Can you hear the size of a reservoir?

MOD510: Mandatory project 2

Deadline: 9. October 2022 (23:59)

Sep 20, 2023

Learning objectives. By completing this project, the student will:

- Implement numerical solvers for the radial diffusivity equation.
- Use the solvers to compute pressure as a function of distance from a well, both at steady-state conditions and for transient flow.
- For the steady-state case, compare the implemented model to an analytical solution.
- Apply the time-dependent model to study pressure decline in a well, and estimate the size of a reservoir from well test data.
- Investigate time efficiences of sparse versus dense matrix solvers.

Read this before starting to code!

In this project you will implement pressure solvers for one-phase fluid flow in porous media. The project starts with theory to motivate and provide background for understanding the physics of fluid flow. However, the solution method we will employ is very similar to what you have already seen for the heat equation. Therefore, **you do not have to understand everything in order to solve the exercises**; for a quick start, jump to section 3.2 and continue from there. The Appendix contains some **practical coding tips**, as well as additional equations you may need for the project.

In the later parts of the project you will have to keep track of many variables and physical units. To stay organized, we **strongly recommend** that you code your pressure solvers into a class. Preferably, you should implement the steady-state solver in Exercise 1 as a special case of the more general time-dependent solver (Exercise 2).

Regardless of how you choose to implement the solvers, you should try to limit code duplication by breaking your code into several smaller functions. A single "top-level" function can be implemented to execute all the necessary steps in order. **Remember to give all functions and classes descriptive names**.

1 Are abandoned reservoirs going to save us?

Reservoirs are most likely going to be very important in the future as carbon capture and storage (CCS) is viewed as one solution to the global warming challenge. Oil and gas is produced, and then the CO_2 can be re-injected in the reservoir as illustrated in figure 1, or CO_2 can be captured directly from the air as done in Iceland.

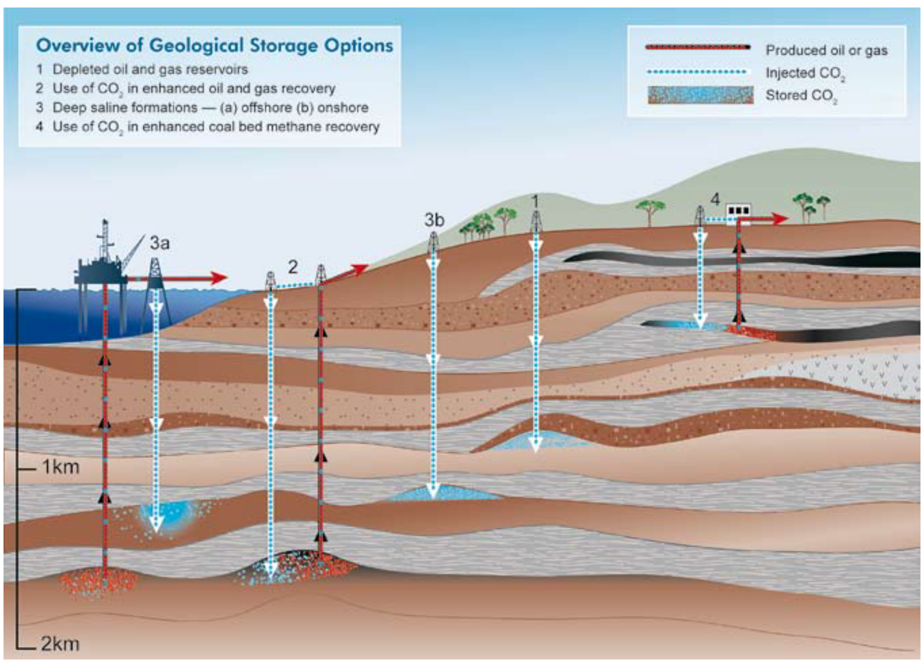

Figure 1: Different CO_2 storage options from [11].

The Intergovernmental Panel on Climate Change (IPCC) claims that the potential of CCS is "considerable". In Norway this has manifested in a Governmental project called "Longship". Part of the Longship project is the Equinor's, Northern light project, which aims to capture CO2 from industries and store it on the NCS, see figure 2

The "Longship" project in Norway was, when it was launched, called The greatest climate project in Norwegian industry ever by the Norwegian Ministry of Petroleum. Norway's position as an oil producing nation and more then 50 years of experience with reservoirs at the NCS, put Norway in a very good position to develop CCS as a viable solution.

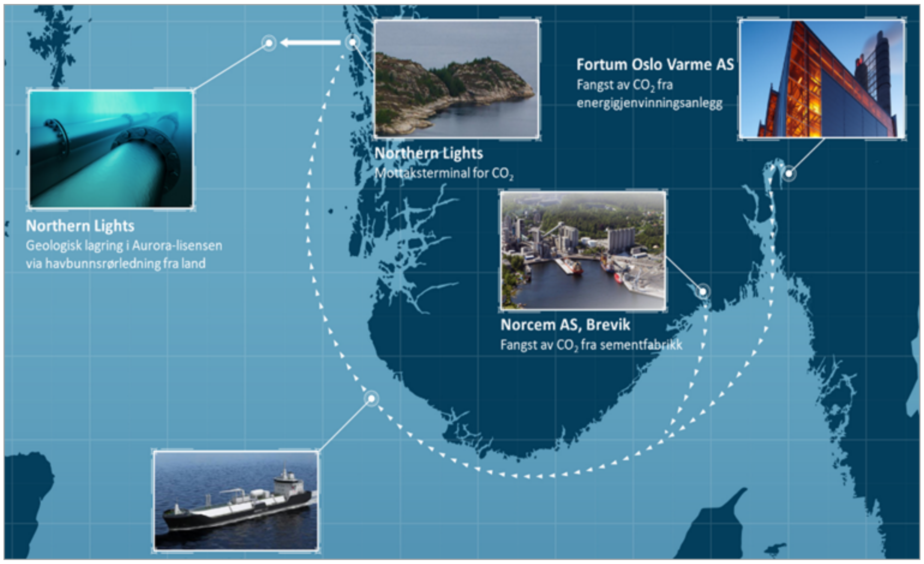

Figure 2: Plans to capture CO2 from Breivik cement factory and Fortum Oslo Varme's waste in Oslo (credit: Gassnova).

In this project we are going to investigate models that are able to tell us about reservoir properties by monitoring how pressure is changing when fluids are injected into them. You will see how models can be used to extract information about a reservoir during operation. This is important as reservoirs are dynamic objects, their properties change during the lifetime.

2 What are reservoirs, and how do fluids move within them?

Reservoirs are large pieces of porous rock that are buried underground. A sedimentary rock consists of many small grains, like sand on a beach. During geological time, the grains have become cemented together, but still there is plenty of room for oil, water, and gas to occupy the pores in-between the grains: Usually, 20-30 % of the total reservoir volume is filled with fluids, but sometimes the porosity can be as large as 50 %.

Fluids are extracted from a reservoir by drilling one or more production wells. Before opening up for production it is important to quantify the *absolute permeability*, which is a quantitative measure of how easily fluids flow through the rock. The absolute permeability plays the same role as electric conductivity in electrodynamics [4]: A high permeability means that fluids flow easily through the rock, just as a highly conductive (low resistance) material transmits electric current easily. What actually drives fluid flow is the *pressure gradient*, ∇p. If we continue the analogy with electricity, ∇p plays the role of the voltage gradient inside a conductor. The magnitude of the pressure gradient depends on both the rock and fluid properties [1]. Initially, pressure differences are established from the natural energy of the reservoir itself. Additional pressure support may be achieved by injecting fluids, typically water and/or gas [5].

During production, it is useful to monitor whether the permeability changes with time. This is because injection or production of fluids may cause clogging of pores close to the wells, e.g, as a result of chemical reactions taking place there. If this happens, the permeability will be reduced, which may cause excessive pressure build-up [8]. Another factor to consider is *reservoir compaction*. As fluids are produced, the pressure in the formation drops, which could lead to large increases in the effective stress in the reservoir, and to seabed subsidence. This happened at the Ekofisk field in the North Sea, where platforms sank by several meters [13], see figure 3 Similar observations have been made in the city of Venice, which is currently sinking. Part of the recorded subsidence has been attributed to groundwater pumping operations [14].

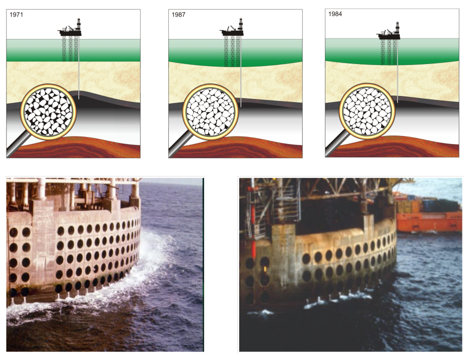

Figure 3: A brief history of the Ekofisk compaction from start of production in 1971, fluids were removed and as a consequence the grains were pushed closer together and the platform sank.

A key tool to avoid the above-mentioned problems is *well pressure testing* [9], which is the topic of this project. It turns out that by studying the time-dependence of well pressures during production, one can learn a lot about a reservoir. This information is clearly extremely relevant for an oil company, but it is of equal importance in geothermal applications, and during groundwater monitoring. Reservoirs are huge ($\sim$ km size), and the well is only about half a foot, but still the information coming from a single producing well can provide valuable information, e.g., estimates for the size of the reservoir, and whether formation damage has likely occurred.

3 Background theory

3.1 Derivation of the diffusivity equation

To derive an equation that describes fluid flow in a reservoir, the starting point is, as always, the *continuity equation.* The system we will consider is illustrated in figure 4. By applying a mass balance to the dotted volume shown in the figure, we can derive a one-dimensional version of the continuity equation [6]:

$$\frac{\partial q(x,t)}{\partial t}A(x) = -\frac{\partial (J(x,t)A(x))}{\partial x} + \dot{\sigma}(x,t)A(x)\,. \tag{1}$$

In the above expression, A is the cross-sectional area at position x, q denotes the mass concentration of fluid, J is the amount of mass flowing into the volume per unit time per unit area (the *mass flux*), and $\dot{\sigma}$ is a source term that describes the amount of mass of fluid *generated* per unit volume per unit time.

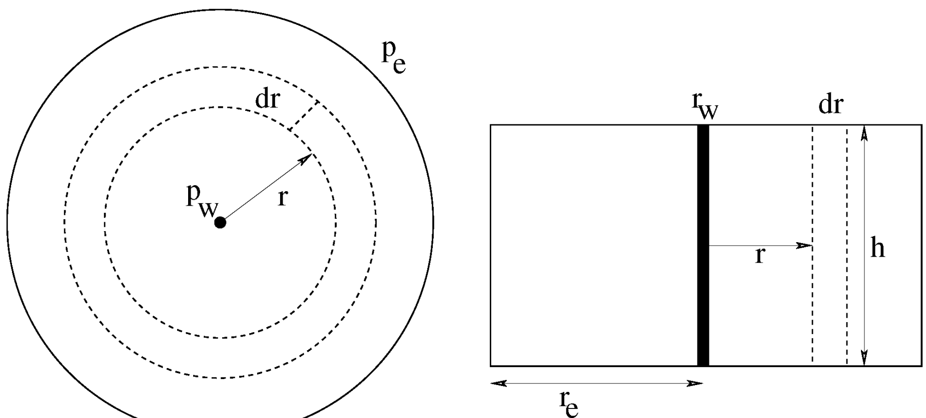

Figure 4: A vertical well placed in the middle of a cylindrical reservoir of constant thickness, h. The reservoir is the volume between the well radius at $r = r_w$ and the exterior radius at $r = r_e$. Left plot: Top view of the reservoir. Right plot: A view from the side.

For the situation depicted in figure 4, fluid flow is radially symmetric, hence we can make the following identifications:

$$x \to r\,, \tag{2}$$
$$A(x) \to A(r) = 2\pi r h\,. \tag{3}$$

Since there are no wells inside the porous medium (only at the inner boundary, $r = r_w$), we have $\dot{\sigma} = 0$. Furthermore, since

$$\text{mass flux} = \frac{\text{mass}}{\text{area}\cdot\text{time}} = \frac{\text{mass}\cdot\text{length}}{\text{volume}\cdot\text{time}} = \text{density}\cdot\text{velocity}\,, \tag{4}$$

we see that $J(r) = \rho u_r$, where ρ is the fluid density, and u_r is the (radial) fluid velocity. The mass concentration term is given by $q = \phi\rho$, where ϕ is the porosity of the porous medium. We have to multiply density with porosity because fluids can only flow inside the void space in-between the grains of the rock.

By inserting all of the above, equation (1) can be reformulated into:

$$
\begin{aligned}
2\pi r h \frac{\partial}{\partial t}(\phi\rho) &= -\frac{\partial}{\partial r}(2\pi r h \rho u_r) \, , \\
\frac{\partial}{\partial t}(\phi\rho) &= -\frac{1}{r}\frac{\partial}{\partial r}(r\rho u_r) \, . \qquad (5)
\end{aligned}
$$

There are some important steps left:

1. We have to relate fluid velocity to pressure. Here, we will use an empirical law named after the French water engineer, Henry Darcy [2]. This equation is fundamental for everyone that studies groundwater flow.

2. The fluid density appears on both sides of equation (5), but we would like to relate fluid flow to *pressure* and not *density*. Fortunately we can express fluid density in terms of pressure by introducing a *fluid compressibility*-factor, c_f. If pressure increases, the density of the fluid increases.

3. On the left-hand side, the time derivative of the porosity enters. The porosity might also change when the fluid pressure changes (e.g., increasing pressure can lead to rock compaction). This can be accounted form by introducing a *rock compressibility*-factor, c_r.

For radially symmetric flow, Darcy's law [7] takes the form

$$
u_r = -\frac{k}{\mu}\frac{\partial p}{\partial r} \, , \qquad (6)
$$

where k is the absolute permeability of the rock, p is fluid pressure, and μ is fluid viscosity. We will not show the full derivation of how to account for compressibility. However, by assuming that the fluid compressibility is low (this is the case for water [3]), and introducing the *total compressibility* $c_t = c_f + c_r$, it is possible to derive the *diffusivity equation*:

$$
\frac{\partial p}{\partial t} = \eta \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial p}{\partial r}) \, , \qquad (7)
$$

where η is the *hydraulic diffusivity*:

$$
\eta = \frac{k}{\mu\phi c_t} \, . \qquad (8)
$$

Interpretation of the diffusivity equation.

The diffusivity equation describes how pressure waves travel in a porous rock. The factor η can be thought of as a diffusion constant, in analogy with molecular diffusion. Whenever you have a diffusion equation, you can use the relation

$$\text{diffusion constant} \approx \frac{\text{length}^2}{4 \cdot \text{time}} .$$

to give a rough estimate of the speed of the process. For example, if $\eta = 2.5\ \text{m}^2/\text{s}$ and the reservoir radius is 1 km, it will take approximately

$$\frac{(1000\ \text{m})^2}{4 \cdot 2.5\ \text{m}^2/\text{s}} \simeq 1.2\ \text{days}$$

for the pressure wave to reach the outer boundaries of the reservoir.

3.2 The radial flow equations including boundary conditions

Before attempting to find a solution to equation (7), we need boundary conditions. According to Darcy's law, the volumetric flow rate at the well bore is

$$Q = -\left(u \cdot A\right)\Big|_{r=r_w} = \frac{2\pi hkr}{\mu}\frac{\partial p}{\partial r}\Big|_{r=r_w} , \tag{9}$$

where we have adopted the sign convention that $Q > 0$ for production of fluids. In this project it will be assumed that Q is constant. At the exterior boundary, we require a constant fluid pressure,

$$p\left(r = r_e\right) = p_{\text{init}} , \tag{10}$$

where p_{init} is the initial fluid pressure in the reservoir. Before implementing the numerical solution, we perform a (clever) coordinate change $r \to y$, where y is

$$y = y(r) \equiv \ln \frac{r}{r_w} . \tag{11}$$

See figure 5, for a visual image of the effect of this transformation. In the new y-coordinate, equations (7), (9), and (10) are transformed into

The equations you will need in the rest of the project.

$$\frac{\partial p}{\partial t} = \eta \frac{e^{-2y}}{r_w^2} \frac{\partial^2 p}{\partial y^2}\,, \tag{12}$$

$$\frac{\partial p}{\partial y}\left(y = y_w\right) = \frac{Q\mu}{2\pi h k}\,, \tag{13}$$

$$p\left(y = y_e\right) = p_{\text{init}}\,. \tag{14}$$

Note: In order to plot the pressure vs the *physical coordinate* r, you will have to invert equation (11)

$$r = r_w e^y \tag{15}$$

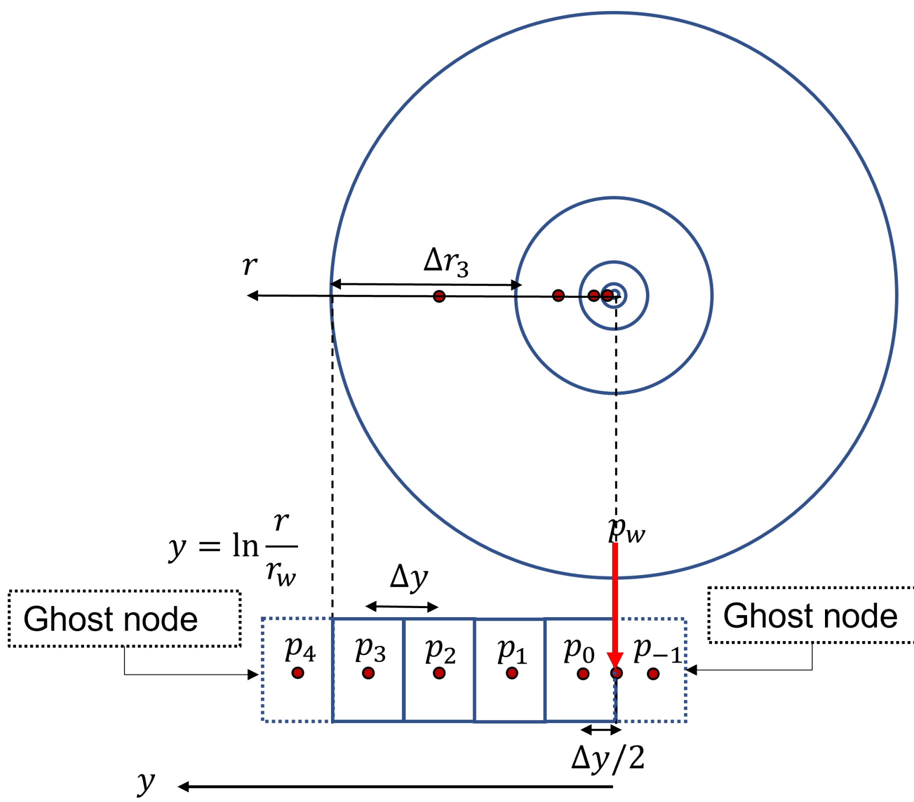

Figure 5: Sketch of the coordinate transformation $r \to y$. Note that the pressure is always evaluated at the center of the y-blocks.

4 Exercise 1: Steady-state solution

The simplest situation is when there is no pressure-variation in time, i.e., steady-state flow. Then, we can replace the partial derivatives with ordinary derivatives, and equations (12), (13), and (14) become:

$$\frac{d^2p}{dy^2}(y) = 0 \text{ for all y}\,, \tag{16}$$

$$\frac{dp}{dy}\,(y = y_w) = \alpha \tag{17}$$

$$p(y = y_e) = p_{\text{init}}\,. \tag{18}$$

> **Notice.**
>
> In exercise 1 we want to focus on numerical implementation aspects. Therefore, we simply set $\alpha \equiv Q\mu/2\pi hk = 1$. Later in the project, you **will have to use physical input parameters** for Q, μ, h, and k in order to match data.

Part 1.

- Show that the analytical solution to equations (16), (17), and (18) is

$$p(y) = p_{\text{init}} + \alpha(y - y_e)\,. \tag{19}$$

Part 2. To enforce a fixed pressure $p_e = p(y_e) = p_{\text{init}}$ at the edge of the reservoir, the "lazy" option for the exterior reservoir boundary is to simply set

$$p_N = p_e\,. \tag{20}$$

Figure 5 illustrates the unknown pressure values for the case $N = 4$.

- Let $N = 4$. When using the "lazy" implementation of the boundary condition at $y = y_e$, derive the following matrix equation, starting from the general finite difference equations for any N.

$$\begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} \alpha\Delta y \\ 0 \\ 0 \\ -p_e \end{pmatrix}. \tag{21}$$

Part 3. What is the truncation error for the finite difference approximation at interior grid points? (be as specific as you can!)

Part 4. We want to investigate, theoretically, the error of the "lazy" approximation to the pressure boundary condition. Use Taylor's formula to

- Find how the order of the numerical error scales when using the "lazy" approximation.
- Use Taylor's formula to show that we can derive the following, "not-so-lazy" version of the boundary condition:

$$p_N = 2p_e - p_{N-1}\,. \tag{22}$$

- Let $N = 4$. What is the matrix equation we now need to solve when using equation (22) for the outer boundary condition?

Part 5.

- For both implementations of the boundary condition at $y = y_e$, solve the matrix equation multiple times by varying the number of grid points.
- Based on your simulation results, make a scatter plot of the numerical error versus grid size. Ideally, you should evaluate the different solutions at *the same physical point* in space. Use the analytical formula given by equation (19) as the "true solution".
- Does the error scale as you expect? Discuss.

Note: You may have to think a little bit on how to calculate the error. As you change the grid resolution, the coordinates also change...

5 Exercise 2: Time-dependent solution

To capture how pressure changes in time, we go back to the original diffusivity equation. We apply the following *implicit time-discretization*

Implicit scheme for interior grid point i.

$$\frac{p_i^{n+1} - p_i^n}{\Delta t} = \eta \cdot \frac{e^{-2y_i}}{r_w^2} \cdot \frac{p_{i+1}^{n+1} + p_{i-1}^{n+1} - 2p_i^{n+1}}{\Delta y^2} \tag{23}$$

Part 1.

- Show that for the special case $N = 4$, the matrix equation we need to solve each time step is:

$$
\underbrace{\begin{pmatrix} 1+\xi_0 & -\xi_0 & 0 & 0 \\ -\xi_1 & 1+2\xi_1 & -\xi_1 & 0 \\ 0 & -\xi_2 & 1+2\xi_2 & -\xi_2 \\ 0 & 0 & -\xi_3 & 1+3\xi_3 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} p_0^{n+1} \\ p_1^{n+1} \\ p_2^{n+1} \\ p_3^{n+1} \end{pmatrix}}_{\mathbf{p}^{n+1}} = \underbrace{\begin{pmatrix} p_0^{n} \\ p_1^{n} \\ p_2^{n} \\ p_3^{n} \end{pmatrix}}_{\mathbf{p}^{n}} + \underbrace{\begin{pmatrix} -\beta\xi_0 \\ 0 \\ 0 \\ 2p_i\xi_{N-1} \end{pmatrix}}_{\mathbf{d}}, \tag{24}
$$

where we have defined

$$\xi_i \equiv \frac{\eta e^{-2y_i}\Delta t}{r_w^2 \Delta y^2},$$

and

$$\beta \equiv \frac{Q\mu\Delta y}{2\pi kh}.$$

Part 2. Again let $N = 4$. Assume default model input parameters (see Appendix B), and that $\Delta t = 0.01$ day.

- Show that the matrix is (in SI units):

$$
\begin{pmatrix}
5.28702460e+03 & -5.28602460e+03 & 0.00000000e+00 & 0.00000000e+00 \\
-9.42633218e+01 & 1.89526644e+02 & -9.42633218e+01 & 0.00000000e+00 \\
0.00000000e+00 & -1.68095582e+00 & 4.36191165e+00 & -1.68095582e+00 \\
0.00000000e+00 & 0.00000000e+00 & -2.99757363e-02 & 1.08992721e+00
\end{pmatrix}
$$

Part 3.

- Implement a simulator that solves the time-dependent problem for any choice of input parameters.

For tips on how to get started, Appendix C.

6 Exercise 3: Accuracy and performance of time-dependent solution

Part 1.

- For several values of N, compare your numerical solver implementation to the *line-source solution* given by equation (27). To do the comparison, you need to plot your solution in terms of the *physical coordinates*, i.e., $r(y) = r_w e^y$.

Note: The line-source solution is only valid at intermediate times, when reservoir boundary effects are negligible.

Part 2. Next, we want to take advantage of the symmetry of the problem. At run-time, the simulator should be able to choose between three different matrix solvers:

1. Dense, using `numpy.linalg.solve`.
2. Sparse using, `scipy.sparse.linalg.spsolve`.
3. Sparse, using the Thomas algorithm). An implementation of the Thomas algorithm can be found in appendix D.
4. Use the `%timeit` option in Jupyter to compare the speed of each solver. How large must N be in order to see a difference?

7 Exercise 4: Match model to well test data

In the final exercise we are going to study data from a well test. During a well test, the production engineer starts to produce from the reservoir, while monitoring how well pressure changes in time.

Part 1. So far, we have calculated the pressure distribution *inside the reservoir*. The actual observable well pressure is missing from our calculations, but we can estimate it by discretizing equation (13).

- Use a first-order finite difference approximation to find a formula for the well pressure in terms of the well block pressure, p_0.
- In your final delivery, make sure that your numerical simulator includes a function to calculate the well pressure as a function of the well-block pressure.

Hint: Use Taylor's formula with step-size $\Delta y/2$.

Part 2. Well test data are available in the text file `well_bhp.dat` (located in the `data` folder).

- Read the well test data into Python, and make a scatter plot of well pressures versus time.

Part 3. Towards the end of the test, we see that the well pressure stabilizes towards a constant value. This indicates that the pressure wave has reached the edge of the reservoir.

For this part you may assume default model input (Appendix B) for all parameters except the following three: k, p_i, and r_e.

- Fit your numerical model to the well test data by changing the values of k, p_i, and r_e.
- Make a plot in which you compare 1) the well test data, 2) your numerical well pressure solution, and 3) the corresponding line-source solution.
- Use a logarithmic scale on the x-axis.

Hints:

- You may try to match the well test curve manually, but it might be easier to use automated curve-fitting.

Part 4.

- Based on the value you found for r_e, what is the total volume of water in the reservoir?

A Line-source solution

There are several transient solutions to the diffusivity equation, but most of them are very difficult to derive and implement [10]. To simplify, it is common to use the *line-source solution*, which is based on slightly different boundary conditions than we have assumed in this project:

- It considers the well to be infinitely small (i.e. a line), thus neglecting the influence of the finite well radius.
- The reservoir is assumed to be infinite in extent.
- Instead of assuming a constant pressure at a fixed distance $r = r_e$, the line-source solution assumes a constant pressure at infinity.

In mathematical terms, the boundary conditions for the line-source solution are:

$$\lim_{r\to 0}\left(\frac{2\pi hk}{\mu} r \frac{\partial p}{\partial r}\right) = Q_{well}\,, \tag{25}$$

and

$$\lim_{r\to\infty} p(r,t) = p_i\,. \tag{26}$$

The solution to equations (7), (25), and (26) is [12]

$$p(r,t) = p_i + \frac{Q\mu}{4\pi kh}\cdot \mathcal{W}\left(-\frac{r^2}{4\eta t}\right)\,, \tag{27}$$

where $\mathcal{W}$ is the *exponential function*, defined in terms of an integral:

$$\mathcal{W}(x) = \int_{-\infty}^{x} \frac{e^u}{u}\, du\,. \tag{28}$$

In Python this function can easily be computed with the `sp.special.expi` function.

B Table of default model input parameters

name	symbol	default value	unit
Well radius	rw	0.318	ft
Outer reservoir boundary	re	1000	ft
Height of reservoir	h	11	ft
Porosity	phi	0.25	dimensionless
Fluid viscosity	mu	1.0	mPas (cP)
Total (rock+fluid) compressibility	ct	7.80E-06	1/psi
Constant flow rate at well	Q	1000	bbl/day
Absolute permeability	k	500	mD
Initial reservoir pressure	pi	4100	psi

C Example code to get you started

```
class PressureSolver:
    """
    A finite difference solver to solve pressure distribution in
    a reservoir, logarithmic grid has been used, y = ln(r/rw)

    The solver uses SI units internally, while "practical field units"
    are required as input.

    Input arguments:

        name                                symbol      unit
        --------------------------------------------------------------
        Number of grid points                N          dimensionless
        Constant time step                   dt         days
        Well radius                          rw         ft
        Outer reservoir boundary             re         ft
        Height of reservoir                  h          ft
        Absolute permeability                k          mD
        Porosity                             phi        dimensionless
        Fluid viscosity                      mu         mPas (cP)
        Total (rock+fluid) compressibility   ct         1 / psi
        Constant flow rate at well           Q          bbl / day
        Initial reservoir pressure           pi         psi
        -------------------------------------------------------------
    """

    def __init__(self,
                 N,
                 dt,
                 rw=0.318,
                 re=1000.0,
                 h=11.0,
                 phi=0.25,
                 mu=1.0,
                 ct=7.8e-6,
                 Q=1000.0,
                 k=500,
                 pi=4100.0):

        # Unit conversion factors (input units --> SI)
        self.ft_to_m_ = 0.3048
```

```
        self.psi_to_pa_ = 6894.75729
        self.day_to_sec_ = 24.*60.*60.
        self.bbl_to_m3_ = 0.1589873

        # Grid
        self.N_ = N
        self.rw_ = rw*self.ft_to_m_
        self.re_ = re*self.ft_to_m_
        self.h_ = h*self.ft_to_m_

        # Rock and fluid properties
        self.k_ = k*1e-15 / 1.01325 # from Darcy to m^2
        self.phi_ = phi
        self.mu_ = mu*1e-3 # from cP to Pas
        self.ct_ = ct / self.psi_to_pa_

        # Initial and boundary conditions
        self.Q_ = Q*self.bbl_to_m3_ / self.day_to_sec_
        self.pi_ = pi*self.psi_to_pa_

        # Time control for simulation
        self.dt_ = dt*self.day_to_sec_

        # TO DO: Add more stuff below here....
        # (grid coordinates, dy, eta, etc.)
```

D Thomas Algorithm

The Thomas algorithm can be implemented quite easily in python, its speed is greatly improved if you use Numba, below you can find a possible implementation.

```
import numba as nb
import numpy as np

@nb.jit(nopython=True)
def thomas_algorithm(l, d, u, r):
    """
    Solves a tridiagonal linear system of equations with the Thomas-algorithm.

    The code is based on pseudo-code from the following reference:

        Cheney, E. W., & Kincaid, D. R.
        Numerical mathematics and computing, 7th edition,
        Cengage Learning, 2013.

    IMPORTANT NOTES:
        - This function modifies the contents of the input vectors l, d, u and rhs.
        - For Numba to work properly, we must input NumPy arrays, and not lists.

    :param l: A NumPy array containing the lower diagonal (l[0] is not used).
    :param d: A NumPy array containing the main diagonal.
    :param u: A NumPy array containing the upper diagonal (u[-1] is not used).
    :param r: A NumPy array containing the system right-hand side vector.
    :return: A NumPy array containing the solution vector.
    """
    # Allocate memory for solution
    solution = np.zeros_like(d)
    n = len(solution)
```

```
    # Forward elimination
    for k in range(1, n):
        xmult = l[k] / d[k-1]
        d[k] = d[k] - xmult*u[k-1]
        r[k] = r[k] - xmult*r[k-1]

    # Back-substitution
    solution[n-1] = r[n-1] / d[n-1]
    for k in range(n-2, -1, -1):
        solution[k] = (r[k]-u[k]*solution[k+1])/d[k]

    return solution
```

E Guidelines for project submission

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 learned 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.
- Avoid duplicating code! If you find yourself copying and pasting a lot of code, it is a strong indication that you should define reuseable functions and/or classes.
- 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 very 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] Laurence Patrick Dake. *Fundamentals of Reservoir Engineering.* Elsevier, 1983.

[2] Henry Darcy. *Les Fontaines Publiques De La Ville De Dijon: Exposition Et Application...* Victor Dalmont, 1856.

[3] Rana A. Fine and Frank J. Millero. Compressibility of water as a function of temperature and pressure. *The Journal of Chemical Physics*, 59(10):5529–5536, 1973.

[4] David J. Griffiths. Introduction to electrodynamics, 2005.

[5] H. Hermansen, GH Landa, JE Sylte, and LK Thomas. Experiences after 10 years of waterflooding the Ekofisk Field, norway. *Journal of Petroleum Science and Engineering*, 26(1-4):11–18, 2000.

[6] Aksel Hiorth. *Computational Engineering and Modeling.* https://github.com/ahiorth/CompEngineering, 2021.

[7] M. King Hubbert. Darcy's law and the field equations of the flow of underground fluids. *Transactions of the AIME*, 207(01):222–239, 1956.

[8] Roland F. Krueger. An overview of formation damage and well productivity in oilfield operations: an update. In *SPE California Regional Meeting.* Society of Petroleum Engineers, 1988.

[9] John Lee, John B. Rollins, and John Paul Spivey. *Pressure Transient Testing.* Richardson, Tex.: Henry L. Doherty Memorial Fund of AIME, Society of Petroleum Engineers, 2003.

[10] Charles Sedwick Matthews and Donald G. Russell. *Pressure Buildup and Flow Tests in Wells*, volume 1. Henry L. Doherty Memorial Fund of AIME New York, 1967.

[11] Bert Metz, Ogunlade Davidson, Heleen De Coninck, et al. *Carbon Dioxide Capture and Storage: Special Report of the Intergovernmental Panel on Climate Change.* Cambridge University Press, 2005.

[12] HJ Ramey. Application of the line source solution to flow in porous media - A review. In *SPE-AIChE Joint Symposium*. OnePetro, 1966.

[13] AM Sulak and J. Danielsen. Reservoir aspects of Ekofisk subsidence. In *Offshore Technology Conference*. Offshore Technology Conference, 1988.

[14] Luigi Tosi, Pietro Teatini, and Tazio Strozzi. Natural versus anthropogenic subsidence of Venice. *Scientific reports*, 3(1):1–9, 2013.