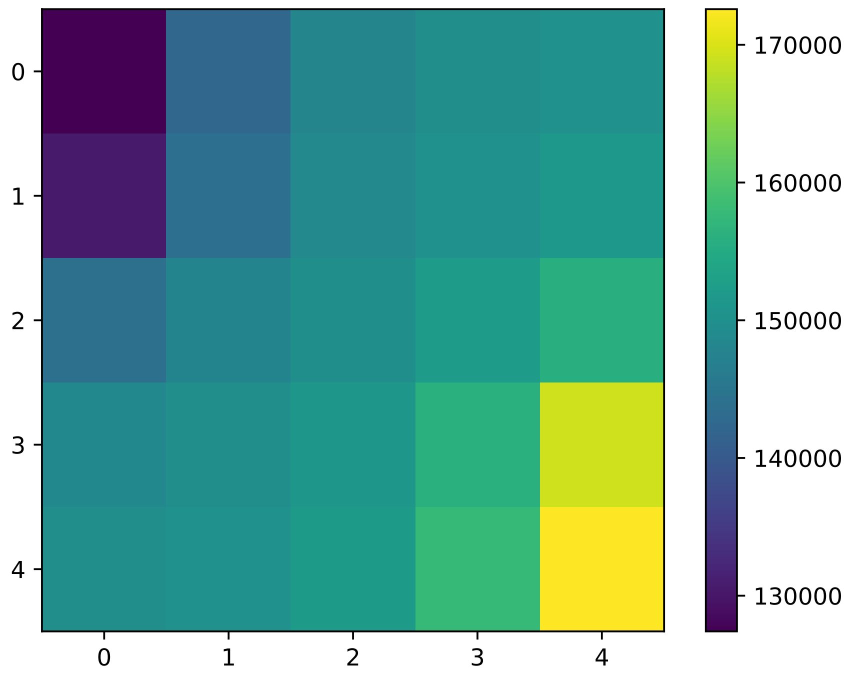
```
b[0]=pl*fAlpha
A[5,5]+=fAlpha
b[5]=pl*fAlpha
A[19,19]+=fAlpha
b[19]=pr*fAlpha
A[24,24]+=fAlpha
b[24]=pr*fAlpha

fCurTime=0.0
while fCurTime < fTime:
  fCurTime += fDeltat
  # Update pressure field
  p=np.dot(np.linalg.inv(A),p+b)
return p.reshape(innSide,innSide)
```

b) Solve the explicit scheme at time $t = 0.1, 1.0, 10.0$. How does this compare with the previous explicit solution, and with the elliptic solution in the previous exercise?

Solution:

Running the code above, and plotting it, gives the output shown in the figure below for time $t = 1.0$. This figure compares, but is slightly different, from the solution for the explicit scheme. The implicit scheme is unconditionally stable, and we can therefore use larger time-steps without the solution becoming unstable. However, the solution will still be more accurate for smaller time-steps.


0
1
2
3
4
170000
160000
150000
140000
130000