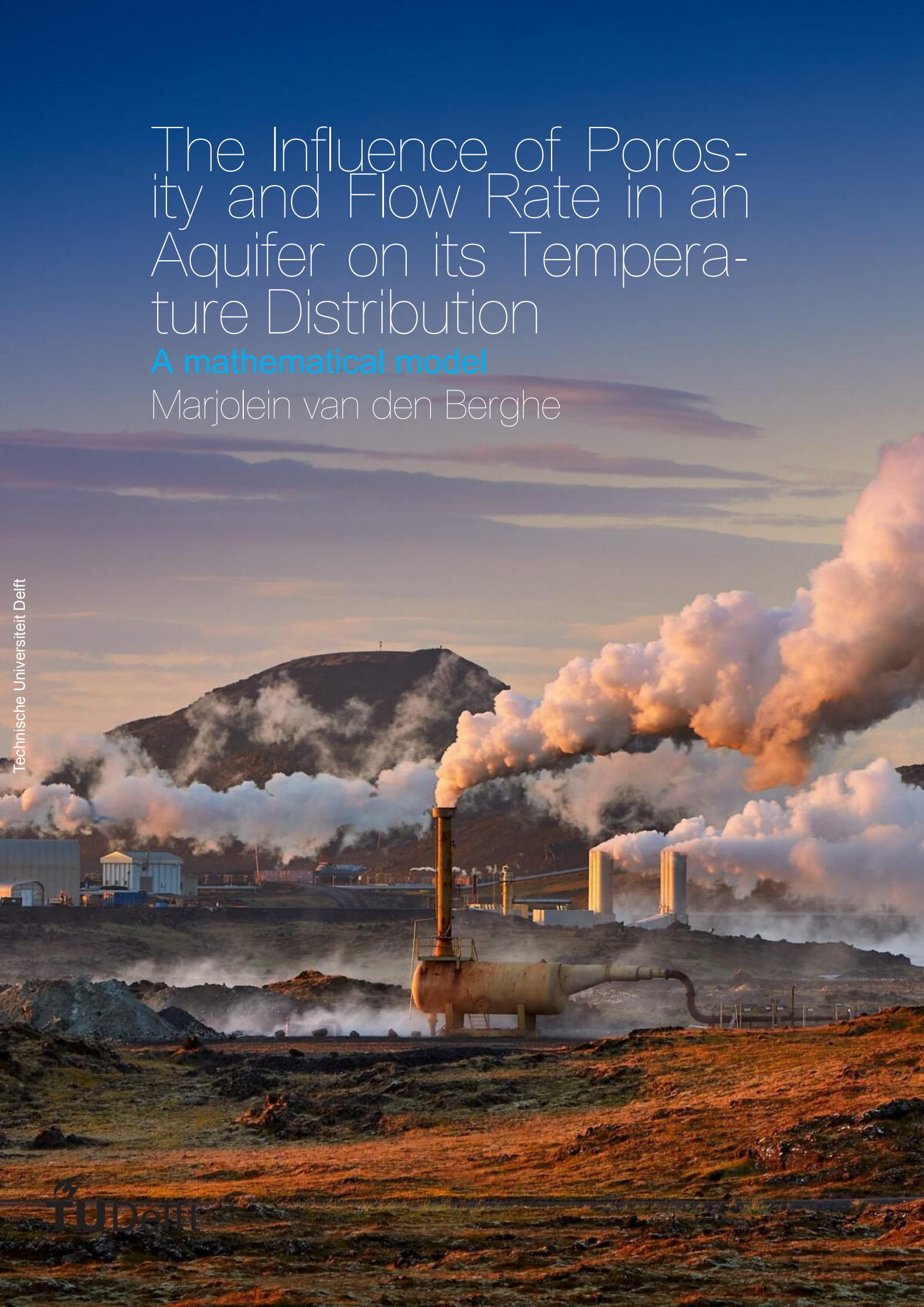
The Influence of Porosity and Flow Rate in an Aquifer on its Temperature Distribution

A mathematical model

Marjolein van den Berghe



Technische Universiteit Delft

The Influence of Porosity and Flow Rate in an Aquifer on its Temperature Distribution

A mathematical model

by

Marjolein van den Berghe

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Friday October 20, 2023 at 10:00 AM.

Student number:	5087724	
Date:	October 13, 2023	
Supervisors:	Prof. dr. ir. C. R. Kleijn,	TU Delft, Applied Sciences
	Dr. B. J. Meulenbroek,	TU Delft, Applied Mathematics

TU Delft Delft University of Technology

Abstract

Geothermal energy can be promising to help the energy transition. Warm water is pumped up from a geothermal well and after it is cooled down, the cold water is reinjected into the hot aquifer underground. The temperature change could cause chemical balances to be disturbed and result in precipitation on the porous medium of the aquifer. This process of clogging of an aquifer is not desirable since it reduces the efficiency of the production of warm water. This research is a stepping stone for modelling these reactions and clogging of the aquifer. The reactions rates depend on the temperature and the concentration of the reactants in the aquifer. For this reason the temperature distribution of the aquifer was numerically derived with finite difference methods. The velocity field is important for modelling both the temperature and the concentrations. Therefore the pressure and velocity field were modelled with use of mass conservation and the Darcy model. The models were solved numerically with finite difference methods and some results were compared to the analytical solution of the model. It was concluded that the numerical model for the velocity field is accurate when compared to analytical solutions of the model equations. In the model used, an increase in porosity resulted in a delay of cooling the aquifer. This is because there was no relation between porosity and permeability implemented in the model. A lower flow rate in the aquifer delays the cooling of the aquifer as well. For further research it is recommended to implement the chemical reactions into the model and use this model to look into how to prevent the clogging in the aquifer.

Contents

List of symbols

Variable	Symbol	Unit
porosity	ϕ	-
Darcy velocity	$\vec{u}$	$\frac{m}{s}$
flow rate	u_0	$\frac{m}{s}$
permeability	k	m^2
viscosity	μ	Pas
pressure	p	Pa
mass	m	kg
density	ρ	$\frac{kg}{m^3}$
volume	V	m^3
surface area	A	m^2
mass flow	Φ	$\frac{kg}{s}$
time	t	s
heat energy density	e	$\frac{J}{m^3}$
specific heat for water	c_w	$\frac{J}{kgK}$
temperature	T	K
convective heat flux density	$\vec{\hat{\phi}}_c$	$\frac{W}{m^2}$
diffusive heat flux density	$\vec{\hat{\phi}}_d$	$\frac{W}{m^2}$
normal unit vector	$\hat{n}$	-
thermal conductivity	λ	$\frac{W}{mK}$
thermal diffusivity	a	$\frac{m^2}{s}$
heat transfer coefficient	h	$\frac{W}{m^2K}$
total heat transfer coefficient	U	$\frac{W}{m^2K}$
diffusion flux density	$\vec{q}_d$	$\frac{mol}{m^2s}$
convective flux density	$\vec{q}_c$	$\frac{mol}{m^2s}$
effective thermal diffusivity	α	$\frac{m^2}{s}$
correction factor for convective heat transport	γ	-
diffusion coefficient	D	$\frac{m^2}{s}$
concentration	c	$\frac{mol}{m^3 water}$

1

Introduction

Geothermal energy is a promising renewable energy source that can be used in the energy transition to dampen climate change. The use of geothermal energy is not new (it is more than 2000 years old), but there is a high increase in usage during the last decade. With respect to 2015, thermal power provided by geothermal installations increased 52% in 2020, resulting in the prevention of 252,6 million tonnes of CO_2 emissions per year worldwide [1].

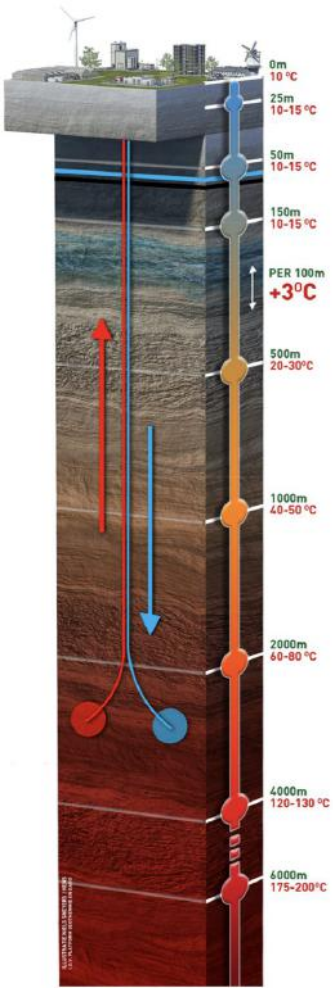


Figure 1.1: Geothermal doublet

A geothermal installation retrieves warm water from an aquifer, a water bearing layer underground, via a geothermal well. The warm water from the production well can be used to heat houses and other buildings, which makes the gas that is currently used for this purpose unnecessary.

The geothermal installation that will be considered in this research works via two wells some space apart from each other, a geothermal doublet (figure 1.1). One well pumps warm water out of the subsurface and the second well pumps cold water into the porous medium in the subsurface. During this process of retrieving warm water, the water in the aquifer cools down. Chemical balances of minerals in the water are disturbed by the cooling and it could be possible that settlements occur. These settlements decrease the porosity and therefore the permeability of the porous medium (figure 1.2) resulting in a higher energy use by the pumps that pump the water in and out of the aquifer. At some point, more energy would be needed for the pumping than the amount of energy that is retrieved from the water. We would like to prevent this from happening and have a better understanding of these processes. However, the interaction between the cold and warm water is deep underground, which makes measurements difficult.

A lot of research has already been done in the field to monitor these processes. One part of the problem is to model the flow in the aquifer. Wood et al [2] determined the pore-scale flow in a packed bed of spherical beads experimentally via particle image velocimetry (PIV) and numerically. The technique of PIV is to track the movement of small particles via illuminated planes. Both methods generated similar results. Another part of the problem is to model the temperature distribution. Jarrahi et al. [3] found a new way to model heat transfer in media which contains fractures. The fractures were represented in the model as porous media with higher permeability. A lot of researches have investigated the clogging of an aquifer and the influence on the permeability of the medium. The transport of fines and clogging due to this transport were modelled by Kanimozhi et al. [4]. Porosity-permeability relations were investigated by Schulz et al. [5]. Litvinenko et al. [6] used random variables for physical properties to account for the heterogeneity in the medium. Yang et al. [7] studied the change of permeability coefficients

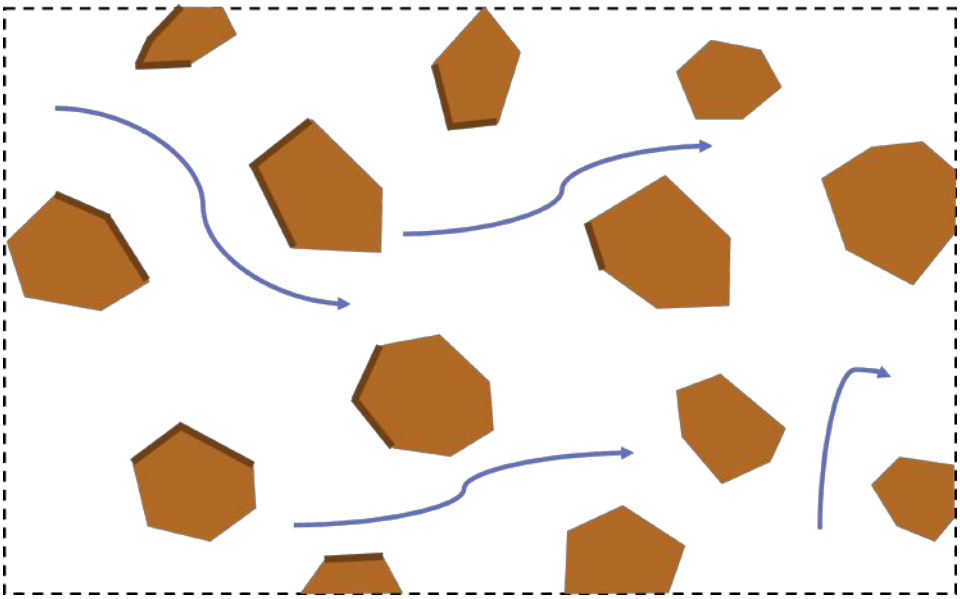


Figure 1.2: Settlements (dark region) on the porous medium (light region) decrease the porosity of the medium. The arrows represent the flow of water through the medium.

and related chemical processes in an indoor sand column. Other geothermal installations than a doublet installation were researched as well. Yin et al. [8] investigated a new method in a coaxial open loop where no water was taken from the aquifer, only heat. A study performed an analysis on the effect of using multiple wells on the cost and lifetime of the geothermal installation [9].

There has been a lot of research on the relation between porosity and permeability. Unfortunately not so much on how the temperature indirectly influences the permeability. For this reason this research will develop and analyse a model describing the velocity field and temperature distribution underground. The model for the velocity field uses Darcy model. The velocity field and temperature distribution are essential in modelling the chemical processes. This research investigates how the temperature distribution is influenced by the porosity of the medium and the flow rate of the injection well. This is a start in analysing clogging due to chemical reactions and finding answers on how to prevent the decrease of permeability in the aquifer.

Chapter 2 describes the derivation of a model for the pressure and velocity field as well as the temperature distribution and concentration of dissolved substances. Chapter 3 will continue with the discretization of the model, followed by a stability analysis. The accuracy of the model is determined by comparing to analytical solutions derived in chapter 4. Chapter 5 presents the results of this accuracy analysis and the numerical velocity field and temperature distribution. This is followed by the conclusion chapter 6.

2

Model derivation

In this chapter we will formulate the model equations that describe fluid flow and transport of heat and chemical species in an aquifer, which will later be solved numerically (chapter 3) and analytically (chapter 4) to validate the numerical approach. We start with explaining the domain of interest and introducing relevant physical properties (section 2.1). The Darcy model will be introduced (section 2.2), which relates the velocity field to the pressure field. After this explanation the model equations are derived from conservation laws together with boundary and initial conditions. With use of the law of mass conservation the model equations for the velocity field are derived and related to the model equations for the pressure field via the Darcy model (section 2.3). From the conservation law of energy the heat equation is derived (section 2.4). In section 2.5 the temperature of the water is related to the temperature of the porous medium from which the model equations for the temperature distribution follow. The model equations for the concentration of dissolved substances follow again from the law of mass conservation (section 2.6). An overview of the relation between the different models is visualized and explained in section 2.7.

2.1. Model of the geometric domain of interest

We will consider an enclosed aquifer some kilometres underground and take a 2D horizontal domain with length L and width H (figure 2.1). We assume that the domain is repeated in the direction of y. The third dimension is averaged into this 2D domain. The aquifer consists of a porous medium saturated with water. The physical properties of the porous medium are of importance for this research. The first property is the porosity ϕ of the medium, indicating which fraction of the total volume is occupied by water. The second one is the permeability k of the medium, which is a measure of how easy the water can flow through the pores of the medium. The permeability is influenced by the structure, the size of grains in the porous medium and by precipitation due to chemical reactions on the grains. Lastly, the temperature of the ground is of interest. We will also have a look into the properties of the water. This research will not consider the velocities of the water at pore-scale, but at a macroscopic scale, the Darcy velocity. This velocity is the so called superficial velocity, which means it is the flux of water through a cross-section of the medium divided by its area. Since this area is partly covered with the porous medium, the Darcy velocity is lower than the real velocity.

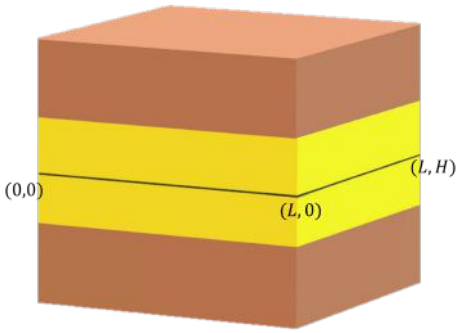


Figure 2.1: Schematic representation of an aquifer (light region) enclosed by two confining beds of low permeability (darker region). The domain of interest is a horizontal rectangular cross-section of the enclosed aquifer, with dimension LxH.

2.2. Darcy model

The pressure gradient in a porous medium is the driving force behind the flowing of water. The equation relating the velocity of the water in porous media with the pressure gradient was discovered by Henry Darcy. Henry Darcy was a French scientist and engineer who lived in the 19th century. In 1834, he designed one of Europe's best water supply systems for its time. This water pipe system in Dijon was solely driven by gravity and used sand as a filter [10]. His knowledge and experience from projects and experiments lead to the finding of the phenomenological Darcy model.

$$\vec{u} = -\frac{k}{\mu}\nabla p, \tag{2.1}$$

with Darcy (or superficial) velocity $\vec{u}$ (m/s), permeability k (m^2), viscosity μ (Pa s) and pressure p (Pa). The Darcy model only holds in the region of very low Reynolds numbers [11], with the Reynolds number based on the water velocity and the pore diameter. Since the water velocity and pore diameter is low, the Darcy model is applicable in this research. The Darcy model will be used in the next section when deriving the model for the pressure field.

2.3. Mass conservation

No water should appear or disappear when flowing through a porous medium, which is stated by the conservation law of mass. With use of the mass conservation law and divergence theorem we will derive an equation for the velocity field. The meaning of the symbols can be found in the list of symbols (page 1).

We start with introducing an expression for the total mass of water m in a small volume ΔV (2.2). The system consists of porous medium and water. The porosity ϕ indicates which fraction of the volume is occupied by water.

$$m = \iiint_{\Delta V} \rho\phi dV \tag{2.2}$$

The mass flow Φ is the amount of water entering the system per time and ρ is the density. Applying the divergence theorem we get the followin expression for the mass flow,

$$\Phi = -\oiint_S \rho\vec{u}\cdot\hat{n}\, dS = -\iiint_{\Delta V} \nabla\cdot\rho\vec{u}\, dV. \tag{2.3}$$

The only change in total mass could be due to mass flow. Differentiating (2.2) with respect to time and setting it equal to (2.3) results in,

$$\frac{dm}{dt} = \frac{d}{dt}\iiint_{\Delta V}\rho\phi dV = \iiint_{\Delta V}\frac{\partial}{\partial t}\rho\phi dV = -\iiint_{\Delta V}\nabla\cdot\rho\vec{u}dV = \Phi. \tag{2.4}$$

From this expression we extract,

$$\frac{\partial}{\partial t}\rho\phi = -\nabla\cdot\rho\vec{u}. \tag{2.5}$$

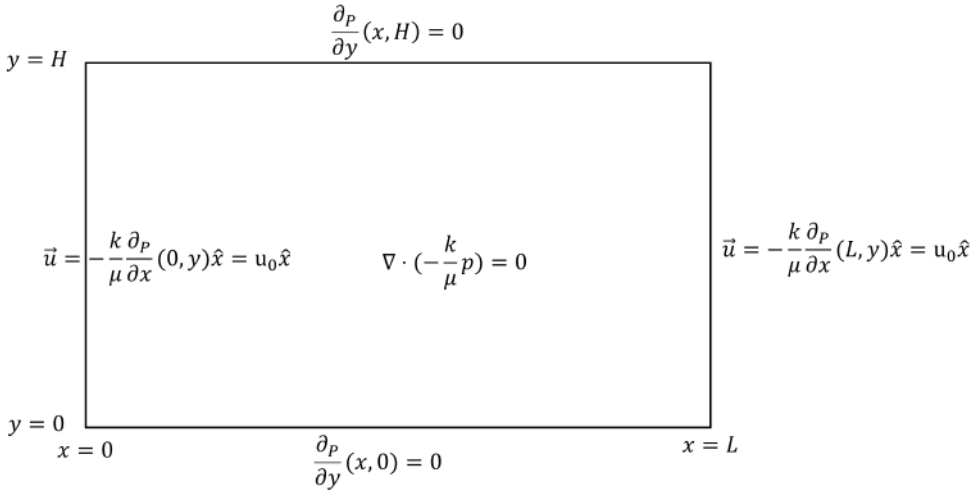


Figure 2.2: Visual representation of model equations for the pressure field p.

In this way we have found an expression for the velocity field to ensure that the mass is conserverd.

We assume a constant density ρ and porosity ϕ. Let the water flow into the domain at $x = 0$ with the same velocity u_0 it flows out of the domain at $x = L$. Furthermore we assume no water flows through the boundaries at $y = 0$ and $y = H$. From these assumptions and equations (2.1) and (2.5), we find our model equations (2.6).

$$\begin{cases} \nabla\cdot\vec{u}=0 \\ \vec{u}=-\frac{k}{\mu}\nabla p \\ \vec{u}(x,0)=\vec{u}(x,H)=-\frac{k}{\mu}\frac{\partial p}{dy}(x,0)=-\frac{k}{\mu}\frac{\partial p}{\partial y}(x,H)=0 \\ \vec{u}(0,y)=\vec{u}(L,y)=-\frac{k}{\mu}\frac{\partial p}{\partial x}(0,y)=-\frac{k}{\mu}\frac{\partial p}{\partial x}(L,y)=u_0 \end{cases} \tag{2.6}$$

Model equations 2.6 can be rewritten for the pressure field (2.7), when you substitute the second model equation into the first model equation. These equations are visualized in figure 2.2.

$$\begin{cases} \nabla\cdot\left(-\frac{k}{\mu}\nabla p\right)=0 \\ \frac{\partial p}{\partial y}(x,0)=\frac{\partial p}{\partial y}(x,H)=0 \\ \frac{\partial p}{\partial x}(0,y)=-\frac{u_0}{\frac{k}{\mu}(0,y)} \\ \frac{\partial p}{\partial x}(L,y)=-\frac{u_0}{\frac{k}{\mu}(L,y)} \end{cases} \tag{2.7}$$

2.4. Heat equation

After deriving the model for the velocity and pressure field (2.6, 2.7) we will continue with the temperature distribution described by the heat equation. We will consider two ways to transport heat: conduction (or diffusion) and convection. Starting with an energy balance over a small volume and with use of Fourier's law we will derive the heat diffusion-convection equation.

Volumetric heat energy density e is linearly related to temperature.

$$e(x,y,t)=c_w\rho T(x,y,t), \tag{2.8}$$

with heat energy density e (J/m^3), specific heat for water c_w $(J/(kg\cdot K))$, density ρ (kg/m^3) and temperature T (K). The specific heat c_w can be considered constant in the temperature range of interest. Water with energy density e flows into the small volume with rate $\vec{u}$. Therefore equation (2.8) can be used to find the convection flux density $\vec{\hat{\phi}}_c$ (2.9).

$$\vec{\hat{\phi}}_c=e\vec{u}=c_w\rho T\vec{u} \tag{2.9}$$

Fourier's law (2.10) relates the diffusion flux density $\vec{\phi}_d$ phenomenologically to the temperature gradient.

$$\vec{\phi}_d = -\lambda \nabla T, \tag{2.10}$$

with heat diffusion flux density $\vec{\phi}_d$ (W/m^2), thermal conductivity λ $(W/(m \cdot K))$ and temperature T (K). We calculate the changes to the heat energy density due to the diffusion and convection flux via the divergence theorem over a small volume ΔV (2.11). Here the normal unit vector $\hat{n}$ is defined positive when pointing out of the surface.

$$\Delta V \phi \frac{\partial e}{\partial t} = -\oint_S \vec{\phi}_d \cdot \hat{n} dS - \oint_S c_w \rho T \vec{u} \cdot \hat{n} dS = -\iint_{\Delta V} \nabla \cdot \vec{\phi}_d dV - \iint_{\Delta V} \nabla \cdot (c_W \rho T \vec{u}) dV = \Delta V (\lambda \nabla^2 T - c_w \rho \nabla \cdot (T\vec{u})) \tag{2.11}$$

From equation (2.11) we find the heat equation by dividing by $\Delta V c_w \rho \phi$ and introducing $a = \frac{\lambda}{c_w \rho}$ (m^2/s).

$$\frac{\partial T}{\partial t} = \frac{1}{c_w \rho} \frac{\partial e}{\partial t} = \frac{a}{\phi} \nabla^2 T - \frac{1}{\phi} \nabla \cdot (T\vec{u}). \tag{2.12}$$

2.5. Temperature coupling

Technically, there are two temperatures to be considered, the temperature of the water T_w and the temperature of the porous medium T_s. We will show that the heat transfer between the two materials is fast with respect to the time scale of the model. From a heat balance over a volume with initial temperature T_0 and surroundings with temperature T_1, the following expression for the average grain temperature $\langle T \rangle$ is derived by [12],

$$\frac{T_1 - \langle T \rangle}{T_1 - T_0} = exp\left(\frac{-6U}{\rho_s c_s d} t\right). \tag{2.13}$$

Here, U is the total heat transfer coefficient, combining the heat transferred from the grain and the heat transferred inside the grain and d is the diameter of the grain. U is not constant throughout the cooling process, therefore equation (2.13) is not an exact solution. An expression for the total heat transfer coefficient U is given in equation (2.14).

$$\frac{1}{U(t)} = \frac{1}{h_i(t)} + \frac{1}{h_e} = \frac{d}{Nu_i(t)\lambda_s} + \frac{d}{Nu_e \lambda_w} \approx \frac{d}{6,6\lambda_s} + \frac{d}{2\lambda_w}, \tag{2.14}$$

with λ_s and λ_w the thermal conductivities of the grain and the water (table 2.1).

Table 2.1: Material properties of water and sand (quartz) [13]

Variable	Value	Units
λ_w	0.6	$W/(mK)$
ρ_w	$1 \cdot 10^3$	kg/m^3
c_w	$4.18 \cdot 10^3$	$J/(kgK)$
ϕ	0.2 [14]	-
λ_s	1.1	$W/(mK)$
ρ_s	$2.6 \cdot 10^3$	kg/m^3
c_s	$0.8 \cdot 10^3$	$J/(kgK)$

From equations (2.13) and (2.14), it is approximated that for a grain of sand it takes $t \approx 0.1\ s$ to cool down when $d = 1.8 \cdot 10^{-4}\ m$ [14] and for a slightly larger $d = 1.7 \cdot 10^{-3}\ m$, $t \approx 4\ s$. The cooling of the sand will thus be very fast compared to the time needed for the water to stream past the grain ($t_0 = 5.4\ s$ and $t_0 = 54\ s$, respectively). Therefore the temperature of the porous medium and the temperature of the water can be approximated to be the same, denoted by T. Nevertheless, the heat diffusivity coefficients of the materials differ and we construct two heat equations, for the temperature of the water (2.15) and the medium (2.16). In the porous medium no convective heat transport will take place.

$$\frac{\partial T_w}{\partial t} = \frac{a_w}{\phi}\nabla^2 T_w - \frac{1}{\phi}\nabla\cdot(T_w\vec{u}) + \frac{hA}{c_w\rho_w\phi V}(T_s - T_w) \tag{2.15}$$

$$\frac{\partial T_s}{\partial t} = \frac{a_s}{1-\phi}\nabla^2 T_s - \frac{hA}{c_s\rho_s(1-\phi)V}(T_s - T_w) \tag{2.16}$$

From these two equations and the approximation $T_w \approx T_s$, we construct one equation (2.17) by multiplying the two equations with $c_w\rho_w\phi$ and $c_s\rho_s(1-\phi)$, respectively and then adding them.

$$(c_w\rho_w\phi + c_s\rho_s(1-\phi))\frac{\partial T}{\partial t} = (c_W\rho_w a_w + c_S\rho_s a_s)\nabla^2 T - c_w\rho_w\nabla\cdot(T\vec{u}) \tag{2.17}$$

Division by $(c_w\rho_w\phi + c_s\rho_s(1-\phi))$ and using the definition of the heat diffusivity coefficient a, we derive a modified heat equation that takes the heat transport of both the water and the porous medium into account,

$$\frac{\partial T}{\partial t} = \frac{\lambda_w + \lambda_s}{c_w\rho_w\phi + c_s\rho_s(1-\phi)}\nabla^2 T - \frac{c_w\rho_w}{c_w\rho_w\phi + c_s\rho_s(1-\phi)}\nabla\cdot(T\vec{u}) =: \alpha\nabla^2 T - \gamma\nabla\cdot(T\vec{u}). \tag{2.18}$$

We introduced the effective thermal diffusivity α and the correction factor for convective heat transport γ. The domain initially has a high temperature T_{high}. At the boundary $x = 0$, water is pumped into the domain with a low temperature T_{low}. No heat transport is assumed to take place at $y = 0$ and $y = H$, since the domain is repeated at higher and lower values of y. The Péclet number $\frac{\gamma u_x \Delta x}{\alpha}$ tells that the heat transport is dominated by convection and diffusion plays a much smaller role. The boundary at $x = L$ is set to conduct no heat, as conduction (or diffusion) already is a small portion of the heat transport. This results in the following boundary and initial conditions,

$$\begin{cases} T(0,y,t) = T_{low} \\ \frac{\partial T}{\partial x}(L,y,t) = 0 \\ \frac{\partial T}{\partial y}(x,0,t) = 0 \\ \frac{\partial T}{\partial y}(x,H,t) = 0 \\ T(x>0,y,0) = T_{high} \end{cases} \tag{2.19}$$

2.6. Dissolved substances

In order to model chemical reactions it is important to know the concentration c of the dissolved substances that are part of the reactions. Here we will derive an equation describing the concentration of these substances. The change in concentration of dissolvents in the water depends on diffusion, convection and potential reactions with other materials. The derivation of this equation is analogous to the derivation of the heat equation. We use the law of mass conservation for the concentration as opposed to the law of thermal energy conservation for the temperature. Fick's law for diffusion replaces Fourier's law for conduction to find the diffusion flux density $\vec{q}_d$,

$$\vec{q}_d = -D\nabla c. \tag{2.20}$$

The convection flux density is given by the mass of dissolvent that flows through a cross-section with Darcy velocity $\vec{u}$,

$$\vec{q}_c = c\vec{u}. \tag{2.21}$$

The change in mass inside a small volume ΔV is found with the divergence theorem (2.22).

$$\Delta V\phi\frac{\partial c}{\partial t} = -\oint_S(\vec{q}_d + \vec{q}_c)\cdot\hat{n}dS - r\phi\Delta V = -\iint_{\Delta V}\nabla\cdot(-D\nabla c + c\vec{u})dV - r\phi\Delta V = \Delta V(D\nabla^2 c - \nabla\cdot(c\vec{u}) - r\phi) \tag{2.22}$$

We find the model equation for the concentration of dissolved substances by dividing by $\Delta V\phi$.

$$\frac{\partial c}{\partial t} = \frac{D}{\phi}\nabla^2 c - \frac{1}{\phi}\nabla\cdot(c\vec{u}) - r, \tag{2.23}$$

with D the diffusion coefficient (m^2/s) and r the reaction speed ($mol/m^3 s$). Similar to the heat equation, we expect no flux of dissolvents at the boundaries $y = 0$ and $y = H$. At $x = 0$ the concentration of the dissolved substance of interest is given by the injection concentration c_{inj}. Furthermore, at the start of the injection, the concentration of dissolved substance in the porous domain equals c_{med}.

$$\begin{cases} c(0, y, t) = c_{inj} \\ \frac{\partial c}{\partial x}(L, y, t) = 0 \\ \frac{\partial c}{\partial y}(x, 0, t) = 0 \\ \frac{\partial c}{\partial y}(x, H, t) = 0 \\ c(x > 0, y, 0) = c_{med} \end{cases} \tag{2.24}$$

2.7. Overview of the derived model

A visual summary on how the model equations derived in this chapter relate to one another can be found in figure 2.3. The velocity field (2.6) and pressure field (2.7) are related to each other via the Darcy model (2.1). The velocity field influences the temperature distribution (2.18, 2.19) and the concentration of the dissolvents (2.23, 2.24). Chemical reactions of the dissolvents depend on the temperature distribution and the concentration of the dissolvents, and in their turn influence the concentrations. Chemical reactions could cause settlements to occur, reducing the porosity and therefore the permeability of the medium (figure 1.2).

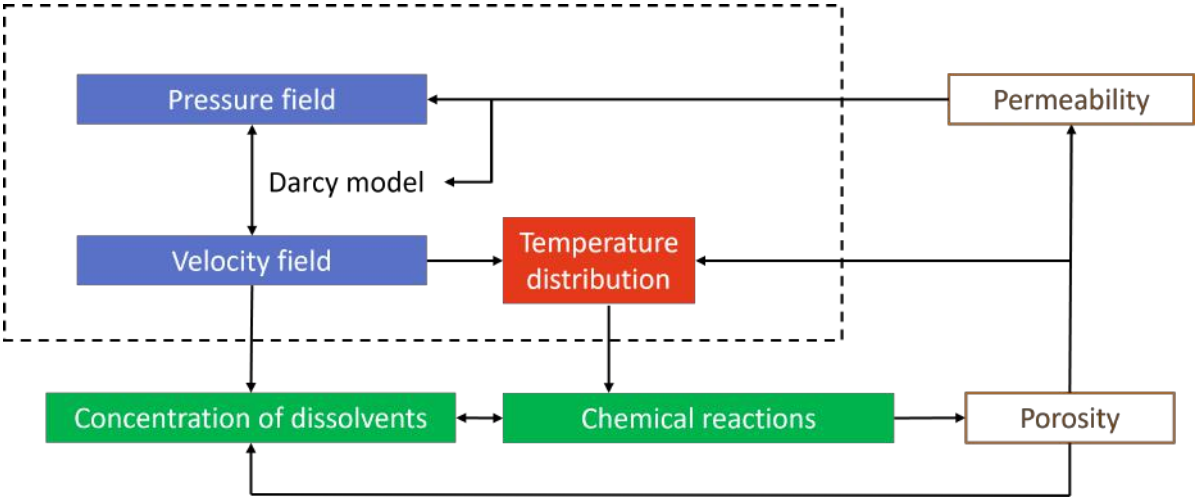


Figure 2.3: Overview of how the model quantities relate to one another. The scope of this research is given by the dashed line.

3

Numerical approximation

The numerical approximations of the model for the pressure field (3.1) and temperature distribution (3.2) are derived. The stability of the numerical approximation for the temperature distribution is analysed as well.

3.1. Pressure field

The model for the pressure field in the ground from equation (2.7) will be solved numerically with use of finite differences. For this purpose the rectangular 2D domain $[0, L]x[0, H]$ will be discretized in rectangles of size $\Delta x \cdot \Delta y$ with $\Delta x = \frac{L}{n_x}$ and $\Delta y = \frac{H}{n_y}$. Here $n_x = 100$ and $n_y = 75$ are the number of grid points in the x-direction and y-direction respectively. Scalar quantities are considered uniform inside a grid cell. The coordinates of the grid points are $[\frac{1}{2}\Delta x + i \cdot \Delta x, \frac{1}{2}\Delta y + j \cdot \Delta y]$ with $i \in [0, n_x - 1]$ and $j \in [0, n_y - 1]$. The model equations are discretized with central differences (appendix A.1) and the resulting linear matrix equation is solved by the direct solver numpy.linalg.solve() [15] from python. When the pressure field is found, the velocities can be derived from the Darcy model (equation 2.1) by approximating the pressure gradient with the difference between two grid points. A staggered grid is used to connect the velocity field with the pressure field, the x-components of the velocity field are calculated at coordinates $[i \cdot \Delta x, \frac{1}{2}\Delta y + j \cdot \Delta y]$ and the y-components at $[\frac{1}{2}\Delta x + i \cdot \Delta x, j \cdot \Delta y]$. This way the velocities have the same index as the pressure point to the right or above. In figure 3.1 the staggered grid is shown with indices to some of the grid points and velocity components.

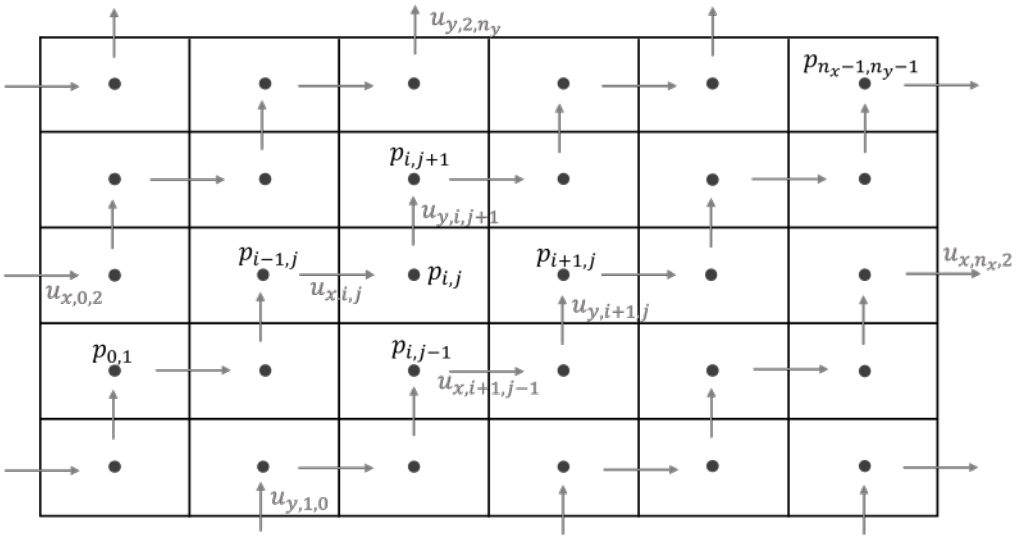


Figure 3.1: Discretization of the domain with pressure p as a scalar quantity and velocity u as vectors.

3.2. Temperature distribution

3.2.1. Numerical approximation

The model for the temperature distribution from equation (2.18) will also be solved with a finite difference method. The same staggered grid (figure 3.1) will be used, with the scalar quantity temperature T on the grid nodes and the velocity components u_x and u_y on the grid cell walls. The time derivative is approximated with the forward Euler method. The diffusion term with central differences and the convection term with upwind differences. The discretization of the model equations can be found in appendix A.2.

3.2.2. Stability

We will check if the chosen numerical method for time integration (forward Euler method) is stable [16]. Solely for this subsection the x-discretization will be indexed by l instead of i, to avoid confusion with the imaginary number i. First we define $\epsilon_{l,j}^k = T_{l,j}^k - T_{l,j}^k$ as the error between the exact solution and the perturbed one. From the discretized heat equation A.16 we find,

$$\epsilon_{l,j}^{k+1} = \frac{\alpha\Delta t}{\Delta x^2}\epsilon_{l+1,j}^k + \left[\frac{\alpha\Delta t}{\Delta x^2} + \frac{\gamma u_{x,l,j}\Delta t}{\Delta x}\right]\epsilon_{l-1,j}^k + \left[1 - 2\frac{\alpha\Delta t}{\Delta x^2} - 2\frac{\alpha\Delta t}{\Delta y^2} - \frac{\gamma u_{x,l+1,j}\Delta t}{\Delta x} - \frac{\gamma u_{y,l,j+1}\Delta t}{\Delta y}\right]\epsilon_{l,j}^k + \frac{\alpha\Delta t}{\Delta y^2}\epsilon_{l,j+1}^k + \left[\frac{\alpha\Delta t}{\Delta y^2} + \frac{\gamma u_{y,l,j}\Delta t}{\Delta y}\right]\epsilon_{l,j-1}^k. \tag{3.1}$$

Substitute for $\epsilon_{l,j}^k = \zeta^k e^{ir_m l\Delta x + is_n j\Delta y}$, where $r_m = \frac{\pi m}{L}$, $m = 1, 2, \ldots, M$ and $M = \frac{L}{\Delta x}$; similarly $s_n = \frac{\pi n}{H}$, with $n = 1, 2, \ldots, N$ and $N = \frac{H}{\Delta y}$. Take $\psi = r_m\Delta x$ and $\eta = s_n\Delta y$,

$$\zeta = 1 + 2\frac{\alpha\Delta t}{\Delta x^2}(cos(\psi) - 1) + 2\frac{\alpha\Delta t}{\Delta y^2}(cos(\eta) - 1) + \frac{\gamma u_x\Delta t}{\Delta x}(e^{-i\psi} - 1) + \frac{\gamma u_y\Delta t}{\Delta y}(e^{-i\eta} - 1), \tag{3.2}$$

where the maximum value of u_x and u_y are given. For absolute stability we require $|\zeta| < 1$ for all values of ψ and η. We take $\psi = \eta = \pi$, since this is the most challenging case. We find a theoretical upper bound for the time step,

$$\Delta t < \frac{\frac{2\alpha}{\Delta x^2} + \frac{2\alpha}{\Delta y^2} + \frac{\gamma u_x}{\Delta x} + \frac{\gamma u_y}{\Delta y}}{\frac{4\alpha^2}{\Delta x^4} + \frac{4\alpha^2}{\Delta y^4} + \frac{8\alpha^2}{\Delta x^2\Delta y^2} + \frac{4\alpha\gamma u_y}{\Delta x^2\Delta y} + \frac{4\alpha\gamma u_x}{\Delta x^3} + \frac{4\alpha\gamma u_x}{\Delta y^2\Delta x} + \frac{4\alpha\gamma u_y}{\Delta y^3} + \frac{\gamma^2 u_x^2}{\Delta x^2} + \frac{\gamma^2 u_y^2}{\Delta y^2} + \frac{2\gamma^2 u_x u_y}{\Delta x\Delta y}}. \tag{3.3}$$

We can compare this result to known conditions on time step Δt for the one dimensional case. We take $\Delta x = \Delta y$ and $u_x = u_y$ Péclet number $\frac{\gamma u_x\Delta x}{\alpha}$,

$$\Delta t < \frac{\frac{4\alpha}{\Delta x^2} + \frac{2\gamma u_x}{\Delta x}}{\frac{16\alpha^2}{\Delta x^4} + \frac{16\alpha\gamma u_x}{\Delta x^3} + \frac{4\gamma^2 u_x^2}{\Delta x^2}} = \frac{\Delta x^2}{2\alpha}\frac{1}{2 + Pe^2}. \tag{3.4}$$

$$\lim_{Pe\to 0} : \Delta t < \frac{\Delta x^2}{4\alpha} \tag{3.5}$$

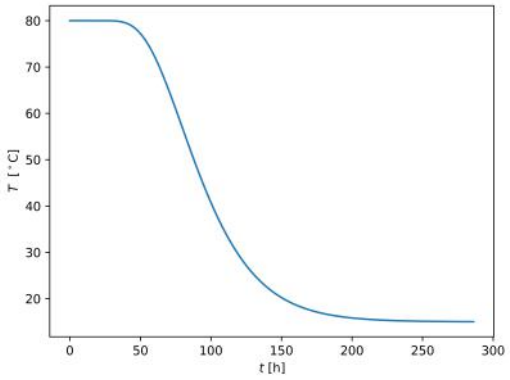
$$\lim_{Pe\to\infty} : \Delta t < \frac{\Delta x^2}{2\alpha Pe} = \frac{\Delta x}{2\gamma u_x} \tag{3.6}$$

The derived conditions on the time step Δt correspond to known conditions for this problem [17], but differ a factor 2. This factor 2 comes from the fact that we simplified a 2D stability condition to a 1D stability condition.

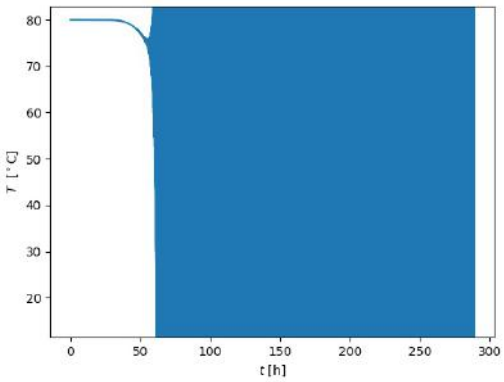
It is interesting to see if the theoretical upper bounds correspond to the upper bounds found when using the code. The results (table 3.1) show that the practical upper bound can be slightly larger than the theoretical upper bound. This was checked for different thermal diffusivity coefficients. Figure 3.2 shows an example of the stability and instability for two time steps.

Table 3.1: Ratio theoretical and practical upper bound for the time step [18] for the heat equation for different values of α, $\gamma = 1.672$, $u_x = 3.17 \cdot 10^{-5}\ m/s$, $u_y = 2.85 \cdot 10^{-18}\ m/s$ $\Delta x = 0.2\ m$ and $\Delta y = 0.15\ m$.

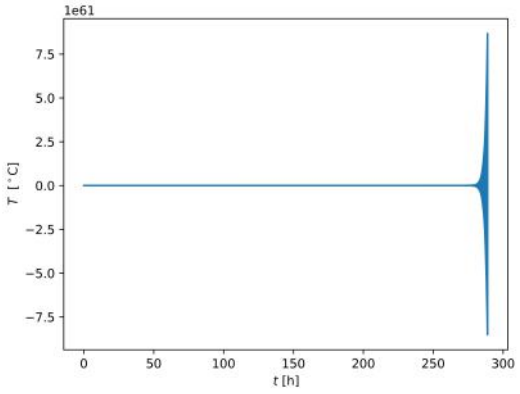
$\alpha\ [m^2/s]$	$\Delta t_{th}\ [s]$	$\Delta t_{pr}\ [s]$	$\frac{\Delta t_{pr}}{\Delta t_{th}}$
$6.8 \cdot 10^{-4}$	10.6	10.6 ± 0.01	1.00 ± 0.01
$6.8 \cdot 10^{-5}$	103	103 ± 1	1.00 ± 0.01
$6.8 \cdot 10^{-6}$	827	847 ± 1	1.024 ± 0.001
$6.8 \cdot 10^{-7}$	2782	3020 ± 10	1.086 ± 0.004



(a) $\Delta t = 103\ s$



(b) $\Delta t = 104\ s$



(c) $\Delta t = 104\ s$

Figure 3.2: Temperature T over time t in the point (20 m, 7.5 m) for different values of Δt. $\alpha = 6.8 \cdot 10^{-5}\ m^2/s$, $\gamma = 1.672$, $u_0 = 3.17 \cdot 10^{-5}\ m/s$

4

Analytical solutions

The aim of this chapter is to find analytical solutions which can be compared with the solutions calculated by the numerical model. From these comparisons the accuracy of the numerical model can be evaluated. The following problems are analytically solved: two 2D nonhomogeneous boundary value problems where the nonhomogeneous boundary conditions are placed at different boundaries (case I and II, sections 4.1 and 4.2) and a 1D transient heat equation (case III, section 4.3).

4.1. Pressure field for case I

We start with the analytical solution of the boundary value problem stated in equation 2.7. Assume $\frac{k}{\mu}$ is constant. Then we are left with a Laplacian on a rectangular domain with two nonhomogeneous Neumann boundary conditions. We look for a characteristic function $r(x, y)$ satisfying the nonhomogeneous boundary conditions, equation (4.1) meets our requirements.

$$r(x, y) = -\frac{u_0 \mu}{k} x \tag{4.1}$$

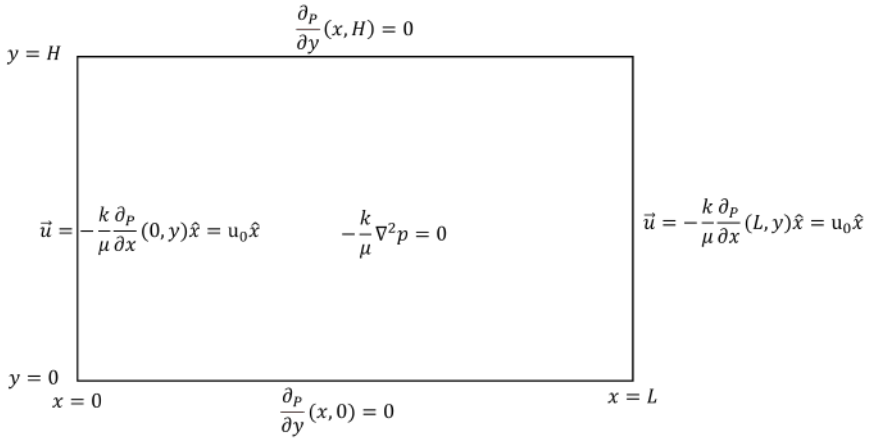


Figure 4.1: Visualization of the 2D boundary value problem of case I for pressure p with nonhomogenous Neumann boundary conditions at $x = 0$ and $x = L$.

We redefine the problem for $v(x,y) := p(x,y) - r(x,y)$,

$$\begin{cases} -\frac{k}{\mu}\nabla^2 v = -\frac{k}{\mu}\nabla^2(p-r) = -\frac{k}{\mu}\nabla^2 p = 0 \\ \frac{\partial v}{\partial x}(0,y) = \frac{\partial p}{\partial x}(0,y) - \frac{\partial r}{\partial x}(0,y) = -\frac{u_0\mu}{k} + \frac{u_0\mu}{k} = 0 \\ \frac{\partial v}{\partial x}(L,y) = \frac{\partial p}{\partial x}(L,y) - \frac{\partial r}{\partial x}(L,y) = -\frac{u_0\mu}{k} + \frac{u_0\mu}{k} = 0 \\ \frac{\partial v}{\partial y}(x,0) = \frac{\partial p}{\partial y}(x,0) - \frac{\partial r}{\partial y}(x,0) = 0 \\ \frac{\partial v}{\partial y}(x,H) = \frac{\partial p}{\partial y}(x,H) - \frac{\partial r}{\partial y}(x,H) = 0 \end{cases} \tag{4.2}$$

We proceed with separation of variables by introducing $v(x,y) = X(x)Y(y)$.

$$\frac{1}{X(x)}\frac{d^2X}{dx^2}(x) = -\frac{1}{Y(y)}\frac{d^2Y}{dy^2}(y) = -\lambda \tag{4.3}$$

We split this equation into two ordinary differential equations for $X(x)$ and $Y(y)$.

$$\begin{cases} \frac{d^2X}{dx^2}(x) + \lambda X(x) = 0 \\ \frac{dX}{dx}(0) = 0 \\ \frac{dX}{dx}(L) = 0 \end{cases} \tag{4.4}$$

From (4.4) we find, $X_n(x) = \cos\left(\frac{n\pi}{L}x\right)$, with $\lambda_n = \left(\frac{n\pi}{L}\right)^2$, for $n = 0, 1, 2, 3....$

$$\begin{cases} \frac{d^2Y(y)}{dy^2} - \lambda Y(y) = 0 \\ \frac{dY}{dy}(0) = 0 \end{cases} \tag{4.5}$$

From (4.5) we find, $Y_n(y) = \cosh\left(\frac{n\pi}{L}y\right)$. The next step is to use superposition of the eigenfunctions,

$$v(x,y) = \sum_{n=0}^{\infty} c_n \cos\left(\frac{n\pi}{L}x\right)\cosh\left(\frac{n\pi}{L}y\right), \tag{4.6}$$

and substitute the last boundary condition into this equation,

$$\frac{dv}{dy}(x,H) = \sum_{n=1}^{\infty} c_n \cos\left(\frac{n\pi}{L}x\right)\frac{n\pi}{L}\sinh\left(\frac{n\pi}{L}H\right) = 0. \tag{4.7}$$

Equation (4.7) holds when $c_n = 0$ for all $n \geq 1$. Only c_0 is undefined and can take up any value. This is expected since the problem only has Neumann boundary conditions. We have found the solution for $v(x,y)$ and from this we find the solution for $p(x,y)$.

$$v(x,y) = c_0 \tag{4.8}$$

$$p(x,y) = v(x,y) + r(x,y) = -\frac{u_0\mu}{k}x + c_0 \tag{4.9}$$

A visualization of the solution is given in figure 4.2.

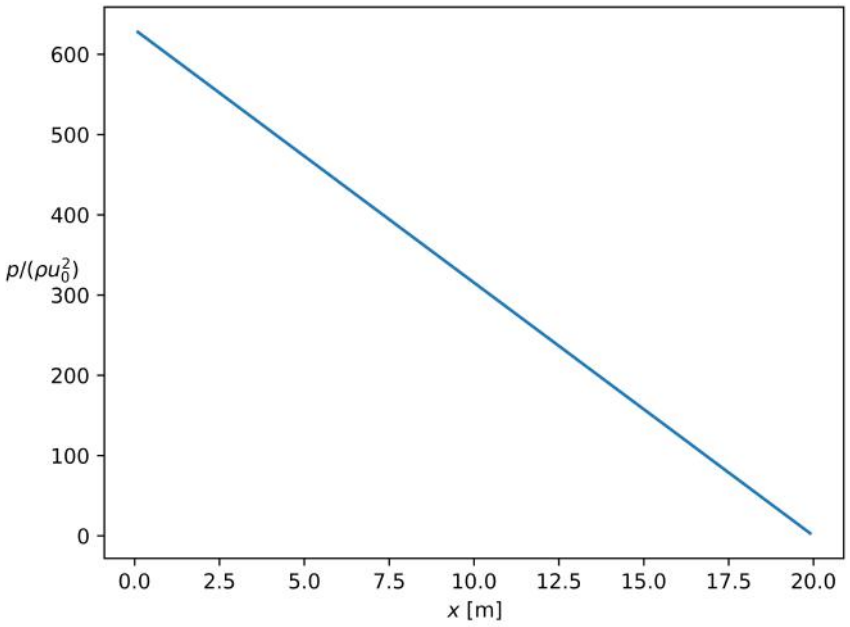


Figure 4.2: Pressure field for case I with $\frac{k}{\mu} = 1\ \frac{m^4}{Ns}$ and $u_0 = 0.0000317\ m/s$ for all $y \in [0, H]$.

4.2. Pressure field for case II

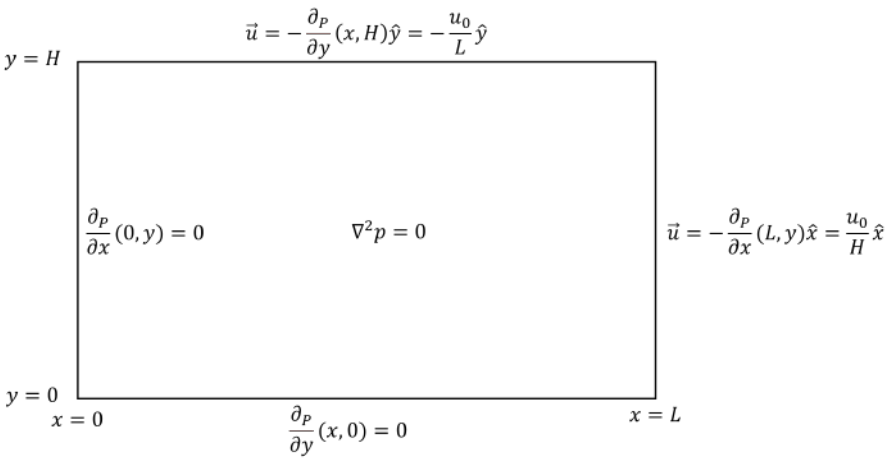


Figure 4.3: Visualization of a 2D boundary value problem with nonhomogenous boundary conditions, where water streams around the corner.

In this section we will derive the analytical solution of case II (figure 4.3). This problem is similar to case I, where the nonhomogeneous boundary conditions are placed at different boundaries. The nonhomogeneous boundary conditions are placed at different boundaries to ensure the solution has a 2D character. The purpose of this derivation is to test the accuracy of the numerical model for a more complex problem. We assume the domain to be homogeneous and therefore the physical properties porosity ϕ, density ρ, permeability k and viscosity μ to be constant in space and time. For simplicity $\frac{k}{\mu}$ is set to be equal to 1. These assumptions translate to the following nonhomogeneous boundary value

problem for pressure p (see also figure 4.3),

$$\begin{cases} \nabla^2 p = 0 \\ \frac{\partial p}{\partial x}(0,y) = 0 \\ \frac{\partial p}{\partial x}(L,y) = -\frac{u_0}{H} \\ \frac{\partial p}{\partial y}(x,0) = 0 \\ \frac{\partial p}{\partial y}(x,H) = \frac{u_0}{L}. \end{cases} \tag{4.10}$$

The general idea of the solution method is to use the method of eigenfunction expansion using Green's formula [19], because this method allows nonhomogeneous boundaries. In the previous section we did not need this method because we could translate the problem to a homogeneous problem with help of a characteristic function. We derive the eigenfunctions of the related homogeneous problem for either x or y. Pressure field $p(x,y)$ is then approximated with an eigenfunction expansion of the found eigenfunctions. The coefficients of the expansion are then calculated using the orthogonality of the eigenfunctions and Green's formula.

We start with rewriting the partial differential equation from equation (4.10) as follows,

$$\frac{\partial^2 p}{\partial x^2} = -\frac{\partial^2 p}{\partial y^2}. \tag{4.11}$$

We choose to use the related homogeneous eigenvalue problem for y, this gives $\phi_n(y) = \cos(\frac{n\pi}{H}y)$, with $\lambda_n = (\frac{n\pi}{H})^2$ and $n = 0, 1, 2, \ldots$. Via eigenfunction expansion we find an expression for $p(x,y)$,

$$p(x,y) \sim \sum_{n=0}^{\infty} b_n(x)\phi_n(y). \tag{4.12}$$

Using the orthogonality of the eigenfunctions ϕ_n we find for $b_n(x)$,

$$b_n(x) = \frac{\int_0^H p(x,y)\phi_n(y)dy}{\int_0^H \phi_n(y)^2 dy}. \tag{4.13}$$

We use the $\sim$ notation in 4.12, because an equality wouldn't hold at $y = H$. This is because $\phi_n(y)$ satisfies the homogeneous boundary condition and $p(x,y)$ does not. This also means that the solution to be found will not satisfy the boundary condition at $y = H$. For this reason we can't take the derivative with respect to y. We can however take the derivative with respect to x in the following way,

$$\frac{\partial^2 p}{\partial x^2} = \sum_{n=0}^{\infty} \frac{d^2 b_n(x)}{dx^2}\phi_n(y) = -\frac{\partial^2 p}{\partial y^2}, \tag{4.14}$$

from which with use of orthogonality of the eigenfunctions follows that,

$$\frac{d^2 b_n(x)}{dx^2} = \frac{-\int_0^H \frac{\partial^2 p}{\partial y^2}\phi_n(y)dy}{\int_0^H \phi_n(y)^2 dy}. \tag{4.15}$$

Applying Green's formula (4.16) results in,

$$\int_0^H \left(p\frac{d^2\phi_n}{dy^2} - \phi_n\frac{\partial^2 p}{\partial y^2}\right)dy = \left(p\frac{d\phi_n}{dy} - \phi_n\frac{\partial p}{\partial y}\right)\Bigg|_0^H, \tag{4.16}$$

$$\int_0^H \left(-p\lambda_n\phi_n - \phi_n\frac{\partial^2 p}{\partial y^2}\right)dy = -\phi_n(H)\frac{\partial p}{\partial y}(x,H) = \frac{u_0}{L}(-1)^{n+1}. \tag{4.17}$$

When rearranging (4.17) and substituting in (4.15) together with (4.13) we find differential equations for $b_n(x)$,

$$\frac{d^2 b_n(x)}{dx^2} = \frac{\int_0^H \lambda_n \phi_n p dy + \frac{u_0}{L}(-1)^{n+1}}{\int_0^H \phi_n^2 dy} = \lambda_n b_n(x) + \frac{\frac{u_0}{L}(-1)^{n+1}}{\int_0^H \phi_n^2 dy}. \tag{4.18}$$

We need boundary conditions at $x = 0$ and $x = L$ to solve these differential equations. We substitute the boundary condition for $p(x, y)$ at (L, y) into the eigenfunction expansion (4.12),

$$\frac{\partial p}{\partial x}(L, y) = \sum_{n=0}^{\infty} \frac{db_n(L)}{dx}\phi_n = -\frac{u_0}{H}. \tag{4.19}$$

Using the orthogonality of the eigenfunctions, we derive the boundary condition for $b_n(x)$ at $x = L$,

$$\frac{db_n(L)}{dx} = \frac{\int_0^H -\frac{u_0}{H}\phi_n dy}{\int_0^H \phi_n^2 dy} = \begin{cases} -\frac{u_0}{H} & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases} \tag{4.20}$$

In a similar way we derive the boundary condition at $x = 0$,

$$\frac{db_n(0)}{dx} = 0. \tag{4.21}$$

Now all the ingredients are found and we can solve the differential equations (4.18) for $b_n(x)$.

Solving the differential equations for $b_n(x)$

The boundary condition for $b_n(x)$ is different for $n = 0$ and $n \neq 0$. Therefore we will solve the differential equations for $n = 0$ and $n \neq 0$, separately.

The differential equation for $n = 0$ is as follows,

$$\begin{cases} \frac{d^2 b_0(x)}{dx^2} & = -\frac{u_0}{LH} \\ \frac{db_0(0)}{dx} & = 0 \\ \frac{db_0(L)}{dx} & = -\frac{u_0}{H}. \end{cases} \tag{4.22}$$

Integrating the differential equation of (4.22) once gives an integration constant which turns out to be 0 when applying the boundary condition at $x = 0$,

$$\frac{db_0(x)}{dx} = -\frac{u_0}{LH}x. \tag{4.23}$$

Integrating (4.23) again gives another constant c_1,

$$b_0(x) = -\frac{u_0}{2LH}x^2 + c_1. \tag{4.24}$$

This constant c_1 will remain arbitrary since we did not specify a reference pressure point when we introduced only Neumann boundary conditions. We have found the solution for $b_0(x)$ (equation 4.24).

Next we solve the differential equations for $n \neq 0$,

$$\begin{cases} \frac{d^2 b_n(x)}{dx^2} - \lambda_n b_n = \frac{2u_0}{LH}(-1)^{n+1} \\ \frac{db_n(0)}{dx} = \frac{db_n(L)}{dx} = 0 \end{cases} \tag{4.25}$$

There are two solutions to the homogeneous part of the problem,

$$b_{n,h_1}(x) = Ae^{\frac{n\pi}{H}x} \tag{4.26}$$

$$b_{n,h_1}(x) = Be^{-\frac{n\pi}{H}x}. \tag{4.27}$$

We will use variation of parameters [20] to solve the nonhomogeneous part of the problem as well. The wronskian of the two found homogeneous solutions equals $-2AB\frac{n\pi}{H}$, from which the particular solution is found,

$$b_{n,p}(x) = \frac{2Hu_0}{n^2\pi^2 L}(-1)^n \tag{4.28}$$

When we substitute $b_n(x) = b_{n,h_1}(x) + b_{n,h_2}(x) + b_{n,p}$ into the differential equations, we see that the found solution indeed solves the differential equations (4.18),

$$\left(\frac{n\pi}{H}\right)^2 (Ae^{\frac{n\pi}{H}x} + Be^{-\frac{n\pi}{H}x}) - \left(\frac{n\pi}{H}\right)^2 (Ae^{\frac{n\pi}{H}x} + Be^{-\frac{n\pi}{H}x} + \frac{2Hu_0}{n^2\pi^2 L}(-1)^n) = \frac{2u_0}{LH}(-1)^{n+1}. \tag{4.29}$$

The next step is to find the coefficients A and B using the boundary conditions (4.20) and (4.21). From (4.21) we find $A = B$ (equation 4.30).

$$\frac{db_n(0)}{dx} = \frac{n\pi}{H}\left(Ae^{\frac{n\pi}{H}0} - Be^{-\frac{n\pi}{H}0}\right) = \frac{n\pi}{H}(A - B) = 0 \tag{4.30}$$

When substituting $A = B$ into the other boundary condition (4.20) we find,

$$\frac{db_n(L)}{dx} = \frac{n\pi}{H}\left(Ae^{\frac{n\pi}{H}L} - Ae^{-\frac{n\pi}{H}L}\right) = 2A\frac{n\pi}{H}\sinh\left(\frac{n\pi}{H}L\right) = 0. \tag{4.31}$$

Since $n \neq 0$, the sinushyperbolicus has no zero and we conclude $A = 0$. From this we have found the solution,

$$p(x,y) = \sum_{n=0}^{\infty} b_n(x)\phi_n(y) \tag{4.32}$$

$$b_n(x) = \begin{cases} -\frac{u_0}{2LH}x^2 + c_1 & n = 0 \\ \frac{2Hu_0}{n^2\pi^2 L}(-1)^n & n \neq 0 \end{cases} \tag{4.33}$$

$$\phi_n(y) = \cos\left(\frac{n\pi}{H}y\right). \tag{4.34}$$

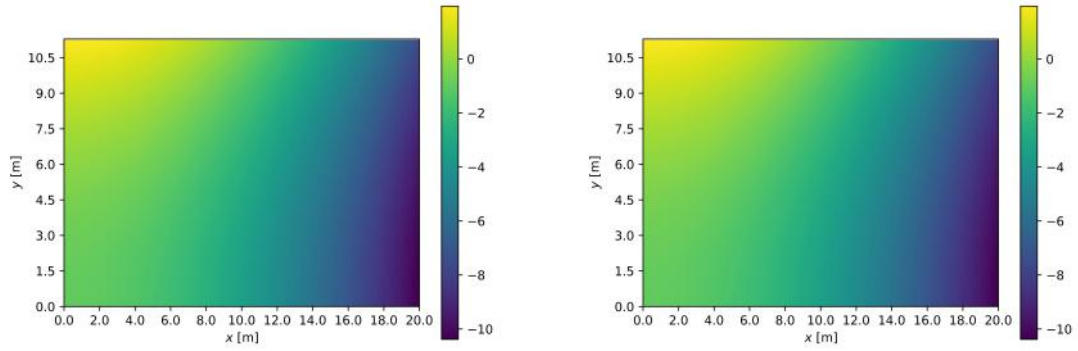
We could have started with taking the homogeneous equivalent boundary value problem for x. Following a similar procedure for this case the solution would have been,

$$p(x,y) = \sum_{n=0}^{\infty} a_n(y)\psi(x) \tag{4.35}$$

$$a_n(y) = \begin{cases} \frac{u_0}{2LH}y^2 + c_1 & n = 0 \\ -\frac{2Lu_0}{n^2\pi^2 H}(-1)^n & n \neq 0 \end{cases} \tag{4.36}$$

$$\psi_n(x) = \cos\left(\frac{n\pi}{L}x\right) \tag{4.37}$$

We see that the solutions (4.32-4.34) are very similar to the solutions (4.35-4.37), the x and y are interchanged as well as H and L and the signs. These differences follow from the difference in the boundary conditions. Visualizations of the solutions are given in figure 4.4. We see that the two solutions generate the same pressure field.



(a) Eigenfunction expansion with $\psi_n(x)$

(b) Eigenfunction expansion with $\phi_n(y)$

Figure 4.4: Pressure fields of the analytical solutions (with N=1000 terms). The reference points of the pressure fields were set equal and at position (0,0). The pressure fields were divided by |p(0,0)|.

4.3. Transient temperature distribution for case III

The 1D version of the diffusion-convection equation (2.18) with general coefficients looks like,

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} - \beta \frac{\partial T}{\partial x} \tag{4.38}$$

The effective thermal diffusivity α is assumed to be positive. The effective Darcy velocity of the water is assumed to be constant and is replaced by β (denoted by $\gamma\vec{u}$ in equation 2.18). If we apply separation of variables, this differential equation would be easier to solve without the convection term. An expression for $T(x,t)$ can be defined to accomplish this [21],

$$T(x,t) = e^{-\frac{\beta^2}{4\alpha}t} e^{\frac{\beta}{2\alpha}x} v(x,t) =: Av(x,t) \tag{4.39}$$

Substituting equation (4.39) into equation (4.38), confirms this claim,

$$\begin{aligned} -\frac{\beta^2}{4\alpha} Av + A\frac{\partial v}{\partial t} &= \alpha\left(\frac{\beta^2}{4\gamma^4} Av + 2\frac{\beta}{2\alpha} A\frac{\partial v}{\partial x} + A\frac{\partial^2 v}{\partial x^2}\right) - \beta\left(\frac{\beta}{2\alpha} Av + A\frac{\partial v}{\partial x}\right) \\ A\frac{\partial v}{\partial t} &= \left(2\frac{\beta^2}{4\alpha} - \frac{\beta^2}{2\alpha}\right) Av + (\beta - \beta) A\frac{\partial v}{\partial x} + \alpha A\frac{\partial^2 v}{\partial x^2} \\ \frac{\partial v}{\partial t} &= \alpha \frac{\partial^2 v}{\partial x^2}. \end{aligned} \tag{4.40}$$

Using this expression for $T(x,t)$ (4.39), we can convert our initial problem (4.41) into a new problem (4.42). This new problem is solely based on diffusion.

$$\begin{cases} \frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} - \beta \frac{\partial T}{\partial x} \\ T(0,t) = 0 \\ \frac{\partial T}{\partial x}(L,t) = 0 \\ T(x>0,0) = T_{high} - T_{low} =: \Delta T \end{cases} \tag{4.41}$$

$$\begin{cases} \frac{\partial v}{\partial t} = \alpha \frac{\partial^2 v}{\partial x^2} \\ v(0,t) = 0 \\ \frac{\beta}{2\alpha} v(L,t) + \frac{\partial v}{\partial x}(L,t) = 0 \\ e^{\frac{\beta}{2\alpha}x} v(x>0,0) = \Delta T \end{cases} \tag{4.42}$$

We use separation of variables, introduce $v(x,t) = X(x)h(t)$ and substitute this expression into equation (4.42),

$$\frac{1}{\alpha h(t)}\frac{dh}{dt}(t) = \frac{1}{X(x)}\frac{d^2X}{dx^2}(x) = -\lambda. \tag{4.43}$$

$$\begin{cases} \frac{d^2X}{dx^2}(x) = -\lambda X(x) \\ X(0) = 0 \\ \frac{\beta}{2\alpha}X(L) + \frac{dX}{dx}(L) = 0 \end{cases} \tag{4.44}$$

We will analyse if λ is positive, zero or negative. Let $\lambda \leq 0$, the result is the following,

$$X(x) = Ae^{\sqrt{-\lambda}x} + Be^{-\sqrt{-\lambda}x} \tag{4.45}$$

$$X(0) = A + B = 0 \tag{4.46}$$

$$\frac{\beta}{2\alpha}X(L) + \frac{dX}{dx}(L) = A\frac{\beta}{2\alpha}\left(e^{\sqrt{-\lambda}L} - e^{-\sqrt{-\lambda}L}\right) + A\sqrt{-\lambda}\left(e^{\sqrt{-\lambda}L} + e^{-\sqrt{-\lambda}L}\right) = 0 \tag{4.47}$$

Equation 4.47 can be rewritten by multiplying with $e^{\sqrt{-\lambda}L}$ into,

$$e^{2\sqrt{-\lambda}L} = \frac{\frac{\beta}{2\alpha} - \sqrt{-\lambda}}{\frac{\beta}{2\alpha} + \sqrt{-\lambda}} \tag{4.48}$$

The left hand side of equation 4.48 must be larger than or equal to 1 since the power is nonnegative. However, the right hand side must be smaller than or equal to 1. Therefore the only solution of equation 4.48 is for $\lambda = 0$, and this results in the trivial solution. We conclude that the eigenvalue problem for $X(x)$ (equations 4.44) only has positive eigenvalues. If $\lambda > 0$, we use goniometric functions to find the solution,

$$X(x) = A\cos(\sqrt{\lambda}x) + B\sin(\sqrt{\lambda}x) \tag{4.49}$$

$$X(0) = A\cos(0) + B \cdot 0 = 0 \tag{4.50}$$

$$\frac{\beta}{2\alpha}X(L) + \frac{dX}{dx}(L) = \frac{\beta}{2\alpha}B\sin(\sqrt{\lambda}L) + B\sqrt{\lambda}\cos(\sqrt{\lambda}L) = 0 \tag{4.51}$$

$$tan(\sqrt{\lambda}L) = -\frac{2\alpha\sqrt{\lambda}}{\beta}. \tag{4.52}$$

Values for λ can be obtained numerically with equation (4.52). Next, the time dependent part of the solution is,

$$\frac{dh}{dt}(t) = -\lambda\alpha h(t) \tag{4.53}$$

$$h(t) = Ce^{-\lambda\alpha t}. \tag{4.54}$$

Superposition of the eigenfunctions gives,

$$v(x,t) = \sum_{n=1}^{\infty} c_n \sin(\sqrt{\lambda_n}x)e^{-\lambda_n\alpha t}, \tag{4.55}$$

and from inserting the initial condition follows,

$$e^{\frac{\beta}{2\alpha}x}v(x>0,0) = e^{\frac{\beta}{2\alpha}x}\sum_{n=1}^{\infty} c_n \sin(\sqrt{\lambda_n}x) = \Delta T. \tag{4.56}$$

If the eigenfunctions $sin(\sqrt{\lambda_n}x)$ are orthogonal, an expression for c_n can be found,

$$c_n = \frac{\int_0^L \Delta T e^{-\frac{\beta}{2\alpha}x}\sin(\sqrt{\lambda_n}x)dx}{\int_0^L \sin^2(\sqrt{\lambda_n}x)dx}. \tag{4.57}$$

If the eigenfunctions are orthogonal, the following expression should be zero for all $n \neq m$,

$$\int_0^L \sin(\sqrt{\lambda_n}x)\sin(\sqrt{\lambda_m}x)dx = \frac{\sqrt{\lambda_m}}{\lambda_n - \lambda_m}\sin(\sqrt{\lambda_n}L)\cos(\sqrt{\lambda_m}L) - \frac{\sqrt{\lambda_n}}{\lambda_n - \lambda_m}\cos(\sqrt{\lambda_n}L)\sin(\sqrt{\lambda_m}L). \tag{4.58}$$

Unfortunately, the eigenfunctions did not seem to be orthogonal (table 4.1). To find the coefficients

Table 4.1: Eigenvalues with use of the root-finder scipy.optimize.bisect() function from python and the inproduct of their eigenfunction with the first eigenfunction.

n	λ_n	$\int_0^L \sin(\sqrt{\lambda_1}x)\sin(\sqrt{\lambda_n}x)dx$
1	0.010	0.240
2	0.059	1.299
3	0.158	1.193
4	0.306	-0.626
5	0.504	0.278
6	0.75	0.203
7	1.046	0.095
8	1.392	0.125
9	1.787	0.038
10	2.231	-0.067

c_n equation (4.57) cannot be used. Instead the interval $[0, L]$ is discretized in $[x_1, x_2, ..., x_i, ..., x_N]$ and for each value of x_i we find an equation from (4.56). These N equations can be written as a matrix equation, which can be solved for c_n with the direct solver numpy.linalg.solve() from python. We find the solution of the 1D convection-diffusion equation (4.38) after substituting (4.55) into (4.39),

$$T(x,t) = \sum_{n=1}^{\infty} c_n e^{-\left(\frac{\beta^2}{4\alpha}+\lambda_n\alpha\right)t} e^{\frac{\beta}{2\alpha}x}\sin(\sqrt{\lambda_n}x), \tag{4.59}$$

with the eigenvalues λ_n derived numerically from equation (4.52) and the coefficients c_n from equation (4.56). In reality, the solution is shifted by T_{low}. The analytical solution is displayed in figure 4.5. One can see that at $t = 0$, the solution is not a neat step function.

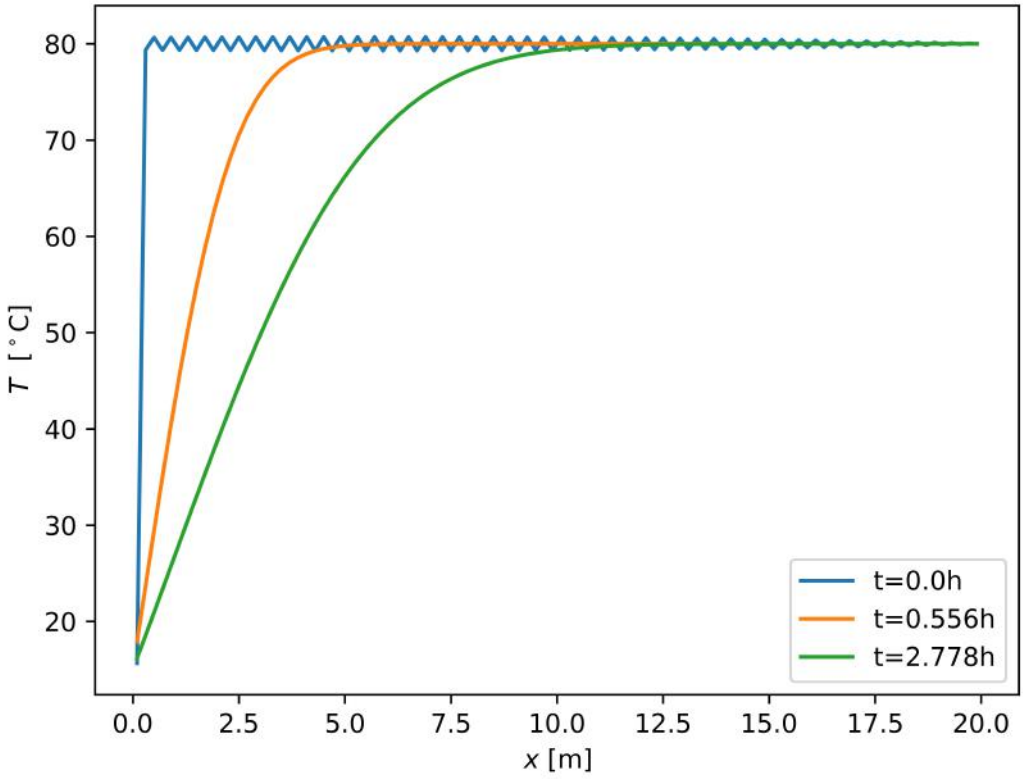


Figure 4.5: Temperature distribution of the analytical solution of the first 100 terms for different times t, where $\alpha \cdot 10^3 = 6.8 \cdot 10^{-4}\ m^2/s$ and $\beta = 5.3 \cdot 10^{-5}\ m/s$.

5

Results

In this chapter the results of the numerical model are presented and discussed. The first part discusses results that describe the accuracy of the numerical model (section 5.1) including a comparison with the analytical model (section 5.2). The second part presents the results of the numerical model (section 5.3), describing the velocity field and temperature distribution.

5.1. Mass conservation of numerical model

One of the two equations that form the basis of the model, is the mass continuity equation (2.5). We expect that the mass conservation holds for the result generated by the numerical model. To check if this is indeed the case, equation 5.1 was used to compare the in- and outflow per grid cell. See figure 3.1 for a visualization of the indices.

$$\text{error mass conservation per cell} = \frac{(u_{x,i,j} - u_{x,i+1,j})\Delta y + (u_{y,i,j} - u_{y,i,j+1})\Delta x}{(|u_{x,i,j}| + |u_{x,i+1,j}|)\Delta y + (|u_{y,i,j}| + |u_{y,i,j+1}|)\Delta x} \cdot 100\% \qquad (5.1)$$

We will consider the case described by equations 2.7 and 2.1 first (case I). A schematic representation of this case is presented in figure 5.1. The mass conservation is checked for different settings for $\frac{k}{\mu}$, namely $\frac{k}{\mu}$ a) is a constant, b) is a smooth function of x, c) has a constant value or d) a smoothly varying value everywhere except of a rectangular subdomain where it has a constant smaller value. The results for case I for these different settings, are shown in figure 5.2. It is clear that the error is very small and of the order of machine accuracy. In the case $\frac{k}{\mu} = 1$ (fig 5.2a), we see that the error is lowest in the middle part of the x range. The inflow is larger than the outflow for low values for x, and the other way around for high values of x. Furthermore, the block of lower permeability seems to have little effect in the case $\frac{k}{\mu} = 1$. In the case $\frac{k}{\mu} = 1 + e^{-\frac{x}{1\ m}}$, the block seems to enlarge the error. Lastly in the case $\frac{k}{\mu} = 1 + e^{-\frac{x}{1\ m}}$,we see fluctuations in the error as vertical stripes and the error is a factor 10 larger. Secondly, we will consider case II with different boundary conditions described in equation (4.10). A schematic representation is shown in figure 5.3. In case II presented in figure 5.4), the error is even smaller than for case I.

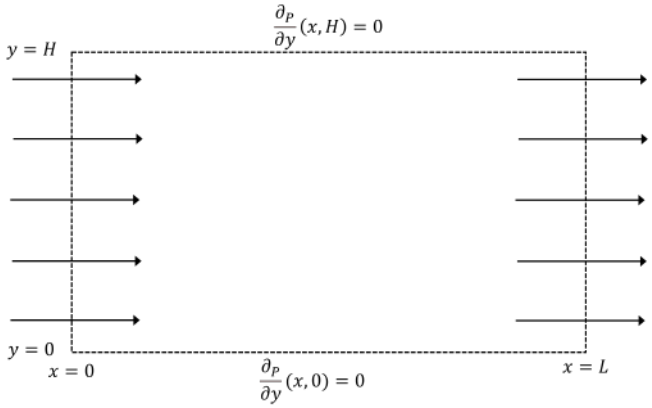
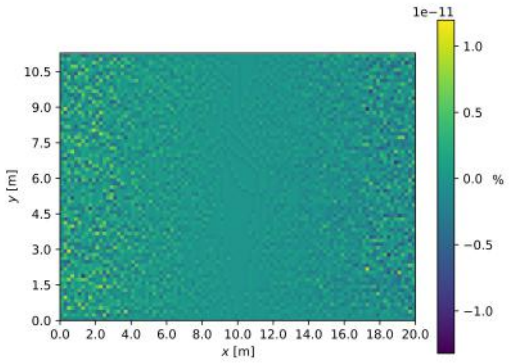
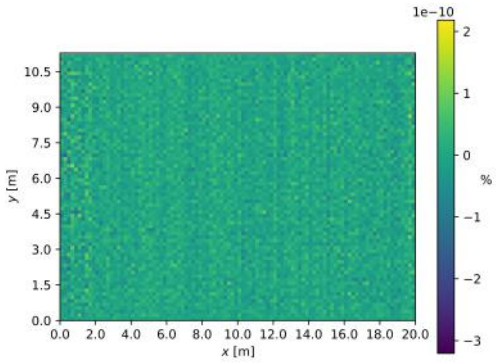


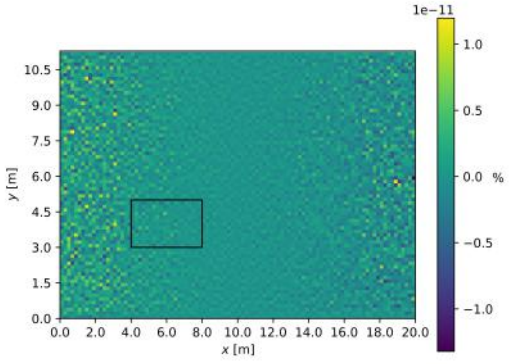
Figure 5.1: Schematic representation of the domain and boundary conditions of case I.



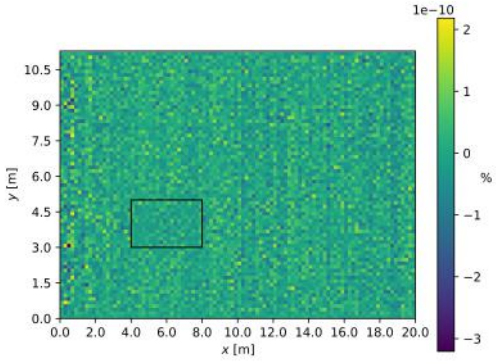
(a) $\frac{k}{\mu} = 1 \frac{m^4}{Ns}$



(b) $\frac{k}{\mu} = 1 + e^{-\frac{x}{1\,m}} \frac{m^4}{Ns}$



(c) $\frac{k}{\mu} = 0.5 \frac{m^4}{Ns}$ for $(x, y) \in [4, 8]x[3, 5]$ and $\frac{k}{\mu} = 1 \frac{m^4}{Ns}$ elsewhere



(d) $\frac{k}{\mu} = 0.5 \frac{m^4}{Ns}$ for $(x, y) \in [4, 8]x[3, 5]$ and $\frac{k}{\mu} = 1 + e^{-\frac{x}{1\,m}} \frac{m^4}{Ns}$ elsewhere

Figure 5.2: Error percentage in mass conservation per cell according to equation 5.1, for different values of $\frac{k}{\mu}$ for case I.

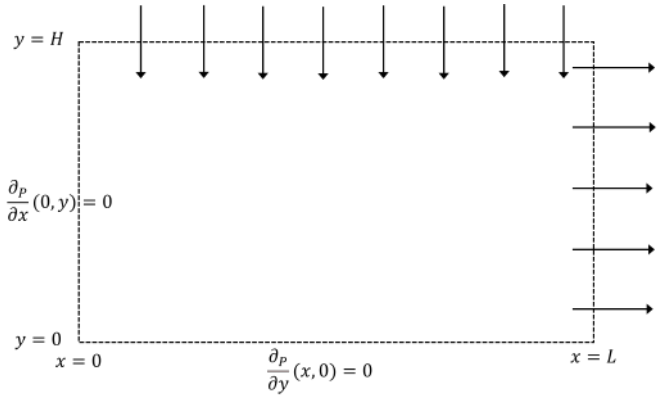


Figure 5.3: Schematic representation of the domain and boundary conditions of case II.

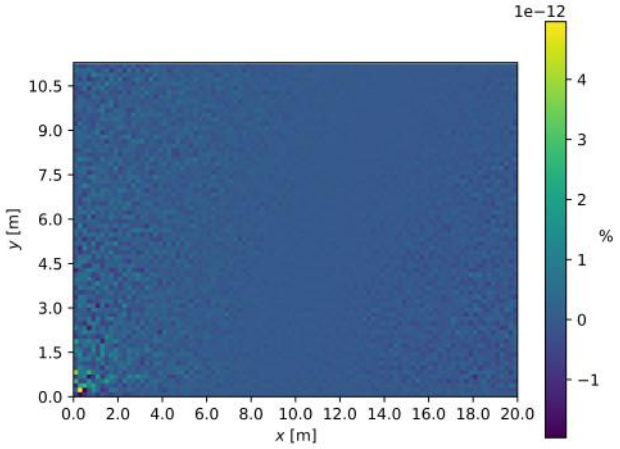


Figure 5.4: Error percentage in mass conservation per cell according to equation 5.1 for $\frac{k}{\mu} = 1\ \frac{m^4}{Ns}$.

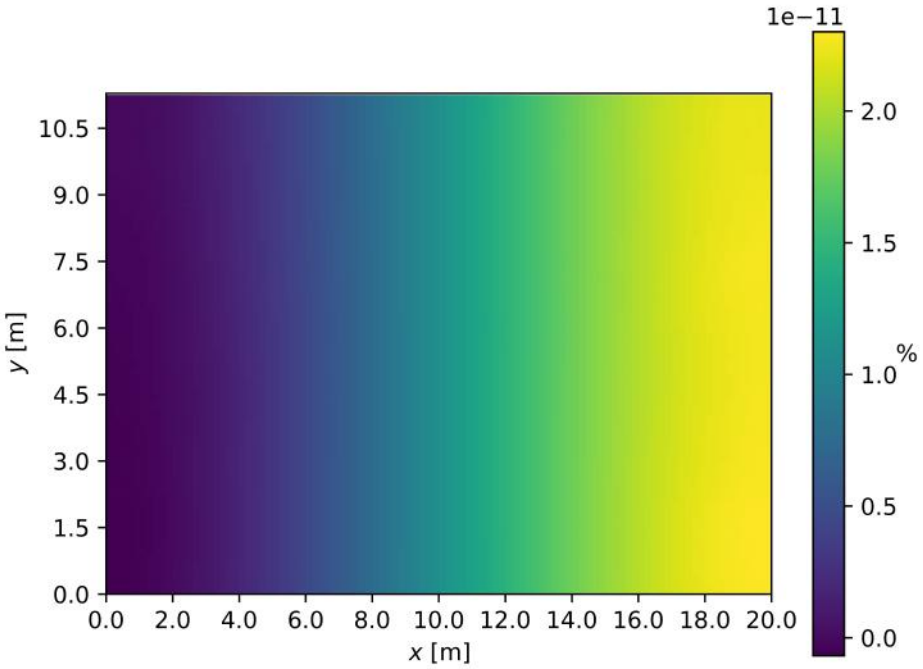
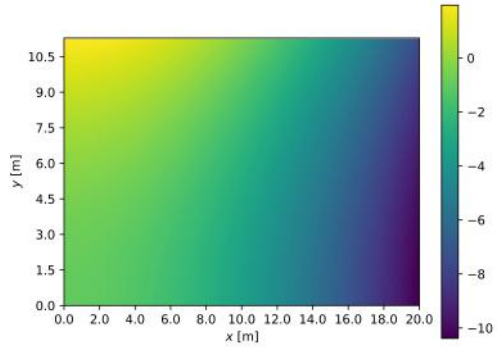


Figure 5.5: Relative difference between the analytical and numerical solution for case I.

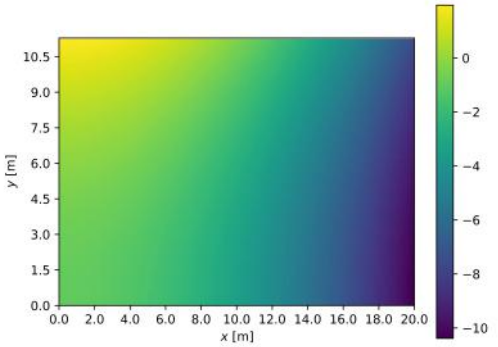
5.2. Comparison with analytical model

In chapter 4 the analytical solutions to the pressure field were found for case I (figure 5.1) and case II (figure 5.3). In this section we will compare these analytical solutions with the numerical solution. For case I the pressure was found to be a linear function of x. The relative difference with the numerical solution is plotted in figure 5.5. In section 4.2 the analytical solution of case II is derived with eigenfunction expansion. There were two eigenfunction expansions, one for x ($\psi_n(x)$) and one for y ($\phi_n(y)$). This section will focus on comparing the analytical solutions with each other and the numerical solution (figure 5.6). The results displayed in figure 5.6 a), b) and c) show that the analytical and numerical solutions are very similar. The first 1000 terms of the infinite sum of the analytical solutions were programmed, adding more terms decreases the relative differences (figure 5.7). In figure 5.6 f) we see that the relative difference has a repetitive pattern. This is probably caused by the eigenfunctions $\phi_n(y) = \cos\left(\frac{n\pi}{L}y\right)$ of the analytical solution. A jump is shown on the boundary at $y = H$ as expected. The analytical solution doesn't hold at this boundary because the eigenfunction expansion satisfies a homogeneous boundary condition, in contrast to the nonhomogeneous boundary condition of the problem. In figure 5.6 e) the relative difference is much smaller than in f). Here we also see a jump at the boundary (here at $x = L$) caused by the eigenfunctions $\phi_n(x) = \cos\left(\frac{n\pi}{H}\right)$ satisfying a homogeneous boundary condition. When comparing the two eigenfunction expansions (figure 5.6) it becomes clear that the eigenfunction expansion for y is more dominant and at both the nonhomogeneous boundary conditions there is a jump.

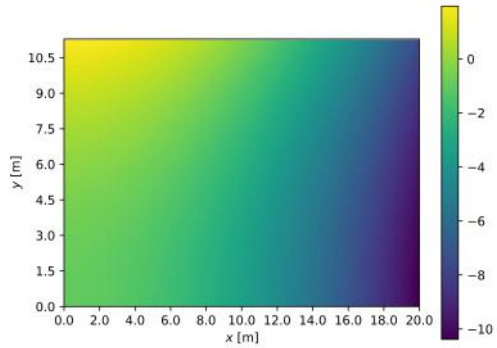
In section 4.3 the analytical solution for the heat equation was derived. The analytical solution and the numerical solution are both shown in figure 5.8. For the two time stamps larger than 0, the solutions are close to one another but still differ for $x < L/2$. At $t = 0$, the analytical solution oscillates strongly. It is clear that the analytical solution is not yet accurate enough. This possible comes from rounding errors when deriving the eigenvalues that may be enlarged by solving the matrix equation for c_n.



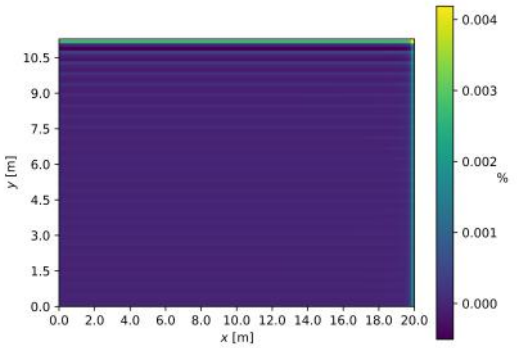
(a) Eigenfunction expansion with $\psi_n(x)$



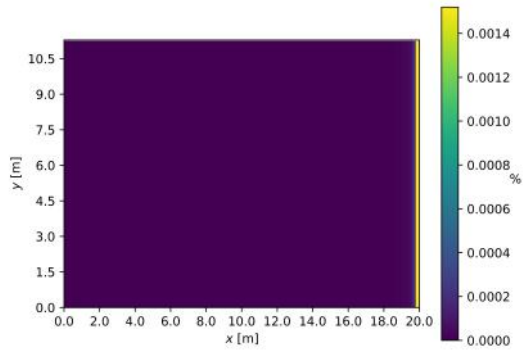
(b) Eigenfunction expansion with $\phi_n(y)$



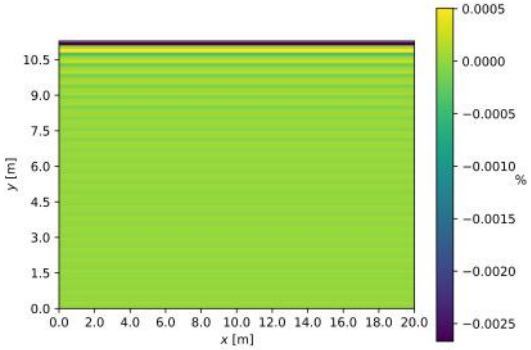
(c) Numerical solution



(d) Relative difference between eigenfunction expansions $\phi_n(y)$ and $\psi_n(x)$; $(p_\phi - p_\psi)/max\ p_\phi \cdot 100\ \%$

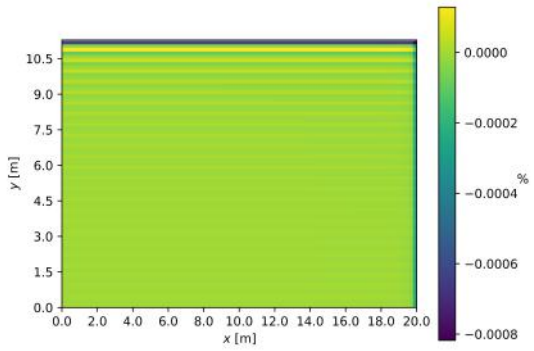


(e) Relative difference between numerical solution and eigenfunction expansion with $\psi_n(x)$; $(p_{num} - p_\psi)/max\ p_{num} \cdot 100\ \%$

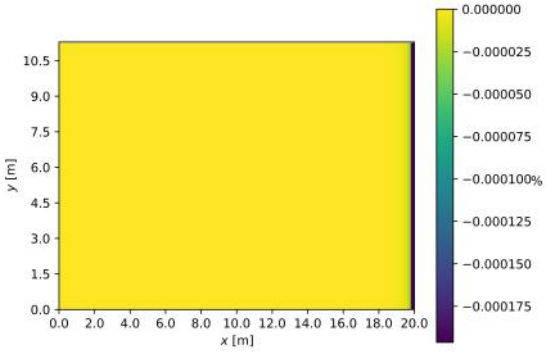


(f) Relative difference between numerical solution and eigenfunction expansion with $\phi_n(y)$; $(p_{num} - p_\phi)/max\ p_{num} \cdot 100\ \%$

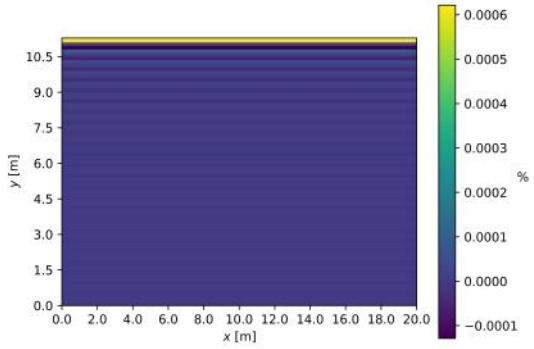
Figure 5.6: Pressure fields for case II (figure 5.3) and the relative differences between the numerical and analytical solutions (with N=1000 terms). The reference points of the pressure fields (a-c) were equal and at position (0,0). The pressure fields were divided by |p(0,0)|.



(a) Relative difference between eigenfunction expansions $\phi_n(y)$ and $\psi_n(x)$; $(p_\phi - p_\psi)/max\ p_\phi \cdot 100\ \%$



(b) Relative difference between numerical solution and eigenfunction expansion with $\psi_n(x)$; $(p_{num} - p_\psi)/max\ p_{num} \cdot 100\ \%$



(c) Relative difference between numerical solution and eigenfunction expansion with $\phi_n(y)$; $(p_{num} - p_\phi)/max\ p_{num} \cdot 100\ \%$

Figure 5.7: Relative differences between the numerical and analytical solutions (with N=2000 terms) for the pressure fields for case II.

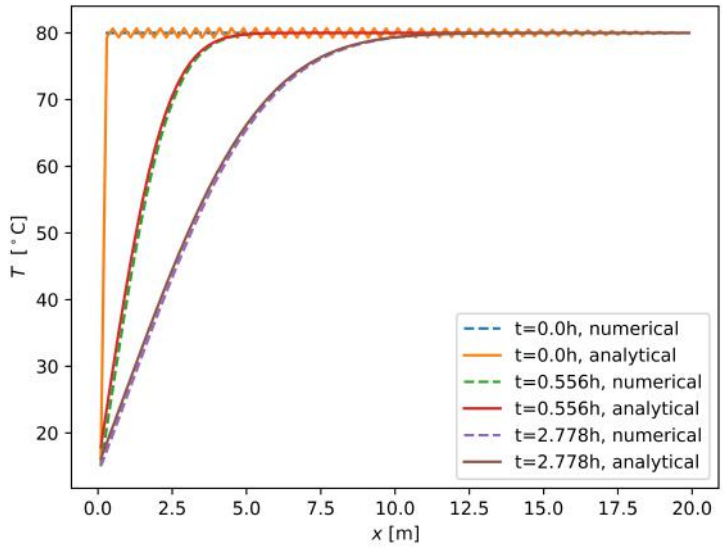
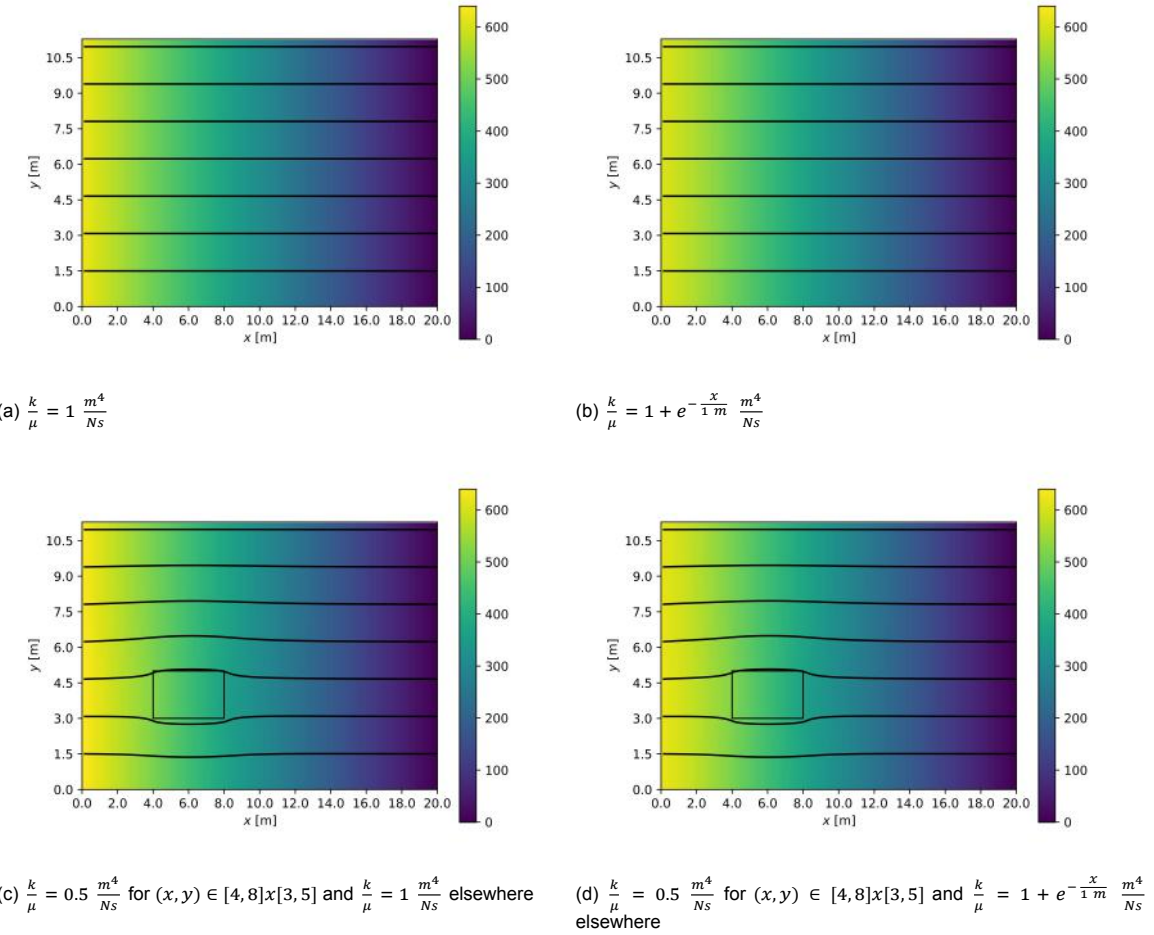


Figure 5.8: Numerical and analytical solution (first 100 terms) for the temperature distribution at different times, where $\alpha \cdot 10^3 = 6.8 \cdot 10^{-4}\ m^2/s$ and $\beta = 5.3 \cdot 10^{-5}\ m/s$.



(a) $\frac{k}{\mu} = 1\ \frac{m^4}{Ns}$

(b) $\frac{k}{\mu} = 1 + e^{-\frac{x}{1\,m}}\ \frac{m^4}{Ns}$

(c) $\frac{k}{\mu} = 0.5\ \frac{m^4}{Ns}$ for $(x, y) \in [4,8]x[3,5]$ and $\frac{k}{\mu} = 1\ \frac{m^4}{Ns}$ elsewhere

(d) $\frac{k}{\mu} = 0.5\ \frac{m^4}{Ns}$ for $(x, y) \in [4,8]x[3,5]$ and $\frac{k}{\mu} = 1 + e^{-\frac{x}{1\,m}}\ \frac{m^4}{Ns}$ elsewhere

Figure 5.9: Pressure field (colour gradient) $p/(\rho_w u_0^2)$ and streamplot for different values of $\frac{k}{\mu}$.

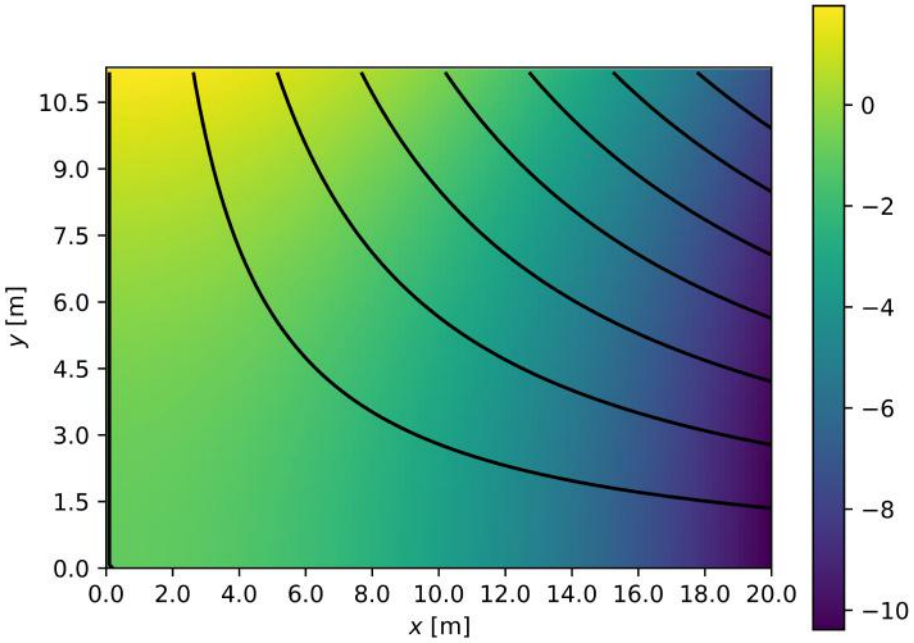
5.3. Numerical model results for case I and II

There are two results to be shown, the pressure field and velocity field. The pressure and velocity field are shown in figure 5.9 for case I for different values of $\frac{k}{\mu}$. The pressure field is visible in the colour gradient and the velocity field in the lines of the streamplot. The lines of the streamplot describe where the value of the streamfunction is constant. With the streamfunction defined by,

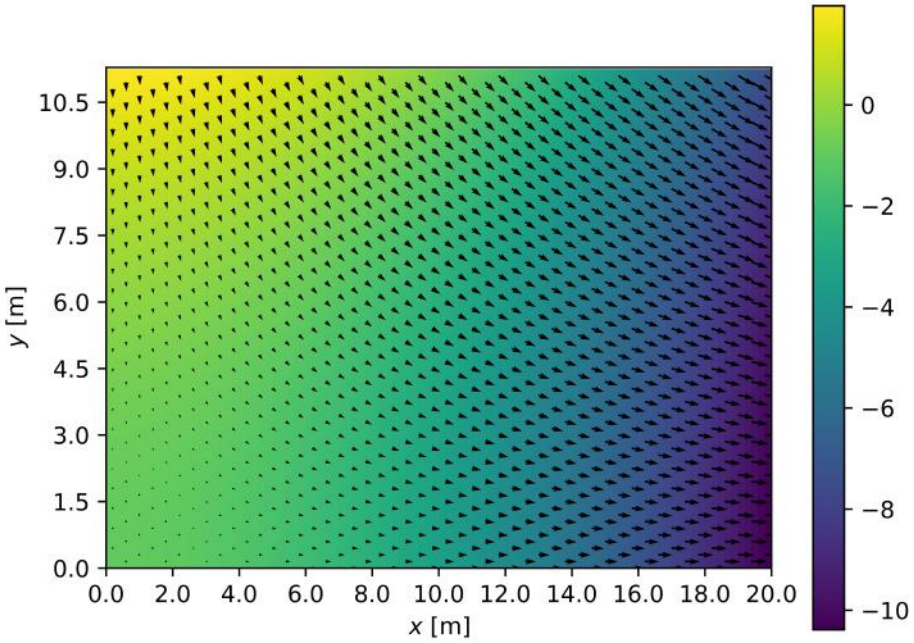
$$\Phi(x, y) = \int_{y'=0}^{y'=y} \vec{u} \cdot \hat{x}\, dy. \tag{5.2}$$

The pressure field for the case $\frac{k}{\mu} = 1\ \frac{m^4}{Ns}$ and $\frac{k}{\mu} = 1 + e^{-\frac{x}{1\,m}}\ \frac{m^4}{Ns}$ seem very similar. However, for the latter the pressure difference is slightly smaller. In figure 5.9a)-b) the water flows in a straight line from $x = 0$ towards $x = L$. This is as expected from the analytical solution found in section 4.1. In figure 5.9 a block in the domain is set to have a lower permeability. The streamplots indeed show that the water partly flows around these blocks of lower permeability.

Figure 5.10 shows the pressure and velocity field for case II. The water flows from boundary $y = H$ towards boundary $x = L$ as we expect from the boundary conditions. Accordingly, the pressure field is the highest at $y = H$ and lowest at $x = L$. In figure 5.10b) we can see that there does not flow any water through the boundaries $x = 0$ and $y = 0$.



(a) Numerical solution of pressure field for case II with streamfunction.



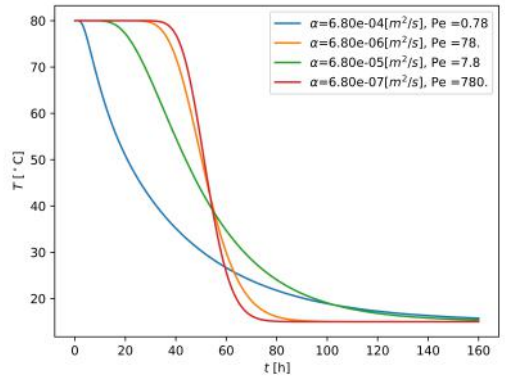
(b) Numerical solution of pressure field for case II with velocity field.

Figure 5.10: Streamplot and velocity field numerically derived from equation 4.10. The pressure fields were divided by |p(0,0)|.

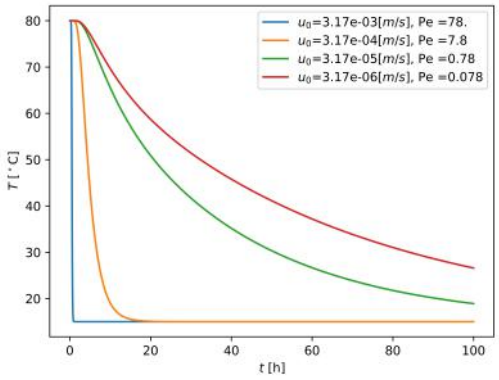
5.4. Numerical model results for case III

5.4.1. Parameter variation

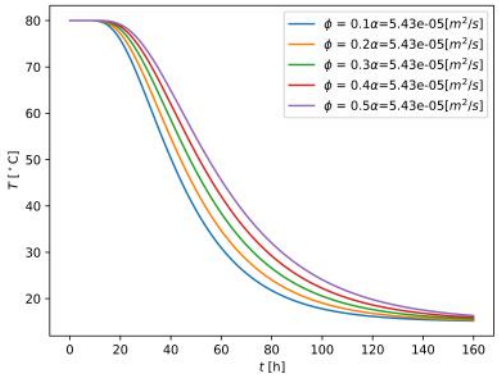
We are interested to see how different parameters influence the temperature distribution in time. Three parameters are analyzed, the effective thermal diffusivity coefficient α, the initial velocity of the water u_0 and the porosity ϕ of the medium (see figure 5.11). The molecular thermal diffusivity coefficient depends on the physical properties of the materials considered and therefore does not change. However, the effective diffusivity coefficient could differ up to a factor 10^3 due to the dispersion of the water. We see different characteristics of the temperature distribution when the Péclet number is smaller than 10 or larger than 10 (figure 5.11 a)). We see the same difference in characteristics in figure 5.11 b). Furthermore, the cooling of the porous medium is delayed when the flow rate u_0 is smaller. Figure 5.11 c) shows a delay in cooling when the porosity ϕ is increased. This can be explained by the fact that the permeability of the medium is not related to the porosity in the used model. Physically however, the permeability does depend on the porosity. The flow rate u_0 therefore doesn't change in this model when the porosity is changed. The volume of the water in the medium is enlarged, and the flow rate is relatively smaller. This results in a delay of cooling the medium.



(a) $u_0 = 3.17 \cdot 10^{-5}\ m/s$ and $\phi = 0.2$



(b) $\alpha = 6.8 \cdot 10^{-4}\ m^2/s$ and $\phi = 0.2$



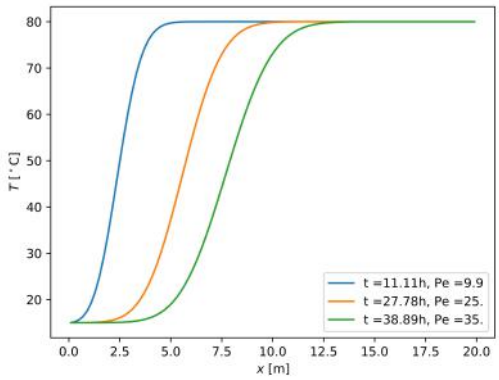
(c) $u_0 = 3.17 \cdot 10^{-5}\ m/s$, Pe=0.78.

Figure 5.11: The temperature T [° C] in time [h] in the point (10 m, 7.5 m) for varying parameters.

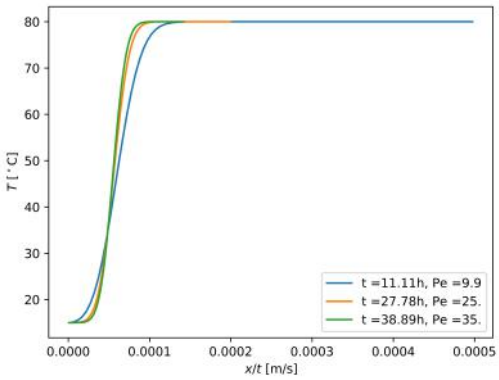
5.4.2. Rescaling the temperature distribution

The purpose of rescaling is to be able to show the temperature distribution for general x and t. The characteristic length of the convection-diffusion problem varies in time. This was taken into account when calculating the Péclet number,

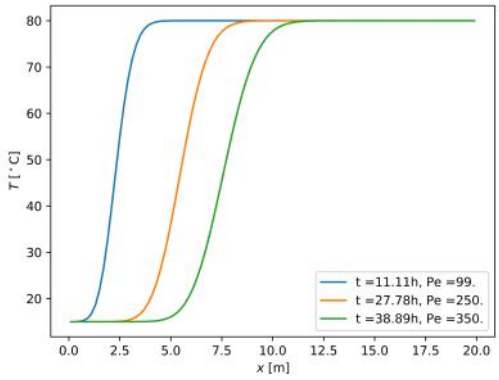
$$Pe = \frac{\gamma u_0^2 t}{\alpha}. \tag{5.3}$$



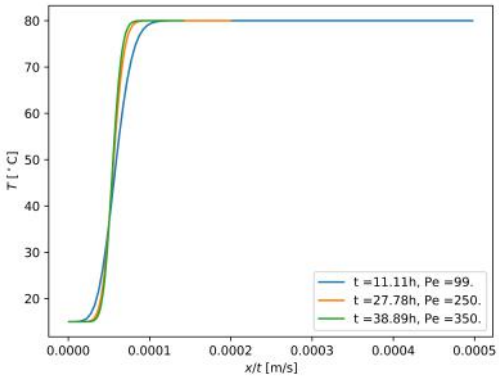
(a) $\alpha \cdot 10 = 6.8 \cdot 10^{-6} m^2/s$



(b) $\alpha \cdot 10 = 6.8 \cdot 10^{-6} m^2/s$



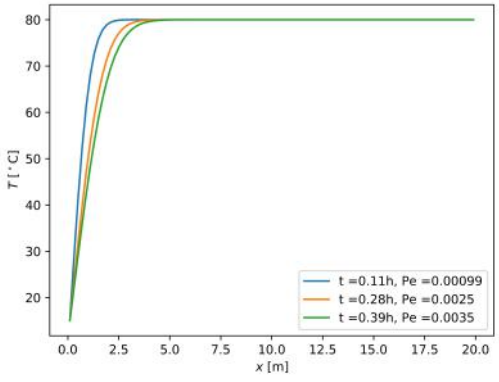
(c) $\alpha = 6.8 \cdot 10^{-7} m^2/s$



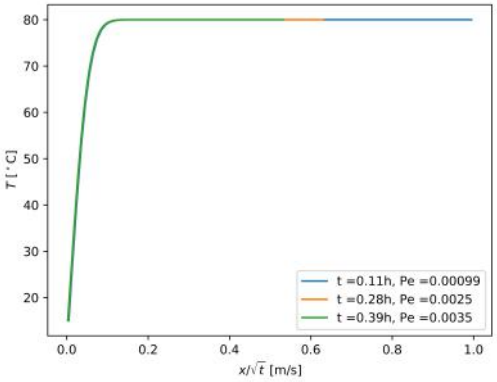
(d) $\alpha = 6.8 \cdot 10^{-7} m^2/s$

Figure 5.12: Rescaling of the temperature distribution for high Péclet numbers.

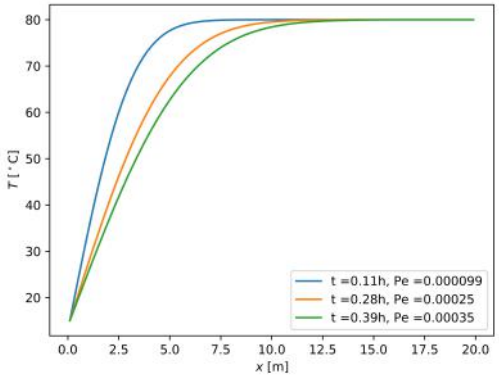
Figure 5.12 shows that the temperature distribution is a function of x/t when the Péclet number is large, indicating the convection dominance of the distribution. The function depending on x/t decribing the high temperature for high values of Péclet numbers is the error function. For low values of the Péclet number the temperature distribution is a function depending on $x/\sqrt{t}$ (figure 5.13).



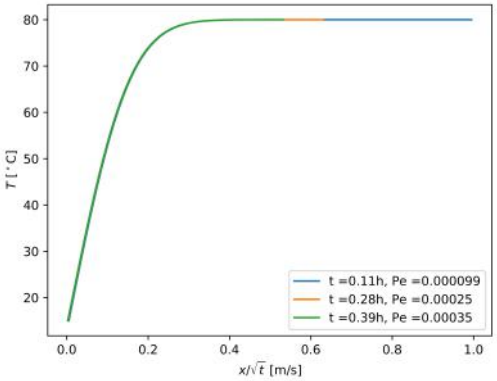
(a) $\alpha \cdot 10^4 = 6.8 \cdot 10^{-3} m^2/s$



(b) $\alpha \cdot 10^4 = 6.8 \cdot 10^{-3} m^2/s$



(c) $\alpha \cdot 10^3 = 6.8 \cdot 10^{-4} m^2/s$



(d) $\alpha \cdot 10^3 = 6.8 \cdot 10^{-4} m^2/s$

Figure 5.13: Rescaling of the temperature distribution for low Péclet numbers.

6

Conclusion

In this research the influence of porosity and flow rate in an aquifer on its temperature distribution was investigated. For this purpose a numerical model was derived with use of finite difference methods and the forward Euler method to describe the pressure field, velocity field and temperature distribution. The accuracy of these numerical models were analysed by comparing their results with the results of analytical solutions.

For the pressure and velocity field the results are very accurate. The analytical and numerical solutions for the pressure and velocity field both described a quasi-linear velocity field along the x-axis, with a negligible y-component. Accordingly, the pressure field described a decreasing linear dependence on the x-coordinate. Furthermore, it was shown that the velocity field of the numerical model obeyed the conservation law of mass up to machine precision for different values of permeability: a constant value and a smooth function with and without a block with lower permeability of a factor two.

There was some difference in the temperature distribution for the analytical and numerical solution. It is expected that this difference comes from an error in the analytical solution. The analytical problem could not be solved purely analytical, some numerical help was needed for deriving the eigenvalues and coefficients of the eigenfunction expansion. Probably, some errors from the numerical computation influenced the results of the analytical solution. Therefore the accuracy of the numerical model for the temperature distribution could not be determined with certainty. The temperature distribution of the numerical solution is described by a complementary error function. For low Péclet numbers this function depends on $x/\sqrt{t}$ and for high numbers on x/t.

Finally, we conclude the following on the influence of parameters on the temperature distribution. The characteristics of the temperature distribution changes when the flow rate of the aquifer is changed. For higher values of flow rate, the Péclet number increases and the characteristic of the temperature distribution changes accordingly. Furthermore, a decreasing flow rate translates into a delay of cooling the aquifer as we would expect. In the current model, an increase in porosity results in a delay of cooling the aquifer. This is because in the current model the permeability is not related to the porosity. Therefore the flow rate is unchanged and relatively smaller compared to the volume of the water in the medium.

For further research it is advised to analyse errors occuring in the numerical computation for the analytical solution in order to determine the accuracy of the numerical model for the temperature distribtuion. In a new model the permeability can be related to the porosity. Chemical reactions that would cause a decrease of porosity can be implemented. When the chemical reactions are implemented in the model, one can look into preventing chemical clogging.

References

[1] John W Lund and Aniko N Toth. Direct utilization of geothermal energy 2020 worldwide review. *Geothermics*, 90:101915, 2021.

[2] B.D. Wood, S.V. Apte, J.A. Liburdy, R.M. Ziazi, X. He, J.R. Finn, and V.A. Patil. A comparison of measured and modeled velocity fields for a laminar flow in a porous medium. *Advances in Water Resources*, 85:45–63, 2015.

[3] Miad Jarrahi, Kayla R Moore, and Hartmut M Holländer. Comparison of solute/heat transport in fractured formations using discrete fracture and equivalent porous media modeling at the reservoir scale. *Physics and Chemistry of the Earth, Parts A/B/C*, 113:14–21, 2019.

[4] B Kanimozhi, P Rajkumar, RS Kumar, S Mahalingam, Vivek Thamizhmani, Arun Selvakumar, S Ravikumar, R Kesavakumar, and Venkat Pranesh. Kaolinite fines colloidal-suspension transport in high temperature porous subsurface aqueous environment: Implications to the geothermal sandstone and hot sedimentary aquifer reservoirs permeability. *Geothermics*, 89:101975, 2021.

[5] Raphael Schulz, Nadja Ray, Simon Zech, Andreas Rupp, and Peter Knabner. Beyond Kozeny–Carman: Predicting the Permeability in Porous Media. *Transport in Porous Media*, 130:487–512, 2019.

[6] Alexander Litvinenko, Dmitry Logashenko, Raul Tempone, Gabriel Wittum, and David Keyes. Solution of the 3d density-driven groundwater flow problem with uncertain porosity and permeability. *GEM-International Journal on Geomathematics*, 11:1–29, 2020.

[7] Jie Yang, Wei-xin Ren, Bo Kang, and Yuezan Tao. Experimental investigation on chemical clogging mechanism of loose porous media in recharge process of groundwater heat pump. *Environmental Technology*, 44(16):2357–2373, 2023.

[8] Hongmei Yin, Chaofan Song, Ling Ma, Liuhua Gao, Xuan Yang, Wenjia Li, and Jun Zhao. Analysis of flow and thermal breakthrough in leaky downhole coaxial open loop geothermal system. *Applied Thermal Engineering*, 194:117098, 2021.

[9] Tingting Ke, Shaopeng Huang, Wei Xu, and Xuxiang Li. Study on heat extraction performance of multiple-doublet system in hot sedimentary aquifers: Case study from the xianyang geothermal field, northwest china. *Geothermics*, 94:102131, 2021.

[10] Craig T. Simmons. Henry Darcy (1803–1858): Immortalised by his Scientific legacy. *Hydrogeology Journal*, 16(6):1023–1038, 2008. doi: 10.1007/s10040-008-0304-3.

[11] Magali Billen. 2.5: Darcy's Law - Flow in a Porous Medium — geo.libretexts.org. https://geo.libretexts.org/Courses/University_of_California_Davis/GEL_056%3A_Introduction_to_Geophysics/Geophysics_is_everywhere_in_geology../02%3A_Diffusion_and_Darcy's_Law/2.05%3A_Darcy's_Law_-_Flow_in_a_Porous_Medium. [Accessed 11-08-2023].

[12] Rob Mudde and Harrie van den Akker. *Fysische transportverschijnselen*. Delft: Delft Academic Press, 2014.

[13] L.P.B.M. Janssen and M.M.C.G. Warmoeskerken. *Transport Phenomena Data Companion*. Delft: VSSD, 2006. ISBN -13 9789071301599.

[14] J.Veldkamp, C. Geel, and E. Peters. Characterization of aquifer properties of the Brussels Sand Member from cuttings. Technical Report TEUE819001, TNO, 2022.

[15] numpy.linalg.solve; NumPy v1.26 Manual — numpy.org. https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html. [Accessed 19-09-2023].

[16] Pim Jager. Eolisch sediment transport. Bachelor thesis, Delft University of Technology, 2022.

[17] Sedat Biringen. A note on the numerical stability of the convection-diffusion equation. *Journal of Computational and Applied Mathematics*, 7(1):17–20, 1981.

[18] Ifan G. Hughes and Thomas P.A. Hase. *Measurements and their Uncertainties*. New York: Oxford University Press, 2010. ISBN 978-0-19-956633-4.

[19] Richard Haberman. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*, chapter 8.4 method of eigenfunction expansion using Green's formula. Harlow: Pearson Education, 2014. ISBN 13: 978-1-292-03985-5.

[20] Martin Braun. *Differential Equations and Their Application*. New York: Springer, 1992.

[21] Małgorzata B. Glinowiecka-Cox. Analytical Solution of 1D Diffusion-Convection Equation with Varying Boundary Conditions. Bachelor thesis, Portland State University, 2022.

[22] C. Vuik, F.J. Vermolen, M.B. van Gijzen, and M.J. Vuik. *Numerical Methods for Ordinary Differential Equations*. Delft: Delft Academic Press, 2018.

A

Numerical scheme

A.1. Pressure field

The numerical approximation of pressure p at location $\left[\left(i+\frac{1}{2}\right)\Delta x, \left(j+\frac{1}{2}\right)\Delta y\right]$ is $p_{i,j}$, similarly for $\kappa := \frac{k}{\mu}$. The equations from (2.7) are approximated by the central difference method.

$$\nabla p \approx \frac{p_{i+\frac{1}{2},j} - p_{i-\frac{1}{2},j}}{\Delta x}\hat{x} + \frac{p_{i,j+\frac{1}{2}} - p_{i,j-\frac{1}{2}}}{\Delta y}\hat{y} \tag{A.1}$$

$$\begin{aligned}\nabla \cdot (-\kappa \nabla p) &\approx \frac{-\kappa_{i+\frac{1}{2},j}\frac{p_{i+1,j}-p_{i,j}}{\Delta x} + \kappa_{i-\frac{1}{2},j}\frac{p_{i,j}-p_{i-1,j}}{\Delta x}}{\Delta x} + \frac{-\kappa_{i,j+\frac{1}{2}}\frac{p_{i,j+1}-p_{i,j}}{\Delta y} + \kappa_{i,j-\frac{1}{2}}\frac{p_{i,j}-p_{i,j-1}}{\Delta y}}{\Delta y} \\ &= \frac{-\kappa_{i+\frac{1}{2},j}p_{i+1,j} + \left(\kappa_{i+\frac{1}{2},j} + \kappa_{i-\frac{1}{2},j}\right)p_{i,j} - \kappa_{i-\frac{1}{2},j}p_{i-1,j}}{\Delta x^2} \\ &\quad + \frac{-\kappa_{i,j+\frac{1}{2}}p_{i,j+1} + \left(\kappa_{i,j+\frac{1}{2}} + \kappa_{i,j-\frac{1}{2}}\right)p_{i,j} - \kappa_{i,j-\frac{1}{2}}p_{i,j-1}}{\Delta y^2}\end{aligned} \tag{A.2}$$

κ at the boundary between two grids is defined as the harmonic mean of the κ values at those grid points, ie.

$$\kappa_{i+\frac{1}{2},j} := \frac{2}{\frac{1}{\kappa_{i,j}} + \frac{1}{\kappa_{i+1,j}}} \tag{A.3}$$

The boundary conditions from (2.7) are approximated via central differences,

$$\frac{\partial p}{\partial x}(0,y) \approx \frac{p_{0,j} - p_{-1,j}}{\Delta x} = -\frac{u_0}{\kappa_{-\frac{1}{2},j}} \tag{A.4}$$

$$\frac{\partial p}{\partial x}(L,y) \approx \frac{p_{n_x,j} - p_{n_x-1,j}}{\Delta x} = -\frac{u_0}{\kappa_{n_x-\frac{1}{2},j}} \tag{A.5}$$

$$\frac{\partial p}{\partial y}(x,0) \approx \frac{p_{i,0} - p_{i,-1}}{\Delta y} = 0 \tag{A.6}$$

$$\frac{\partial p}{\partial y}(x,H) \approx \frac{p_{i,n_y} - p_{i,n_y-1}}{\Delta y} = 0 \tag{A.7}$$

Setting equation (A.2) equal to zero, we find an expression for the numerical approximations of p for the interior of the domain (A.8). For $1 \leq i \leq n_x - 2$ and $1 \leq j \leq n_y - 2$,

$$\frac{-\kappa_{i+\frac{1}{2},j}p_{i+1,j} + \left(\kappa_{i+\frac{1}{2},j} + \kappa_{i-\frac{1}{2},j}\right)p_{i,j} - \kappa_{i-\frac{1}{2},j}p_{i-1,j}}{\Delta x^2} + \frac{-\kappa_{i,j+\frac{1}{2}}p_{i,j+1} + \left(\kappa_{i,j+\frac{1}{2}} + \kappa_{i,j-\frac{1}{2}}\right)p_{i,j} - \kappa_{i,j-\frac{1}{2}}p_{i,j-1}}{\Delta y^2} = 0. \quad \text{(A.8)}$$

In order to find expressions at the boundary of the domain, we substitute equations (A.4- A.7) into equation (A.8). Some examples are given of such substitutions (A.9-A.12). For $i = 0$ and $j = 0$,

$$\frac{-\kappa_{i+\frac{1}{2},j}p_{i+1,j} + \kappa_{i+\frac{1}{2},j}p_{i,j}}{\Delta x^2} + \frac{-\kappa_{i,j+\frac{1}{2}}p_{i,j+1} + \kappa_{i,j+\frac{1}{2}}p_{i,j}}{\Delta y^2} = \frac{u_0}{\Delta x}. \quad \text{(A.9)}$$

For $i = 0$ and $1 \leq j \leq n_y - 2$,

$$\frac{-\kappa_{i+\frac{1}{2},j}p_{i+1,j} + \kappa_{i+\frac{1}{2},j}p_{i,j}}{\Delta x^2} + \frac{-\kappa_{i,j+\frac{1}{2}}p_{i,j+1} + \left(\kappa_{i,j+\frac{1}{2}} + \kappa_{i,j-\frac{1}{2}}\right)p_{i,j} - \kappa_{i,j-\frac{1}{2}}p_{i,j-1}}{\Delta y^2} = \frac{u_0}{\Delta x}. \quad \text{(A.10)}$$

For $1 \leq i \leq n_x - 2$ and $j = n_y - 1$,

$$\frac{-\kappa_{i+\frac{1}{2},j}p_{i+1,j} + \left(\kappa_{i+\frac{1}{2},j} + \kappa_{i-\frac{1}{2},j}\right)p_{i,j} - \kappa_{i-\frac{1}{2},j}p_{i-1,j}}{\Delta x^2} + \frac{\kappa_{i,j-\frac{1}{2}}p_{i,j} - \kappa_{i,j-\frac{1}{2}}p_{i,j-1}}{\Delta y^2} = 0. \quad \text{(A.11)}$$

For $i = n_x - 1$ and $j = n_y - 1$,

$$\frac{\kappa_{i-\frac{1}{2},j}p_{i,j} - \kappa_{i-\frac{1}{2},j}p_{i-1,j}}{\Delta x^2} + \frac{\kappa_{i,j-\frac{1}{2}}p_{i,j} - \kappa_{i,j-\frac{1}{2}}p_{i,j-1}}{\Delta y^2} = -\frac{u_0}{\Delta x}. \quad \text{(A.12)}$$

The numerical scheme is solved by translating the derived equations into matrix form and use the function numpy.linalg.solve from python.

A.2. Temperature distribution

This section will explain how the heat equation was discretized. Similarly as for p, $T_{i,j}^k$ is the numerical approximation of T at location $\left[\left(i + \frac{1}{2}\right)\Delta x, \left(j + \frac{1}{2}\right)\Delta y\right]$ on time $k\Delta t$. The diffusion term of equation (2.18) is approximated by central difference method (A.13) and the convection term by the upwind difference (A.14) [22]. Finally, the time derivative term is approximated by the forward difference (A.15). Using equations (A.13-A.15), the discretized heat equation can be derived (A.16).

$$\nabla^2 T \approx \left(\frac{T_{i+1,j}^k - 2T_{i,j}^k + T_{i-1,j}^k}{\Delta x^2} + \frac{T_{i,j+1}^k - 2T_{i,j}^k + T_{i,j-1}^k}{\Delta y^2}\right) \quad \text{(A.13)}$$

$$\left(\frac{\partial u_x T}{\partial x} + \frac{\partial u_y T}{\partial y}\right) \approx \left(-\frac{u_{x,i+1,j}T_{i,j}^k - u_{x,i,j}T_{i-1,j}^k}{\Delta x} - \frac{u_{y,i,j+1}T_{i,j}^k - u_{y,i,j}T_{i,j-1}^k}{\Delta y}\right) \quad \text{(A.14)}$$

$$\frac{\partial T}{\partial t} \approx \frac{T_{i,j}^{k+1} - T_{i,j}^k}{\Delta t} \quad \text{(A.15)}$$

$$T_{i,j}^{k+1} = T_{i,j}^k + \Delta t\alpha\left(\frac{T_{i+1,j}^k - 2T_{i,j}^k + T_{i-1,j}^k}{\Delta x^2} + \frac{T_{i,j+1}^k - 2T_{i,j}^k + T_{i,j-1}^k}{\Delta y^2}\right) + \Delta t v\left(-\frac{u_{x,i+1,j}T_{i,j}^k - u_{x,i,j}T_{i-1,j}^k}{\Delta x} - \frac{u_{y,i,j+1}T_{i,j}^k - u_{y,i,j}T_{i,j-1}^k}{\Delta y}\right) \quad \text{(A.16)}$$

The boundary conditions and initial condition from (2.19) are discretized in a similar manner as for the pressure field with central differences.

$$T(0,y,t) \approx T_{0,j}^{k} = T_{low} \tag{A.17}$$

$$\frac{\partial T}{\partial x}(L,y,t) \approx \frac{T_{n_x,j}^{k} - T_{n_x-1,j}^{k}}{\Delta x} = 0 \tag{A.18}$$

$$\frac{\partial T}{\partial y}(x,0,t) \approx \frac{T_{i,0}^{k} - T_{i,-1}^{k}}{\Delta y} = 0 \tag{A.19}$$

$$\frac{\partial T}{\partial y}(x,H,t) \approx \frac{T_{i,n_y}^{k} - T_{i,n_y-1}^{k}}{\Delta y} = 0 \tag{A.20}$$

$$T(x,y,0) \approx T_{i,j}^{0} = \begin{cases} T_{low}, & i = 0 \\ T_{high}, & i \neq 0 \end{cases} \tag{A.21}$$

Equation (A.16) holds for the interior of the domain, i.e. $0 < i < n_x - 1$ and $0 < j < n_y - 1$. At the edge of the domain, boundary conditions (A.18-A.20) should be substituted into equation (A.16). Some examples of these substitutions are given, the other cases will be similar to the given examples.
For $i = n_x - 1$ and $0 < j < n_y - 1$,

$$T_{n_x-1,j}^{k+1} = T_{n_x-1,j} + \Delta t\alpha\left(\frac{-T_{n_x-1,j}^{k} + T_{n_x-2,j}^{k}}{\Delta x^2} + \frac{T_{n_x-1,j+1}^{k} - 2T_{n_x-1,j}^{k} + T_{n_x-1,j-1}^{k}}{\Delta y^2}\right) + \Delta t v\left(-\frac{u_{x,i+1,j}T_{i,j}^{k} - u_{x,i,j}T_{i-1,j}^{k}}{\Delta x} - \frac{u_{y,i,j+1}T_{i,j}^{k} - u_{y,i,j}T_{i,j-1}^{k}}{\Delta y}\right). \tag{A.22}$$

For $0 < i < n_x - 1$ and $j = 0$,

$$T_{i,0}^{k+1} = T_{i,0}^{k+1} + \Delta t\alpha\left(\frac{T_{i+1,0}^{k} - 2T_{i,0}^{k} + T_{i-1,0}^{k}}{\Delta x^2} + \frac{T_{i,1}^{k} - T_{i,0}^{k}}{\Delta y^2}\right) + \Delta t v\left(-\frac{u_{x,i+1,0}T_{i,0}^{k} - u_{x,i,0}T_{i-1,0}^{k}}{\Delta x} - \frac{(u_{y,i,1} - u_{y,i,0})T_{i,0}^{k}}{\Delta y}\right). \tag{A.23}$$

For $i = n_x - 1$ and $j = 0$,

$$T_{n_x-1,0}^{k+1} = T_{n_x-1,0}^{k} + \Delta t\alpha\left(\frac{-T_{n_x-1,0}^{k} + T_{n_x-2,0}^{k}}{\Delta x^2} + \frac{T_{n_x-1,1}^{k} - T_{n_x-1,0}^{k}}{\Delta y^2}\right) + \Delta t v\left(-\frac{u_{x,n_x,0}T_{n_x-1,0}^{k} - u_{x,n_x-1,0}T_{n_x-2,0}^{k}}{\Delta x} - \frac{(u_{y,n_x-1,1} - u_{y,n_x-1,0})T_{n_x-1,0}^{k}}{\Delta y}\right). \tag{A.24}$$

B

Code

B.1. Numerical model

```
   import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib.animation import PillowWriter
import copy

#from cartesian to x-lexicographic ordering
def lex(i, j, N):
   return i+j*N
#analytical solution for p
def p_ana(x, u0, p0):
   #return -u0*(np.log(1+np.exp(x))-np.log(2))+p0 #kappa=1+e^(-x)
   return -u0*x+p0 #kappa = 1

#discretization
L=20     #m
nx=100
dx=L/nx
H=11.25 #m
ny=75
dy=H/ny
print(dx,dy)

#and other variables
u0 = 0.0000317 #m/s from 1km/1y
lam_w=0.6      #W/mK
rho_w = 10**3  #kg/m^3
c_w = 4.18*10**3 #J/kgK
phi = 0.2
lam_s = 1.1    #W/mK
rho_s = 2.6*10**3 #kg/m^3
c_s = 0.8*10**3 #J/kgK

#coefficients for heat equation
alpha = (lam_w+lam_s)/(c_w*rho_w*phi+c_s*rho_s*(1-phi))
gam = c_w*rho_w/(c_w*rho_w*phi+c_s*rho_s*(1-phi))
print(alpha)
alpha=alpha*1000
print(alpha, gam)

T_high=80
```

```
T_low = 15
dt=10

#define grid
#grid points for scalar quantities
x = [dx/2+i*dx for i in range(nx)] #x_0 to x_(n-1)
y = [dy/2+j*dy for j in range(ny)] #y_0 to y_(n-1)

#crossing points of the grid
bx = [i*dx for i in range(nx+1)] #bx_0 to bx_n
by = [j*dy for j in range(ny+1)] #by_0 to by_n


#define kappa := permeability/viscosity
#define different cases
case = 1       #1:uniform grid, 2:1+e^-x, 3:block with 0.5 and 1 else, 4:block
    with 0.5 and 1+e^-x else

def kappa(x,y,case):
   if case == 1:
      return 10^3
   elif case == 2:
      return 1+np.exp(-x)
   elif case == 3:
      if 4<x<8 and 3<y<5:
         return 0.5
      #elif 10<x<13 and 3<y<5:
       #  return 0.00000005
      else:
         return 1
   elif case == 4:
      if 4<x<8 and 3<y<5:
         return 0.5
      else:
         return 1+np.exp(-x)
   else:
      print('Invalid value for case')
#SOLVE THE PRESSURE FIELD
#define A and f for Ap=f
A=np.zeros((nx*ny, nx*ny))
f=np.zeros((nx*ny,1))
k=0
for i in range(nx):
   for j in range(ny):
      kap = kappa(x[i],y[j],case)
      if j==0:    #boundary condition at y=0
         kap_N = 1/(0.5/kap+0.5/kappa(x[i],y[j+1],case))
         A[k, lex(i,j+1, nx)]-=kap_N/dy**2
         A[k, lex(i,j,nx)]+=kap_N/dy**2
      elif j==ny-1: #boundary condition at y=H
         kap_S = 1/(0.5/kap+0.5/kappa(x[i],y[j-1],case))
         A[k, lex(i,j-1,nx)]-=kap_S/dy**2
         A[k, lex(i,j,nx)]+=kap_S/dy**2
      else:
         kap_N = 1/(0.5/kap+0.5/kappa(x[i],y[j+1],case))
         kap_S = 1/(0.5/kap+0.5/kappa(x[i],y[j-1],case))
         A[k, lex(i,j+1, nx)]-=kap_N/dy**2
         A[k, lex(i,j-1,nx)]-=kap_S/dy**2
         A[k, lex(i,j,nx)]+=(kap_N+kap_S)/dy**2
      if i==0:    #boundary condition at x=0
         kap_E = 1/(0.5/kap+0.5/kappa(x[i+1],y[j],case))
```

```
            A[k, lex(i+1,j,nx)]-=kap_E/dx**2
            A[k, lex(i,j,nx)]+=kap_E/dx**2
            f[k, 0] = u0/dx
            k+=1
        elif i==nx-1: #boundary condition at x=L
            kap_W = 1/(0.5/kap+0.5/kappa(x[i-1],y[j],case))
            A[k, lex(i-1,j,nx)]-=kap_W/dx**2
            A[k, lex(i,j,nx)]+=kap_W/dx**2
            f[k,0] = -u0/dx
            k+=1
        else:
            kap_E = 1/(0.5/kap+0.5/kappa(x[i+1],y[j],case))
            kap_W = 1/(0.5/kap+0.5/kappa(x[i-1],y[j],case))
            A[k, lex(i+1,j,nx)]-=kap_E/dx**2
            A[k, lex(i-1,j,nx)]-=kap_W/dx**2
            A[k, lex(i,j,nx)]+=(kap_W+kap_E)/dx**2
            k+=1
#solve for p
p = np.linalg.solve(A,f)
p = np.reshape(p, (ny,nx))
p-=p[ny-1,nx-1] #take reference point


#SOLVE VELOCITY FIELD
u_x = np.zeros((ny, nx+1))
u_y = np.zeros((ny+1, nx))
u_x[:,0] = u0
u_x[:,nx] = u0
u_y[0,:] = 0
u_y[ny,:] = 0
for i in range(1,nx):
    u_x[0,i]=-1/(0.5/kappa(x[i-1],y[0],case)+0.5/kappa(x[i],y[0],case))
        *(p[0,i]-p[0,i-1])/dx
    for j in range(1,ny):
        u_y[j,0]=-1/(0.5/kappa(x[0],y[j-1],case)+0.5/kappa(x[0],y[j],case))
            *(p[j,0]-p[j-1,0])/dy
        u_x[j,i]=-1/(0.5/kappa(x[i-1],y[j],case)+0.5/kappa(x[i],y[j],case))
            *(p[j,i]-p[j,i-1])/dx
        u_y[j,i]=-1/(0.5/kappa(x[i],y[j-1],case)+0.5/kappa(x[i],y[j],case))
            *(p[j,i]-p[j-1,i])/dy
print(np.max(u_x), np.max(u_y))

#check mass conservation per cell
cons = np.zeros((ny,nx))
for i in range(nx):
    for j in range(ny):
        cons[j,i]=((u_x[j,i]-u_x[j,i+1])*dy+(u_y[j,i]-u_y[j+1,i])*dx)/
            ((np.abs(u_x[j,i])+np.abs(u_x[j,i+1]))*dy+(np.abs(u_y[j,i])
                +np.abs(u_y[j+1,i]))*dx)*100

#figure for mass conservation
plt.matshow(cons)
plt.xlabel('x')
plt.ylabel('y')
xlab=[i*10 for i in range(nx//10+1)]
ylab=[i*10 for i in range(ny//10+1)]
plt.xticks(xlab, [round(i/nx*L,2) for i in xlab])
plt.yticks(ylab, [round(i/ny*H,2) for i in ylab])
plt.ylim(ny,0)
plt.colorbar()
plt.savefig('mass cons per cell kap=1+e-x, blok.jpg', dpi=1000)
```

```
plt.show()


#find average velocities at grid points
mu_x = np.zeros((ny,nx))
mu_y = np.zeros((ny,nx))
for i in range(nx):
   for j in range(ny):
      mu_x[j,i]=(u_x[j,i]+u_x[j,i+1])/2
      mu_y[j,i]=(u_y[j,i]+u_y[j+1,i])/2
mx = nx/L*np.array(x*ny)
my = np.repeat(np.array(y)/H*ny,nx)

#plot pressure field and velocity field with arrows
step=6
plt.matshow(p)
plt.colorbar()
plt.xlabel('x')
plt.ylabel('y')
xlab=[i*10 for i in range(nx//10+1)]
ylab=[i*10 for i in range(ny//10+1)]
plt.xticks(xlab, [round(i/nx*L,2) for i in xlab])
plt.yticks(ylab, [round(i/ny*H,2) for i in ylab])
plt.ylim(ny,0)
plt.quiver(mx[::step], my[::step], list(np.reshape(mu_x,(1,nx*ny))[0])[::step],
    list(-np.reshape(mu_y,(1,nx*ny))[0])[::step], width=0.002)
plt.show()

#SOLVE TEMPERATURE DISTRIBUTION
T_init = np.ones((ny,nx))*T_high
T_init[:,0] = T_low
T = [T_init]
for t in range(int(10000/dt)+1):
   T_old = copy.deepcopy(T[-1])
   if t%100==0:
      print(t)
   T_new = copy.deepcopy(T_init)
   for j in range(ny):
      for i in range(1, nx):
         T_new[j,i]=T_old[j,i]
         if j==0:
            T_new[j,i]+=dt*(alpha/dy**2+gam*u_y[j,i]/dy)*T_old[j,i]
            T_new[j,i]+=dt*(alpha/dy**2)*T_old[j+1,i]
         elif j==ny-1:
            T_new[j,i]+=dt*(alpha/dy**2+gam*u_y[j,i]/dy)*T_old[j-1,i]
            T_new[j,i]+=dt*(alpha/dy**2)*T_old[j,i]
         else:
            T_new[j,i]+=dt*(alpha/dy**2+gam*u_y[j,i]/dy)*T_old[j-1,i]
            T_new[j,i]+=dt*(alpha/dy**2)*T_old[j+1,i]
         if i==nx-1:
            T_new[j,i]+=dt*(alpha/dx**2+gam*u_x[j,i]/dx)*T_old[j,i-1]
            T_new[j,i]+=dt*(alpha/dx**2)*T_old[j,i]
         else:
            T_new[j,i]+=dt*(alpha/dx**2+gam*u_x[j,i]/dx)*T_old[j,i-1]
            T_new[j,i]+=dt*(alpha/dx**2)*T_old[j,i+1]
         T_new[j,i]+=dt*(-2*alpha/dx**2-2*alpha/dy**2-gam*u_x[j,i+1]/dx
               -gam*u_y[j+1,i]/dy)*T_old[j,i]
   T.append(T_new)
   #print(T_new)

#change of temperature in a single point over time
```

```
Tpunt = [T[i][int(ny/2),int(nx-1)] for i in range(len(T))]
tpunt = [dt*i/3600 for i in range(len(T))]
Tana={}

plt.plot(tpunt,Tpunt)
#plt.ylim((15,80))
plt.xlabel('$\it{t}$ [h]')
plt.ylabel('$\it{T}$ [$^\circ$C]')
#plt.savefig('stability dt=104v2.jpg', dpi=1000)
plt.show()
T=T[::100]

print(dt, alpha, dx, dy, np.max(u_x), np.max(u_y))

#animate the temperature distribution
def update(i):
   matrix.set_array(T[i])
   return matrix

fig, ax = plt.subplots()
matrix = ax.imshow(T[0])
plt.colorbar(matrix)

ani = animation.FuncAnimation(fig, update, frames = len(T), interval = 100)
writer = PillowWriter(fps=30)
#ani.save('v4 perm 1+blok a-6 dt 100.gif', writer=writer)
```

B.2. Analytical pressure and velocity field

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

#from cartesian to x-lexicographic ordering
def lex(i, j, N):
   return i+j*N
def p_ana(x, u0, p0):
   #return -u0*(np.log(1+np.exp(x))-np.log(2))+p0
   return -u0*x+p0

#discretization
L=20     #m
nx=100
dx=L/nx
H=11.25  #m
ny=75
dy=H/ny
print(dx,dy)
#and other variables
u0 = 0.0000317 #m/s from 1km/1y
rho = 10**3


#define grid
#grid points for scalar quantities
x = [dx/2+i*dx for i in range(nx)] #x_0 to x_(n-1)
y = [dy/2+j*dy for j in range(ny)] #y_0 to y_(n-1)

#crossing points of the grid
```

```
bx = [i*dx for i in range(nx+1)] #bx_0 to bx_n
by = [j*dy for j in range(ny+1)] #by_0 to by_n



#ANALYTICAL SOLUTION
def p0(x,y,N):
   u0=0.0000317
   L=20
   H=11.25
   val =0
   for i in range(N):
      if i==0:
         val+=-u0/2/L/H*x**2
      else:
         val+=(2*H*u0*(-1)**i/i**2/np.pi**2/L)*np.cos(i*np.pi*y/H)
   return val
def p1(x,y,N):
   u0=0.0000317
   L=20
   H=11.25
   val =0
   for i in range(N):
      if i==0:
         val+=u0/2/L/H*y**2
      else:
         val+=(-2*L*u0*(-1)**i/i**2/np.pi**2/H)*np.cos(i*np.pi*x/L)
   return val

def p0elmnt(x,y,N):
   u0=0.0000317
   L=20
   H=11.25
   if N==0:
      return -u0/2/L/H*x**2
   else:
      return(2*H*u0*(-1)**N/N**2/np.pi**2/L)*np.cos(N*np.pi*y/H)

def p1elmnt(x,y,N):
   u0=0.0000317
   L=20
   H=11.25
   if N==0:
      return u0/2/L/H*y**2
   else:
      return (-2*L*u0*(-1)**N/N**2/np.pi**2/H)*np.cos(N*np.pi*x/L)


#find the analytical solution for eigenfunction expansion for y and x
py = np.zeros((ny, nx))
px = np.zeros((ny, nx))
for i in range(nx):
   for j in range(ny):
      py[j,i]=p0(i*dx+dx/2, j*dy+dy/2, 2000)
      px[j,i]=p1(i*dx+dx/2, j*dy+dy/2, 2000)

#see if the analytical sum converges

#errorx = []
#errory = []
```

```
#errorxy = []
#Npoints = []
#py = np.zeros((ny, nx))
#px = np.zeros((ny, nx))
#for n in range(250):
#   print(n)
#   for i in range(nx):
#      for j in range(ny):
#         py[j,i]+=p0elmnt(i*dx+dx/2, j*dy+dy/2, n)
#         px[j,i]+=p1elmnt(i*dx+dx/2, j*dy+dy/2, n)
#   py -= py[0,0]-p[0,0]
#   px -= px[0,0]-p[0,0]
#   Npoints.append(n)
#   errorx.append(np.max(np.abs(p-px)))
#   errory.append(np.max(np.abs(p-py)))
#   errorxy.append(np.max(np.abs(py-px)))
#plt.plot(Npoints, errorx, label = 'diff ana and px')
#plt.plot(Npoints, errory, label = 'diff ana and py')
#plt.plot(Npoints, errorxy, label = 'diff px and py')
#plt.legend()
#plt.show()

#NUMERICAL SOLUTION
#define kappa := permeability/viscosity
#define different cases
case = 1       #1:uniform grid, 2:1+e^-x, 3:block with 0.5 and 1 else, 4:block
    with 0.5 and 1+e^-x else

def kappa(x,y,case):
   if case == 1:
      return 1
   elif case == 2:
      return 1+np.exp(-x)
   elif case == 3:
      if 4<x<8 and 3<y<5:
         return 0.00000005
      #elif 10<x<13 and 3<y<5:
       #  return 0.00000005
      else:
         return 1
   elif case == 4:
      if 4<x<8 and 3<y<5:
         return 0.5
      else:
         return 1+np.exp(-x)
   else:
      print('Invalid value for case')

#define A and f
A=np.zeros((nx*ny, nx*ny))
f=np.zeros((nx*ny,1))
k=0
for i in range(nx):
   for j in range(ny):
      kap = kappa(x[i],y[j],case)
      if j==0:    #boundary condition at y=0
         kap_N = 1/(0.5/kap+0.5/kappa(x[i],y[j+1],case))
         A[k, lex(i,j+1, nx)]-=kap_N/dy**2
         A[k, lex(i,j,nx)]+=kap_N/dy**2
      elif j==ny-1: #boundary condition at y=H
         kap_S = 1/(0.5/kap+0.5/kappa(x[i],y[j-1],case))
```

```
            A[k, lex(i,j-1,nx)]-=kap_S/dy**2
            A[k, lex(i,j,nx)]+=kap_S/dy**2
            f[k,0]+= u0/L/dy
        else:
            kap_N = 1/(0.5/kap+0.5/kappa(x[i],y[j+1],case))
            kap_S = 1/(0.5/kap+0.5/kappa(x[i],y[j-1],case))
            A[k, lex(i,j+1, nx)]-=kap_N/dy**2
            A[k, lex(i,j-1,nx)]-=kap_S/dy**2
            A[k, lex(i,j,nx)]+=(kap_N+kap_S)/dy**2
        if i==0:    #boundary condition at x=0
            kap_E = 1/(0.5/kap+0.5/kappa(x[i+1],y[j],case))
            A[k, lex(i+1,j,nx)]-=kap_E/dx**2
            A[k, lex(i,j,nx)]+=kap_E/dx**2
            k+=1
        elif i==nx-1: #boundary condition at x=L
            kap_W = 1/(0.5/kap+0.5/kappa(x[i-1],y[j],case))
            A[k, lex(i-1,j,nx)]-=kap_W/dx**2
            A[k, lex(i,j,nx)]+=kap_W/dx**2
            f[k,0] += -u0/dx/H
            k+=1
        else:
            kap_E = 1/(0.5/kap+0.5/kappa(x[i+1],y[j],case))
            kap_W = 1/(0.5/kap+0.5/kappa(x[i-1],y[j],case))
            A[k, lex(i+1,j,nx)]-=kap_E/dx**2
            A[k, lex(i-1,j,nx)]-=kap_W/dx**2
            A[k, lex(i,j,nx)]+=(kap_W+kap_E)/dx**2
            k+=1
#solve for p
p = np.linalg.solve(A,f)
p = np.reshape(p, (ny,nx))
p-=p[0,0]-py[0,0]

#solve for u
u_x = np.zeros((ny, nx+1))
u_y = np.zeros((ny+1, nx))
u_x[:,0] = 0
u_x[:,nx] = u0/H
u_y[0,:] = 0
u_y[ny,:] = -u0/L
for i in range(1,nx):
    u_x[0,i]=-1/(0.5/kappa(x[i-1],y[0],case)+0.5/kappa(x[i],y[0],case))
        *(p[0,i]-p[0,i-1])/dx
    for j in range(1,ny):
        u_y[j,0]=-1/(0.5/kappa(x[0],y[j-1],case)+0.5/kappa(x[0],y[j],case))
            *(p[j,0]-p[j-1,0])/dy
        u_x[j,i]=-1/(0.5/kappa(x[i-1],y[j],case)+0.5/kappa(x[i],y[j],case))
            *(p[j,i]-p[j,i-1])/dx
        u_y[j,i]=-1/(0.5/kappa(x[i],y[j-1],case)+0.5/kappa(x[i],y[j],case))
            *(p[j,i]-p[j-1,i])/dy

#plot pressure field and velocity field with streamfunction
stream = np.zeros((ny+1, nx+1))
for i in range(nx+1):
    for j in range(1, ny+1):
        stream[j,i]=stream[j-1,i]+dy*u_x[j-1,i]
plt.matshow(np.flip(p/np.abs(p[0,0]),0), vmin=minmin, vmax=maxmax)
plt.colorbar()
plt.contour(np.array(bx)/L*nx, np.array(by)/H*ny, np.flip(stream,0),
    colors='black')
plt.xlabel('$\it{x}$ [m]')
plt.ylabel('$\it{y}$ [m]')
```

```
plt.xticks([i-1/2 for i in xlab], [round(i/nx*L,2) for i in xlab])
plt.yticks([i-1/2 for i in ylab], [round((i-ny%10)/ny*H,2) for i in ylab[::-1]])
plt.tick_params(axis="x", top=False, bottom=True, labelbottom=True,
    labeltop=False)
plt.xlim(-1/2,nx-1/2)
plt.ylim(ny-1/2,-3/4)

plt.savefig('num 2d stream.jpg', dpi=1000)
plt.show()

#CHECK DIFFERENCE ANALYTICAL AND NUMERICAL
diff = np.zeros((ny,nx))
diffxy = np.zeros((ny,nx))
diffx = np.zeros((ny,nx))
for i in range(nx):
   for j in range(ny):
      diff[j,i]=(p[j,i]-py[j,i])/np.max(p)*100
      if diff[j,i]<-10:
         print(i,j,p[j,i], py[j,i])
      diffxy[j,i]=(py[j,i]-px[j,i])/np.max(py)*100 #difference between the two
          eigenfunction expansions
      diffx[j,i]=(p[j,i]-px[j,i])/np.max(px)*100 #difference between numerical
          and analytical

#set the same range for the colorbars
minmin = min(np.min(p/np.abs(p[0,0])),np.min(py/np.abs(py[0,0])),
    np.min(px/np.abs(px[0,0])))
maxmax = max(np.max(p/np.abs(p[0,0])),np.max(py/np.abs(py[0,0])),
    np.max(px/np.abs(px[0,0])))


xlab=[i*10 for i in range(nx//10+1)]
ylab=[i*10+ny%10 for i in range(ny//10+1)]

#plot numerical solution with velocity arrows
step=6
plt.matshow(np.flip(p/np.abs(p[0,0]),0))
plt.colorbar()
plt.xlabel('x')
plt.ylabel('y')
plt.xlabel('$\it{x}$ [m]')
plt.ylabel('$\it{y}$ [m]')
plt.xticks([i-1/2 for i in xlab], [round(i/nx*L,2) for i in xlab])
plt.yticks([i-1/2 for i in ylab], [round((i-ny%10)/ny*H,2) for i in ylab[::-1]])
plt.tick_params(axis="x", top=False, bottom=True, labelbottom=True,
    labeltop=False)
plt.xlim(-1/2,nx-1/2)
plt.ylim(ny-1/2,-3/4)
plt.quiver(mx[::step], my[::step][::-1],
    list(np.reshape(mu_x,(1,nx*ny))[0])[::step],
    list(np.reshape(mu_y,(1,nx*ny))[0])[::step], linewidths=3, width=0.003)
plt.savefig('num 2d velocity.jpg', dpi=1000)
plt.show()

#plot difference numerical and analytical
plt.matshow(np.flip(diff,0))
plt.xlabel('$\it{x}$ [m]')
plt.ylabel('$\it{y}$ [m]')
plt.xticks([i-1/2 for i in xlab], [round(i/nx*L,2) for i in xlab])
plt.yticks([i-1/2 for i in ylab], [round((i-ny%10)/ny*H,2) for i in ylab[::-1]])
```

```
plt.tick_params(axis="x", top=False, bottom=True, labelbottom=True,
    labeltop=False)
plt.xlim(-1/2,nx-1/2)
plt.ylim(ny-1/2,-3/4)
cb = plt.colorbar()
cb.set_label('%', rotation = 'horizontal')
plt.savefig('ana 2d flip numvspyv4.jpg', dpi=1000)
plt.show()

#plot difference analytical eigenfunction expansions
plt.matshow(np.flip(diffx,0))
plt.xlabel('$\it{x}$ [m]')
plt.ylabel('$\it{y}$ [m]')
plt.xticks([i-1/2 for i in xlab], [round(i/nx*L,2) for i in xlab])
plt.yticks([i-1/2 for i in ylab], [round((i-ny%10)/ny*H,2) for i in ylab[::-1]])
plt.tick_params(axis="x", top=False, bottom=True, labelbottom=True,
    labeltop=False)
plt.xlim(-1/2,nx-1/2)
plt.ylim(ny-1/2,-3/4)
cb = plt.colorbar()
cb.set_label('%', rotation = 'horizontal')
plt.savefig('ana 2d flip numvspxv4.jpg', dpi=1000)
plt.show()
```

B.3. Analytical temperature distribution

```
   import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import bisect


#and other variables
u0 = 0.0000317 #m/s from 1km/1y
lam_w=0.6      #W/mK
rho_w = 10**3  #kg/m^3
c_w = 4.18*10**3 #J/kgK
phi = 0.2
lam_s = 1.1    #W/mK
rho_s = 2.6*10**3 #kg/m^3
c_s = 0.8*10**3 #J/kgK


#coefficients for heat equation
alpha0 = (lam_w+lam_s)/(c_w*rho_w*phi+c_s*rho_s*(1-phi))
nu = c_w*rho_w/(c_w*rho_w*phi+c_s*rho_s*(1-phi))

gam2 = alpha0*10**3
beta = nu*u0
#beta = 0
dT = 80-15
L=20

nx=100
dx=L/nx
N=100
per = 0.0000001*np.pi/L #pertubation from asymptotes tangens

def trans(gam2, beta, l): #equation for eigenvalues
   return np.tan(np.sqrt(l)*L)+2*gam2*np.sqrt(l)/beta
```

```
def initial_smooth(x, L): #a smooth initial function
   return -(x-L)**2+L**2
def initial(x,L,dx, dT): #initial function
   if x<2*dx/3:
      return 0
   else:
      return dT
def orthogonal(a,b,L): #check orthogonality
   return
       b/(a**2-b**2)*np.sin(a*L)*np.cos(b*L)-a/(a**2-b**2)*np.cos(a*L)*np.sin(b*L)
f = lambda l: trans(gam2, beta, l)

if beta>0: #find eigv via function trans
   eigv = []
   for i in range(1, N+1):
      eigv.append(bisect(f, ((i-1/2)*np.pi/L)**2+per, ((i+1/2)*np.pi/L)**2-per))
else:    #we know the eigenvalues
   eigv = [((i-1/2)*np.pi/L)**2 for i in range(1, N+2)]

xgrid = [dx/2+i*dx for i in range(nx)]
print(eigv[:10])

#check for orthogonality
print(L/2-1/4/eigv[0]*np.sin(2*eigv[0]*L))
for b in eigv[1:10]:
   print(round(b,3))
   print(round(orthogonal(eigv[0], b, L),3))

#print the initial function
plt.plot(xgrid, [initial(x, L, dx, dT) for x in xgrid])
plt.show()

#set up matrix equation to find coefficients c_n
A = np.zeros((nx,N))
b = np.zeros((nx,1))
for i in range(nx):
   b[i,0]=initial_smooth(xgrid[i], L)       #initial function is smooth
       v(x,0)=-(x-L)**2-L
   b[i,0]=initial(xgrid[i], L, dx, dT)
   for j in range(N):
      A[i,j]=np.sin(np.sqrt(eigv[j])*xgrid[i])*np.exp(beta*xgrid[i]/2/gam2)
c = np.linalg.solve(A,b)

#term in summation
def Telement(x,t,n, gam2, beta, L, dT, eigv_n, c):
   return np.exp(-beta**2*t/4/gam2-eigv_n*gam2*t)*np.sin(np.sqrt(eigv_n)*x)
         *np.exp(beta*x/2/gam2)*c[n-1]

Tana={}     #store the cumulative summation
Tanas=[]    #store the elements of the summation separately
plt.figure()
for t in [0, 2000, 10000]:
   Tana[t]=[]
   for n in range(1, len(eigv)):
      Tana[t].append([])
      Tanas.append([])
      i=0
      for x in xgrid:
         Tanas[n-1].append(Telement(x,t,n, gam2, beta, L, dT, eigv[n-1], c))
         if n==1:
```

```
            Tana[t][0].append(float(Telement(x,t,n, gam2, beta, L, dT, eigv[n-1],
                c)))
        else:
            Tana[t][n-1].append(float(Telement(x,t,n, gam2, beta, L, dT,
                eigv[n-1], c))+Tana[t][n-2][i])
        i+=1

    print(Tana[t][n-1])
    plt.plot(xgrid, np.array(Tana[t][n-1])+15, label='t='+str(round(t/3600,
        3))+'h')
plt.xlabel('$\it{x}$ [m]')
plt.ylabel('$\it{T}$ [$^\circ$C]')
plt.legend()
plt.savefig('ana heat eqv1.jpg', dpi=1000)
plt.show()

#see if the summation converges for a given x (here x=L)
plt.plot([i for i in range(len(Tana))], [i[int(nx-1)] for i in Tana], label='som')
plt.plot([i for i in range(len(Tana))], [i[int(nx-1)] for i in Tanas],
    label='losse termen')
plt.legend()
#plt.plot([i for i in range(len(Tana))], [np.max(np.abs(Tmasksom))],
    label='makkelijke som')
plt.show()
```