# MOD510: Mandatory project #2

**Deadline: Sun 6 October 2019**

Sep 25, 2019

**Abstract**

Understanding subsurface flow is challenging, but it also holds great potential value for society. For example, waterflooding is one of the most commonly applied secondary oil recovery techniques, and oil companies spend millions of dollars to develop simulators that can forecast where the injected water goes in the reservoir. This is valuable information both for the companies themselves, as well as for others that can utilize it, e.g., companies that generate electricity from geothermal energy.

Another concern in many areas of the world is the quality of drinking water. Models of subsurface flow are frequently used to predict how accidental spills will migrate and potentially impact the drinking water supply. While oil companies use seismic signals to learn about their reservoirs, this is expensive, and it is rarely done to monitor groundwater flow. Thus, in many cases there is limited knowledge about the subsurface, which goes a long way to justify the use of simpler models.

The goal of this project is to study fluid flow near injection / production wells. Among other things, you will learn how a 2D fluid flow problem can be split up into a series of 1D problems. Mathematically, injection and production wells behave very much like electrical charges, or gravitational fields. If production and injection rates are kept fixed, a constant potential field will be set up, and injected chemicals will follow specific paths, or streamlines, from injectors to producers. Under the simplest conditions, these streamlines will be curves that intersect the potential field at right angles. The technique of using streamlines to solve subsurface flow is powerful, and will in many cases give a good understanding of the flow.

**Notice.**

In this project we have chosen to present most of the theory before the exercises start. Therefore, there will be more information presented in the next section than what you need to solve the *first* exercise. In later exercises, more and more of the theory will drawn upon as it is needed.

**Learning Objectives.** By completing this project, you will:

- Implement your own numerical solver for the 1D radial Laplace equation.
- Trace out streamlines in two dimensions by performing numerical integration.
- Use streamlines to estimate the production profile of an inert tracer moving through an underground formation.
- Compare results from the streamline model with the output from a state-of-the-art finite difference simulator.

# Contents



# 1 Darcy's law for incompressible fluid flow

To make the later derivations as simple as possible, we assume that a) only a single fluid phase is flowing, b) that flow is incompressible, and c) that inertial forces are negligible, so that Darcy's law holds: Under the these assumptions, we may model the macroscopic fluid motion via the equations

$$\nabla \cdot \vec{u} = 0 \text{ (incompressible flow)} \tag{1}$$

$$\vec{u} = -\frac{k}{\mu} \cdot (\nabla P - \rho \vec{g}) \text{ (Darcy's law)}, \tag{2}$$

in which $\vec{u}[\mathrm{LT}^{-1}]$ represents the flux / discharge of fluid (volume per unit area per unit time), $k[\mathrm{L}^2]$ is permeability, $\mu[\mathrm{ML}^{-1}\mathrm{T}^{-1}]$ is viscosity, $P[\mathrm{ML}^{-1}\mathrm{T}^{-2}]$ is fluid pressure, $\rho[\mathrm{ML}^{-3}]$ is fluid density, and $\vec{g}[\mathrm{MT}^{-2}]$ is the vector of gravitational acceleration.

The porous medium is idealized so that we may take it as both homogeneous and isotropic, the effect of gravity is disregarded, and the fluid phase is assumed to have a constant viscosity. By combining (1) and (2) we then find

$$-\frac{k}{\mu}\nabla^2 P = 0\,, \tag{3}$$

or

$$\nabla^2 \phi = 0\,, \tag{4}$$

where we have defined a potential field

$$\phi \equiv -\frac{k}{\mu}\cdot P\,. \tag{5}$$

This is just the Laplace equation.

## 2 Steady-state radial flow from a well

The goal of the first part of the project will be to model the pressure distribution near a well for the following setup: an injection well is placed in the middle of a cylindrical porous medium, it is operated by pumping in fluid at a constant volumetric flow rate $Q$, while the pressure at the exterior boundary $r = r_e$ is taken to be a fixed value,

$$P\left(r = r_e\right) = P_e\,, \tag{6}$$

for some $P_e$. The point at which the fluid enters the formation is the well radius, $r = r_w$. We wish to be able to calculate the steady-state pressure distribution inside the formation, i.e., between $r_w$ and $r_e$.

In cylindrical coordinates, the Laplace equation becomes:

$$\nabla^2\phi = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\phi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2} + \frac{\partial^2\phi}{\partial z^2} = 0\,, \tag{7}$$

where $(r, \theta, z)$ are the coordinates. A considerable simplification occurs for the situation we wish to model, in which flow is assumed to be perfectly radial. Then, the scalar potential $\phi$ is a function of the $r$-coordinate only, i.e., the last two terms in the equation above vanish:

$$\nabla^2\phi = \frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}\phi}{\mathrm{d}r}\right) = 0\,. \tag{8}$$

We wish to calculate the steady-state pressure distribution near the well numerically with finite differences. That is, we have to solve

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}\phi}{\mathrm{d}r}\right) = 0\,, \tag{9}$$

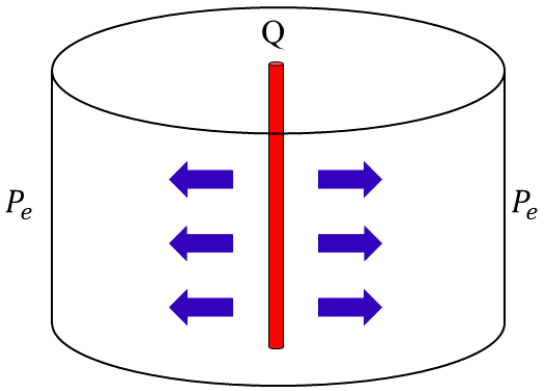


Figure 1: Radial flow from a well.

subject to boundary condition (6), as well as the fixed inflow rate condition at the well:

$$Q = 2\pi h r_w \cdot u(r_w) = 2\pi h r_w \cdot \left.\frac{\mathrm{d}\phi}{\mathrm{d}r}\right|_{r=r_w}, \tag{10}$$

where $u(r_w)$ is the radial flux at the location of the well (the well radius), and $h$ is the (constant) thickness of the porous medium.

## 2.1 Finite difference discretization

We divide the total domain into $N$ cylindrical shells. As illustrated in the figure below, we place our grid points in the *middle* of each cylinder.

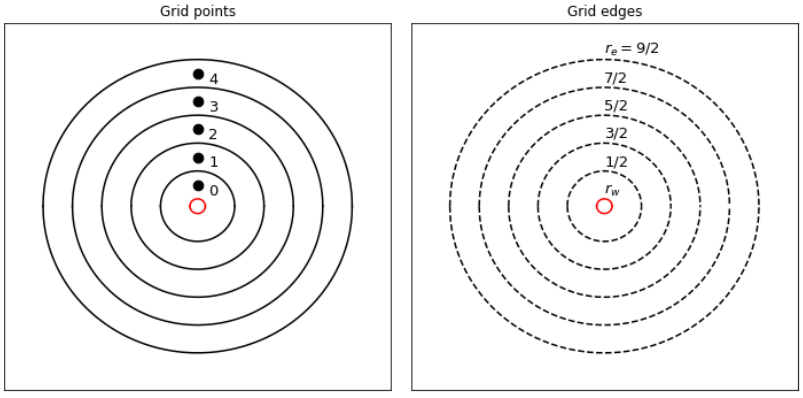


Figure 2: Top view of radial grid ($N = 5$).

The figure makes use of a fixed grid spacing, but in practice one would often choose $\Delta r$ so that each block has the same volume. Hence, for generality we will allow the grid spacing to vary in our mathematical derivations.

The outer radius of grid block $i$ is:

$$r_{out}^{i} \equiv r_{i+1/2} = r_i + \frac{\Delta r_i}{2} = r_w + \sum_{k=0}^{i} \Delta r_k \,. \tag{11}$$

Similarly, the inner radius is

$$r_{in}^{i} \equiv r_{i-1/2} = r_i - \frac{\Delta r_i}{2} = r_w + \sum_{k=0}^{i-1} \Delta r_k = r_{i+1/2} - \Delta r_i \,. \tag{12}$$

We index our grid blocks starting from zero, $i = 0, 1, \ldots, N-1$. Letting $f(r) = r\phi'(r)$, equation (9) for grid point $i$ can be converted to discrete form as follows:

$$0 = \left.\frac{\mathrm{d}f(r)}{\mathrm{d}r}\right|_i \approx \frac{f(r_{i+1/2}) - f(r_{i-1/2})}{\Delta r} \,. \tag{13}$$

We also need to find a way to evaluate the function $f$ at the edges of the grid blocks. For the *inner* grid points we can use centered finite differences once more, i.e.,

$$f(r_{i+1/2}) = r_{i+1/2} \cdot \left.\frac{\mathrm{d}\phi}{\mathrm{d}r}\right|_{r=r_i+1/2} \tag{14}$$

$$\approx r_{i+1/2} \cdot \frac{\phi_{i+1} - \phi_i}{\frac{1}{2}(\Delta r_i + \Delta r_{i+1})} \,, \tag{15}$$

and

$$f(r_{i-1/2}) = r_{i-1/2} \cdot \left.\frac{\mathrm{d}\phi}{\mathrm{d}r}\right|_{r=r_i-1/2} \tag{16}$$

$$\approx r_{i-1/2} \cdot \frac{\phi_i - \phi_{i-1}}{\frac{1}{2}(\Delta r_{i-1} + \Delta r_i)} \,. \tag{17}$$

With the above approximations, we get the following numerical scheme for the inner grid points $i = 1, \ldots, N-2$:

$$T_{i+1/2}\,(\phi_{i+1} - \phi_i) - T_{i-1/2}\,(\phi_i - \phi_{i-1}) = 0 \,, \tag{18}$$

where we have defined a set of constant *transmissibility factors*:

$$T_{i+1/2} = \frac{2r_{i+1/2}}{\Delta r_i(\Delta r_i + \Delta r_{i+1})} \,. \tag{19}$$

For the first grid point $i = 0$ we see that we have a problem, because formula (18) requires us to compute quantities outside of the defined grid (i.e., *inside* the well):

$$T_{1/2}\,(\phi_1 - \phi_0) - T_{-1/2}\,(\phi_0 - \phi_{-1}) = 0 \,, \tag{20}$$

However, the second term in the above equation is really a finite difference approximation to the flux out of the well:

$$T_{-1/2}\,(\phi_0 - \phi_{-1}) \approx \frac{r\frac{\mathrm{d}\phi}{\mathrm{d}r}\Big|_{r=r_w}}{\Delta r_0} = \frac{Q}{2\pi h \Delta r_0}\,. \tag{21}$$

Thus, we simply replace this expression by a constant term that goes on the right-hand side of the equation.

Similarly, for the last grid point $i = N-1$ formula (18) becomes:

$$T_{(N-1)+1/2}\,(\phi_N - \phi_{N-1}) - T_{(N-1)-1/2}\,(\phi_{N-1} - \phi_{N-2}) = 0\,, \tag{22}$$

and this requires us to make evaluations outside of the defined external grid boundary. However, since we know the pressure at $r = r_e$, we also know the value $\phi_e = \phi(r = r_e)$ there. Thus, in the equation above we replace $\phi_N$ by $\phi_e$, and we use a slightly different approximation than before to compute the transmissibility factor,

$$f(r_{i+1/2}) = f(r_{(N-1)+1/2}) = r_e \cdot \frac{\mathrm{d}\phi}{\mathrm{d}r}\Big|_{r=r_e} \tag{23}$$

$$\approx r_e \cdot \frac{\phi_e - \phi_{N-1}}{\frac{\Delta R_{N-1}}{2}}\,, \tag{24}$$

so that we may take

$$T_{(N-1)+1/2} = \frac{2r_e}{\Delta r_{N-1}^2}\,. \tag{25}$$

# 3 Streamlines

Given a (velocity) vector field $\vec{v}$, a streamline is a curve everywhere tangential to the direction of $\vec{v}$. If $\vec{dr} = (dx, dy)$ represents an infinitesimal distance along a streamline, by definition $\vec{dr}$ must be parallel to the velocity field. In two dimensions, this means that:

$$\frac{dx}{v_x} = \frac{dy}{v_y}\,, \tag{26}$$

where $\vec{v} = (v_v, v_y)$ is the velocity field.

We remark that for non-steady flow, streamlines are different from *pathlines*. While the latter are the actual physical trajectories followed by particles in the flow field, streamlines provide a 'snapshot' of the field at any given moment in time. However, in this project we shall assume the velocity field to be frozen in time, so that pathlines and streamlines are the same. If a non-reacting fluid particle is placed at a point $(x(0), y(0))$ at time $t = 0$, we then have

$$\frac{dx(t)}{dt} = v_x(x(t), y(t)) \tag{27}$$

$$\frac{dy(t)}{dt} = v_y(x(t), y(t))\,, \tag{28}$$

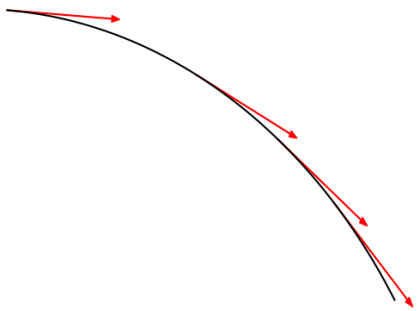

Figure 3: A streamline (solid curve) is everywhere tangent to the direction of the vector (e.g, velocity) field (arrows).

where $(x(t), y(t))$ is the position of the particle at time $t$. At time $t > 0$, the position of the particle is therefore

$$x(t) = x(0) + \int_0^t v_x(x(t'), y(t'))\, dt' \tag{29}$$

$$y(t) = y(0) + \int_0^t v_y(x(t'), y(t'))\, dt' \,. \tag{30}$$

# 4 Exercise 1: Radial flow from a single well

**Part 1.** By combining equations (18), (19), (21), and (25), we obtain a matrix equation $A\vec{\phi} = \vec{b}$, where $\vec{\phi}$ holds the values of the potential function at each of the $N$ grid points.

For the case $N = 5$, show that the coefficient matrix $A$ becomes

$$A = \begin{bmatrix} -T_{1/2} & T_{1/2} & 0 & 0 & 0 \\ T_{1/2} & -(T_{1/2} + T_{3/2}) & T_{3/2} & 0 & 0 \\ 0 & T_{3/2} & -(T_{3/2} + T_{5/2}) & T_{5/2} & 0 \\ 0 & 0 & T_{5/2} & -(T_{5/2} + T_{7/2}) & T_{7/2} \\ 0 & 0 & 0 & T_{7/2} & -(T_{7/2} + T_{9/2}) \end{bmatrix},$$

What is the right-hand side vector?

**Part 2.** The goal is to write a Python program that calculates the estimated pressure distribution at the midpoint of each radial grid block.

First, we need to set up the grid. Finish writing the following Python code:

```
n_blocks = 5  # number of grid blocks / points
r_well = 0.1  # well radius (m)
r_exterior = 20.0  # outer boundary (m)

# Use a constant grid spacing (at least for now):
radial_grid_edges = np.linspace(r_well, r_exterior, n_blocks+1)
dr_vec = radial_grid_edges[1:]-radial_grid_edges[0:-1]

# To do: Calculate grid points r_i, and store them in an array:
radial_grid_midpoints = ...
```

**Part 3.** Next, we need to select values for the other input parameters in our model, e.g.:

```
    # Constant formation and fluid properties:
    viscosity = 1.0e-3  # 1 cP = 0.001 Pa*s
    permeability = 1.0e-12  # m2
    h = 2.0  # formation height (m)

    # Boundary conditions:
    Q = 21.0/3600.0  # 21 m3/hr --> m3/sec
    p_exterior = 100.0e5  # 100 bar ---> Pa
```

Finally, we must set up and solve the matrix equation. During the implementing stage, it is recommended to use numpy.linalg.solve for the solution part, e.g.:

```
# Grid transmissibility factors for grid edges:
T = np.zeros(n_blocks+1)  # T_-0.5, T_0.5, T_1.5, ...,

# Compute T_(i+1/2) for internal grid points:
for i in range(n_blocks-1):
    ... # implement here
    T[i+1] = ...

# Compute transmissibilities at grid boundaries:
T[0] = ...
T[-1] = ...

# Setup and solve matrix equation:
A = np.zeros(shape=(n_blocks, n_blocks))
b = np.zeros(n_blocks)

...  # include code here

phi_sol = np.linalg.solve(A, b)
pressure_sol = ... # convert to pressure
```

Note: It can help to wrap your solution algorithm inside a function, so that you can easily test it later for different values of $N$.

**Part 4.** Going back to equation (9), show that the analytical solution to the steady-state pressure at radial distance $r$ from the well is:

$$P(r) = P_e - \frac{Q\mu}{2\pi kh} \cdot \ln\left(\frac{r}{r_e}\right) . \tag{31}$$

Verify that your numerical solution seems to converge to the exact one as $\Delta r \to 0$.

**Part 5.** Once everything seems to be working, investigate how much you are able to speed up your program if you use a sparse matrix solver, compared to if you use the standard dense matrix solver.

**(Optional) Part 6.** Adapt your code so that it also handles the case with a constant pressure at the well. For this case the analytical solution is most conveniently displayed as:

$$P(r) = P_w + \frac{P_e - P_w}{\ln\left(\frac{r_e}{r_w}\right)} \cdot \ln\left(\frac{r}{r_w}\right) . \tag{32}$$

# 5 Exercise 2: Velocity field for a system of wells

If the surrounding medium is very large, we may neglect the finite radius of the well, and model it as a point source/sink term in the governing fluid flow equations. Assuming flow to/from a well to be perfectly radial in nature and governed by Darcy's law, we have for the case of a single well that

$$u(r) = \frac{\mathrm{d}\phi}{\mathrm{d}r} = \frac{Q}{2\pi h r} , \tag{33}$$

in radial coordinates, where $u = u(r)$ is the radial Darcy flux at position $r$, and $\phi = \phi(r) = -(k/\mu)p(r)$ is the potential scalar field there. Hence, it follows from a simple integration that

$$\phi = \frac{Q}{2\pi h} \cdot \ln(r) + C = \frac{Q}{4\pi h} \cdot \ln\left\{(x - x_w)^2 + (y - y_w)^2\right\} + C \tag{34}$$

in Cartesian coordinates, where $r = \sqrt{(x - x_w)^2 + (y - y_w)^2}$ is the distance to a given point from the location of the well, $(x_w, y_w)$. We do not really care about the numerical value of the constant term $C$; when we take the derivative of $\phi$ to obtain velocity, it disappears.

Suppose instead that we have $N_w$ wells. Let $(x_i, y_i)$ denote the location of well $i$, and let $Q_i$ be the volumetric flow rate of well $i$, with $Q_i > 0$ for injectors and $Q_i < 0$ for producers. By the superposition principle, it can be shown that the total fluid potential is simply the sum of the potentials for each well.

**Part 1.** The Darcy flux $\vec{u}$ measures the rate of fluid flow through cross-sections of the porous medium. However, as the injected fluid has to move around the solid particles (grains), the average macroscopic velocity of fluid particles traveling through the medium has to account for porosity, $n$:

$$\vec{v} = \frac{\vec{u}}{n} . \tag{35}$$

Show that for the general case of $N_w$ wells with arbitrary placements, the resulting velocity field $\vec{v} = (v_x, v_y)$ is given by

$$v_x = \frac{1}{n}\frac{\mathrm{d}\phi}{\mathrm{d}x} = \frac{1}{2\pi hn} \cdot \sum_{i=0}^{N_w-1} \frac{Q_i(x - x_i)}{(x - x_i)^2 + (y - y_i)^2}\,, \tag{36}$$

and

$$v_y = \frac{1}{n}\frac{\mathrm{d}\phi}{\mathrm{d}y} = \frac{1}{2\pi hn} \cdot \sum_{i=0}^{N_w-1} \frac{Q_i(y - y_i)}{(x - x_i)^2 + (y - y_i)^2}\,. \tag{37}$$

**Part 2.** Consider the special case of a *well doublet*: we have two wells, an injector located at $(d, 0)$, and a producer at $(-d, 0)$. Let the constant injection rate be $Q > 0$. Show that in this situation, the components of the velocity vector field are:

$$v_x = \frac{Q}{2\pi hn} \cdot \left\{ \frac{x - d}{(x - d)^2 + y^2} - \frac{x + d}{(x + d)^2 + y^2} \right\}, \tag{38}$$

and

$$v_y = \frac{Q}{2\pi hn} \cdot \left\{ \frac{y}{(x - d)^2 + y^2} - \frac{y}{(x + d)^2 + y^2} \right\}. \tag{39}$$

**Part 3.** Later we are going to use numerical calculations to estimate the total travel time, or time-of-flight, for a non-reacting fluid particle to traverse different streamlines. It is always smart to compare our numerical results with known analytical solutions, should they be available. For our scenario, the time it takes to travel along the shortest, straight-line path from injector to producer is:

$$\tau_0 = \int_d^{-d} \frac{dx}{v_x}\,. \tag{40}$$

Show by exact integration that:

$$\tau_0 = \frac{4\pi hnd^2}{3Q}\,. \tag{41}$$

For a given formation, how will the time until water breakthrough at the producer be affected if we double the distance between the wells, but keep the flow rate the same?

**Part 4.** Finish implementing the following Python routine:

```
def calc_velocity_well_doublet(x, y, d, Q, h, poro):
    """
    Calculate velocity field for a well doublet: an injector
    located at (d,0), and a producer at (-d, 0).

    :param x: x value(s) at which to compute the velocity (m).
    :param y: y value(s) at which to compute the velocity (m).
    :param d: Half-distance between injector and producer (m).
    :param Q: Volumetric injection rate in m3/s.
    :param h: Formation height (m).
    :param poro: Constant formation porosity.
    :return: Components of velocity vector field: (vx, vy)
    """
```

The function should work equally well whether a single pair of numbers is given as input, in which case a single pair of numbers is also returned, or if $x$ and $y$ are numpy arrays.

**Part 5.** Suppose that $d = 200$ m, $h = 110$ m, $n = 0.3$, and $Q = 130$ m$^3$/hr. These values are representative of water injection at the Ekofisk field.

According to the streamline model, how many days does it take before the first injected fluid reaches the producer?

# 6 Exercise 3: Tracing streamlines for a well pair

To trace out a streamline curve, we first 'place' an imaginary fluid particle at a point very close to the injector. Using equations (29) and (30), we then integrate in a step-by-step manner, taking successive time steps until we reach the production well. Alternatively, we can start at the producer, and trace backwards until we reach an injector. In that case, we need to change the sign of the velocity field when doing the integration.

**Part 1.** Unless otherwise is specified, we shall in this exercise assume that $d = 8$ m, $h = 2$ m, $n = 0.25$, and $Q = 21$ m$^3$/hr.

How many hours does it take before the first injected fluid reaches the producer?

**Part 2.** The simplest integration method, though rarely used in practice, is the (explicit) Forward Euler method:

$$x_{n+1} = x_n + v_x(x_n, y_n) \cdot \Delta t \tag{42}$$

$$y_{n+1} = y_n + v_y(x_n, y_n) \cdot \Delta t\,. \tag{43}$$

Here, $x_n$ is the approximate value for the true $x$-coordinate $x(t_n)$ at the $n$-th time step (and similarly for $y$).

Consider the following Python code:

```
# Define model parameters:
Q = 21.0/3600.0  # 21 m3/hr
d = 8.0  # m
h = 2.0  # m
poro = 0.25
```

```
# Trace out a single streamline:
dr = 0.1
dr2 = dr*dr
dt = 0.01*3600.  # constant time step

stl_x = []
stl_y = []
stl_dt = []

x_n = d-dr
y_n = 0.0

while True:
    v_x, v_y = calc_velocity_well_doublet(x_n, y_n, d, Q, h, poro)
    x_n += v_x*dt
    y_n += v_y*dt
    dist2 = (x_n+d)**2+y_n**2
    if dist2 > dr2:
        stl_x.append(x_n)
        stl_y.append(y_n)
        stl_dt.append(dt)
    else:
        break
print('Number of steps taken: {:d}.'.format(len(stl_x)))
print('Last point: ({}, {}).'.format(stl_x[-1], stl_y[-1]))
print('Total time: {} hours.'.format(np.cumsum(stl_dt)[-1]/3600.))

fig_streamline = plt.figure()
plt.plot(stl_x, stl_y)
```

- What does the algorithm do? Compare the output with what you calculated in Part 1.
- What happens as we lower the value of $\Delta r$? Why?
- What happens if we set $x_n = x_n + \Delta r$ initially (i.e., plus instead of minus)?

**Part 3.** Next, let $\vec{x}_0 = (x(0), y(0))$ denote a general point lying a distance $\Delta r$ away from the injection well:

$$x(0) = d + \Delta r \cos \beta_o \tag{44}$$

$$y(0) = \Delta r \sin \beta_o \,. \tag{45}$$

Make a plot showing the points we get as we vary $\beta_o$ from

$$\beta_o = \beta_o(i) = (i+1) \cdot \frac{\pi}{N} \,, \tag{46}$$

for $i = 0, 1, \ldots, N-1$? What is the physical meaning of the parameter $\beta_o$?

**Part 4.** We now want to launch $N = 5$ different streamlines from the injector:

```
# Trace N streamlines from injector to producer:
Q = 21.0/3600.0  # 21 m3/hr
d = 8.0  # m
h = 2.0  # m
poro = 0.25

dr = 0.1  # m
dt = 0.01*3600.0  # 0.01 hrs in seconds
N = 5  # number of streamlines

for i in range(N):
    # To do: trace out the streamline:
    angle = ...
    x0 = ...
    y0 = ...
```

Finish implementing the above algorithm by tracing out each streamline until it hits (close to) the production well. Moreover, for each of the streamlines estimate the time (time-of-flight) $\tau$ it takes for a non-reacting fluid particle to travel along the path from injector to producer.

Next, visualize the streamlines. For each streamline, include in the figure the computed $\tau$-value as a text label above that streamline. Mark out the locations of the two wells in the plot; use a different colour for each well.

**Part 5.** It is not only for the straight-line path from injector to producer that we can compute $\tau$ analytically. Compare your numerical results from the previous exercise with exact values recorded in the following table (obtained with a very complicated formula from [1]):

| Streamline index | Angle, $\beta_o$ | Time-of-flight, $\tau$ (hours) |
|---|---|---|
| 0 | $\pi/5$ | 251.44 |
| 1 | $2\pi/5$ | 34.57 |
| 2 | $3\pi/5$ | 12.61 |
| 3 | $4\pi/5$ | 7.51 |
| 4 | $\pi$ | 6.38 |

Is the agreement between numerical and analytical time-of-flight good for all computed streamlines? If not, what steps could you take to remedy this?

**Part 6.** Extend the code from Part 4 to handle the general case of $N$ streamlines launched from the injector. However, do not trace out all of the streamlines fully, only those that reach the producer within a given period of time.

It is recommended to encapsulate your algorithm within a Python function and/or class, for later reusability.

# 7 Exercise 4: Predict tracer production profiles

Suppose a non-reacting chemical (i.e., a tracer) is injected into the formation. We can use our streamline model to predict the chemical concentration at the

production well as a function of time:

$$C_e(t) \approx \sum_{i=0}^{N-1} C_{i,e}(t)\Delta f\,, \tag{47}$$

where $C_{i,e}$ denotes the concentration of the tracer at the final point of streamline $i$, and $\Delta f = \Delta Q_i/Q$ is the fraction of the total production rate associated with that streamline. Since we launch streamlines at equally spaced angles, they are all assumed to carry the same amount of fluid, which means that $\Delta f = \frac{1}{N}$. Finally, if $\tau_i$ is the total time it takes to traverse streamline $i$, $C_{i,e}(t)$ must equal the initial chemical concentration if $t < \tau_i$, and the injected concentration otherwise.

**Part 1.** Suppose we start injecting water into the formation at time $t = 0$. Use the input parameters from Exercise 3 to plot the percentage of injected water that has reached the production well at times $t > 0$.

To see the effect of streamline resolution, include in the plot the production curve for different values of $N$. The two largest values should give more or less identical results.

**Part 2.** Roughly the same scenario has been simulated with a finite difference simulator. Results from that program can be found in the text file `FD_water_btc.dat`, and are plotted below.

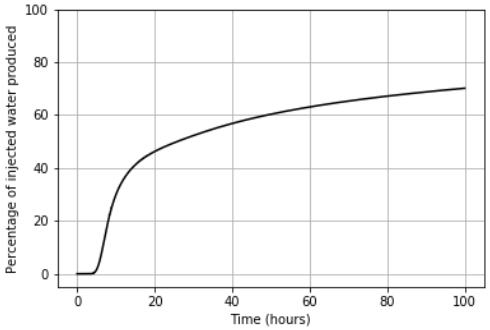


Figure 4: Results from a finite difference simulator.

Make your plot from the previous exercise dimensionless by dividing the time axis by the shortest time-of-flight, $\tau_0$. Use a logarithmic spacing on the time axis, and also include output from the finite difference simulator in the figure.

How long does it take for 50 % of the injected water to reach the producer? Provide your answer both in dimensional and dimensionless units.

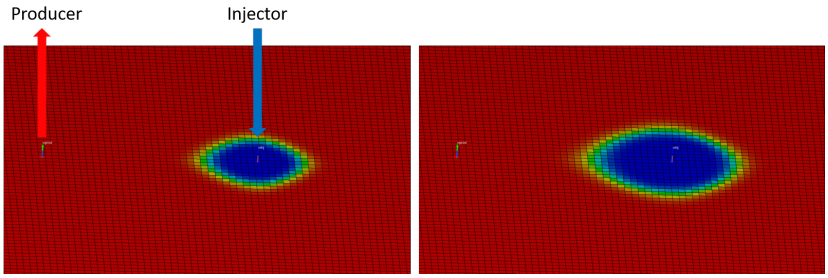


Figure 5: Distribution of injected water at two different times (finite difference simulation).

**Note.**

You will not be able to perfectly match the two simulations, due to different assumptions made regarding boundary conditions in the two cases, and because of inherent numerical errors associated with the finite difference implementation.

**(Optional) Part 3.** Make a Python program capable of tracing streamlines for *arbitrary* well placements. You may assume a single production well, but any number of injectors. (hint: here it is smart to start the streamline integration at the producer, and go backwards)

Apply your program by plotting streamlines for a five-spot well pattern.

# 8 Guidelines for project submission

The assignment is provided both as a PDF, and as a Jupyter notebook. However, the work done to answer the exercises only has to be handed in as a notebook, though you can submit an additional PDF if you want. You should bear the following points in mind when working on the project:

- Start your notebook by providing a short introduction in which you outline the nature of the problem(s) to be investigated.
- End your notebook with a brief summary of what you feel you learnt from the project (if anything). Also, if you have any general comments or suggestions for what could be improved in future assignments, this is the place to do it.
- All code that you make use of should be present in the notebook, and it should ideally execute without any errors (especially run-time errors). If you are not able to fix everything before the deadline, you should give your best understanding of what is not working, and how you might go about fixing it.

- If you use an algorithm that is not fully described in the assignment text, you should try to explain it in your own words. This also applies if the method is described elsewhere in the course material.
- In some cases it may suffice to explain your work via comments in the code itself, but other times you might want to include a more elaborate explanation in terms of, e.g., mathematics and/or pseudocode.
- In general, it is a good habit to comment your code (though it can be overdone).
- When working with approximate solutions to equations, it is always useful to check your results against known exact (analytical) solutions, should they be available.
- It is also a good test of a model implementation to study what happens at known 'edge cases'.
- Any figures you include should be easily understandable. You should label axes appropriately, and depending on the problem, include other legends etc. Also, you should discuss your figures in the main text.
- It is always good if you can reflect a little bit around *why* you see what you see.

# References

[1] Hongbin Zhan. Analytical and numerical modeling of a double well capture zone. *Mathematical Geology*, 31(2):175–193, 1999.