Groundwater I

P. K. Yadav, T. Reimann and several others

Dec 06, 2020

BACKGROUND

The site provides an interactive JUPYTER book with contents typical of a *introductory* groundwater course taught at higher UG level or the early PG level at universities.

The contents/structure provided here are mostly from those developed and lectured by **Prof. Rudolf Liedl** at TU Dresden.

The contents are geared towards **learning through computing**. The computing part is entirely based on *Python* programming language. Previous programming/coding experiences is not required or expected to gain from the provided contents.

The contents of the book are divided into:

1. Lecture Parts
2. Tutorial Parts
3. Self-learning tools

The *lecture* parts are combination of **texts** and **simpler** numerical examples. Only minimum *Python* codes are available on this part. The *tutorial* are mostly **numerical** examples. This aims at teaching also the basic of Python coding. The focus remains on illustrating lecture contents. *Self-learning tools* are interactive tools that supports the lecture and tutorial components and enhances understanding. The codes of the tools can require higher knowledge in Python programming. Therefore, codes are hidden.

The development of the book is based on the wonderful works of the JUPYTER Book Team

CHAPTER

ONE

MAIN CONTRIBUTORS OF THIS VERSION OF THE BOOK

The contents are developed by (not in any order):

- Prof. Rudolf Liedl (TU Dresden)
- Prof. Peter Dietrich (UFZ Leipzig/Uni-Tübingen)
- Prof. B. R. Chahar (Indian Institute of Technology Delhi, Delhi)
- Prof Charles Werth (Uni-Texas Austin, US)
- Dr.rer. nat. Prabhas K Yadav (TU Dresden)
- Dr. Ing. Thomas Reimann (TU Dresden)
- M.Sc. Hanieh Mehrdad (Student Assistant/Numerical contents- TU Dresden)

and several students: Anton, Abhiral, Anne, Sophie, Alexander

CHAPTER

TWO

ACKNOWLEDGMENTS

This work is partly supported by:

1. The Multimediafonds, TU-Dresden
2. The ESTIMATE (LI 727/29-1), project, DFG

2.1 About this Groundwater Course and Contents

The contents provided in this website that forms an interactive book, is based on the Groundwater course created, maintained and lectured by **Prof. Rudolf Liedl** for over 15 years at the Institute of Groundwater Manangement of TU Dresden.

The course structure and contents have been used as an *Introductory* course on Groundwater for M.Sc. level students coming from multiple academic backgrounds.

The **texts** provided in this interactive book is mostly based on the Prof. Liedl's lecture slides and the **interactive codes** are conversion from MS Excel® spreadsheet to *Python* codes.

> **This interactive web-book is dedicated to Prof. Rudolf Liedl efforts to teach, train and inspire us and other students over the years.**

2.1.1 Basic contents structure

The contents of this interactive book can be broadly divided in the following three groups:

1. Aquifer properties and groundwater flow
2. Transport in groundwater
3. Groundwater Modelling

The first group (**aquifer properties and groundwater flow**) makes the core of this course. Here the very basic of groundwater, subsurface structure, properties that quantify groundwater mass and volume budgets, and the flow and other dynamics processes, e.g., abstraction using wells.

The **transport in groundwater** topics focus on the quality aspects of groundwater. In particular, transport equations with and without inclusion of chemical reactions (e.g., sorption, decay) are considered. Eventually, few analytical solutions of transport problems are discussed.

The **Groundwater Modelling** is for introducing the realm of computer modelling of groundwater and transport. Fundamentals of mathematical modelling, e.g., finite different methods, is introduced. The focus remains towards eventual use of MODFLOW (Flopy and Modelmuse interface), which is introduced in a short tutorial form.

2.1.2 What do you need in this course?

You will need the following:

- Laptop/Smartphone/tablet more convenient with internet connection
- Recommended is installed JUPYTER interface.

The course contents can be received as:

- Lectures: Texts reading and interactive manipulation of codes/problems. Short questions (self-test) should be used to check the understanding of the contents.
- Tutorials: Should be understand by manipulating the existing codes. The tutorials should be then independently solved using *JUPYTER* interface (Mybinder or Personal system)
- Additional tools: The course provides several simulation tools- e.g., sieve analysis, effective conductivity, Advection-dispersion etc. These tools (also *Python*) provide high-level interactivity. These tools should be used to enhance learning.
- Question banks/exam questions should be used to self-check the level of understanding.

```
# required libraries
import numpy as np
import matplotlib.pyplot as plt
import panel as pn
import pandas as pd
pn.extension("katex")


#from IPython.display import Image, Video

#import warnings
#warnings.filterwarnings('ignore')
```

2.2 Lecture 1 - Course Introduction/Water Cycle

(The contents presented in this section were re-developed principally by Dr. P. K. Yadav. The original contents were developed by Prof. Rudolf Liedl)

2.2.1 Contents of "Groundwater Module "

The entire contents of this interactive book can be listed with the following points:

- general overview
- types of aquifers, properties of aquifers
- forces and pressure in the subsurface
- laws of groundwater flow and some applications (e.g. groundwater wells)
- quantification of aquifer parameter values via pumping tests
- transport of chemicals (solutes) in groundwater
- retardation and degradation of chemicals in groundwater
- groundwater modelling

2.2.2 Suggested Literature:

- Brassington R. (1988): Field hydrogeology, Wiley & Sons.
- Domenico P. A., Schwartz F. W. (1990): Physical and chemical hydrogeology, Wiley & Sons.
- Fetter C. W. (2001): Applied hydrogeology, Prentice Hall.
- Freeze R. A., Cherry J. A. (1979): Groundwater, Prentice Hall.
- Heath R. C. (1987): Basic groundwater hydrology, USGS Water Supply Paper 2220.
- Price M. (1996): Introducing groundwater, Chapman and Hall.

Additional literature details are provided in the text when used.

2.2.3 What is Hydrogeology?

Hydrogeology is the study of the laws governing the movement of subterranean water, the mechanical, chemical, and thermal interaction of this water with the porous solid, and the transport of energy and chemical constituents by the flow. (Domenico and Schwartz, 1990)

The dominant reliance of groundwater for the drinking globally has made hydrogeology a very important academic course. Also, it is a very important research field. Therefore, several **techniques** and **methods** are now available to explore and understand **Hydrogeological Process**. The methods and techniques can be broadly categorized to:

1. Field works
2. Laboratory experiments
3. Computer modeling

Computer modelling is often the most economical method but its usefullness rely of data obtained from *Field works* and *Laboratory experiments.* Thus, the sequence of techniques/methods to be adopted depends on the available site information.

```
im1 = pn.pane.PNG("images/L01_f_1c.png", width=250)
im2 = pn.pane.PNG("images/L01_f_1b.png", width=275)
im3 = pn.pane.PNG("images/L01_f_1a.png", width=280)

pn.Row(im1, im2, im3)
```

```
Row
    [0] PNG(str, width=250)
    [1] PNG(str, width=275)
    [2] PNG(str, width=280)
```

2.2.4 Example: Groundwater Extraction Well

Groundwater is extracted using a groundwater well applying *hydrogeological* methods and techniques. The procedure followed can be summarized in the following steps:

1. The appropriate extraction location is identified
2. Drilling machine are used to obtain sub-surface structure, i.e. or well logs are obtained. The process is also called well logging.
3. Well logs are studied in detail to identify the characteristics of the subsurface- e.g., how thick is the aquifer or identify environmental consequence of water extraction.

4. The construction of well begins

Groundwater extraction using well is a challenge when aquifers are located very deep from the surface, e.g., in deserts.

```
video1 = pn.pane.Video("images/L01_f_2.mp4", width=600, height=400, loop=False)
video1
```

```
Video(str, height=400, sizing_mode='fixed', width=600)
```

```
#gif_pane = pn.pane.GIF('images/L01_f_2.gif', width=500)
#gif_pane
video2 = pn.pane.Video("images/L01_f_3.mp4", width=600, height=400, loop=False)
#Video("images/L01_f_3.mp4", width=600, embed=True)
video2

pn1 = pn.pane.Markdown("""
**Wells** are placed on the layer or that aquifer part which allows feasible␣
↪extraction of groundwater.
The extraction leads to drop of groundwater level. To ensure that there is␣
↪sustainable extraction,
the drops in the level has to be monitored. Quite often this is done through _
↪computer modelling_. There
already exists several computer models that can use the well logs data (also called␣
↪Borehole) and provide
good estimations of the effects due to extraction. The _computer models_ are also␣
↪able to predict the effects
at larger scales, e.g., regional scales. _Computer models_ are oftenly used these␣
↪days be agencies to determine
quantities such as **travel time**, **capture zones** or obtain **isochrones**, which␣
↪are used for deciding on
groundwater extraction programmes.
""")

pn.Row(pn1, video2)
```

```
Row
    [0] Markdown(str)
    [1] Video(str, height=400, sizing_mode='fixed', width=600)
```

2.2.5 Groundwater and Global Water Cycle

Water bodies that exist on earth is connected, and they function as a cycle, called **Global Water Cycle**. It is estimated that over 57, 700 Km^3 of water actively participates in the cycle each year. **Precipitation** and **evaporation** are the two main components of the cycle in which **temperature** plays the critical role. In the cycle, **Groundwater** receives water from *precipitation,* It then contributes to *evaporation* through subsurface flow or through mostly human intervention (e.g., use for drinking water).

The water cycle provides an approach to judge the sustainability of groundwater extraction. The sustainability of extraction can be obtained if extraction rate approximately equals the replenishing rate. Often the replenishing rate of groundwater is much slower and this has led to groundwater stress in many parts of the world.

```
#gif_pane = pn.pane.GIF('images/L01_f_2.gif', width=500)
#gif_pane
video3 = pn.pane.Video("images/L01_f_4.mp4", width=600, height=400, loop=False)
#Video("images/L01_f_3.mp4", width=600, embed=True)
video3
```

```
Video(str, height=400, sizing_mode='fixed', width=600)
```

```
#gif_pane = pn.pane.GIF('images/L01_f_2.gif', width=500)
#gif_pane
fig5 = pn.pane.PNG("images/L01_f_5.png", width=600)
#Video("images/L01_f_3.mp4", width=600, embed=True)


pn1 = pn.pane.Markdown("""

### Water balance by continents

Groundwater receives water from the _infiltration_ of **runoff** water.

""")

pn.Row(pn1, fig5)
```

```
Row
    [0] Markdown(str)
    [1] PNG(str, width=600)
```

2.2.6 The Hydrological Balance

Since *groundwater* is part of the global water cycle, the balance of the cycle becomes an important topic. In general:

- The *hydrological balance* provides a relationship between various flow rates for a certain area. It is based on the conservation of water volume.
- expressed in words: *inflow* equals *outflow* plus *change in storage*
- expressed by a formula:

$$P = ET + R + \Delta S$$

Where,

where, P = *Precipitation,* ET = *Evapotranspiration,* R = *Runoff,* and ΔS = *Change in Storage*

The *change in storage* can be interpreted in the following way:

- change in storage $\Delta S > 0$: Water volume is increasing with time in the investigation area.
- change in storage $\Delta S < 0$:Water volume is decreasing with time in the investigation area.
- change in storage $\Delta S = 0$: Water volume does not change with time in the investigation area (steady-state or stationary situation, i.e. inflow equals outflow).

2.2.7 Water Volume

So how much water do we have? It is estimated* that the total volume of water on Earth amounts to ca. 1 358 710 150 km^3 ($\approx$ 1018 m^3).

Percent shares:	
Saline water	97.2 %
Fresh water, liquid	0.65 %
Fresh water, ice	2.15 %

The total volume of fresh water on Earth amounts to ca. 38×10^6 km^3 ($\approx$ 1016 m^3).

**Gleick P. (1996): Water re-sources, in: Schneider S. H. (ed.), Encyclopedia of climate and weather 2, Oxford Univ. Press.*

2.2.8 Volume of Available Fresh Water

Fresh water are water with low concentrations of dissolved salts and other total dissolved solids, i.e., sea/ocean water or brackish water are not fresh water. Human activities (drinking water) are directly dependent on fresh activities.

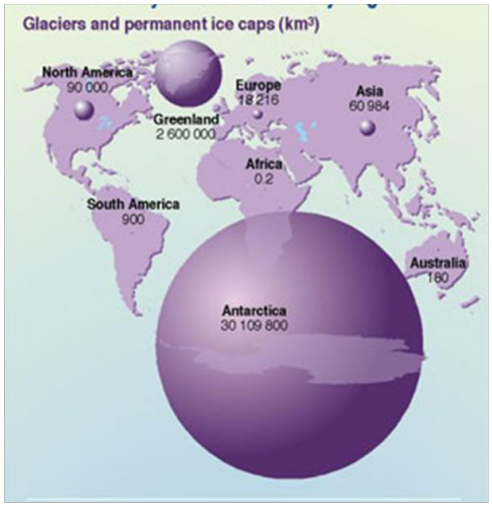
So how much *fresh water* do we have?

It is estimated* that the total volume of available fresh water (liquid) on Earth amounts to ca. 8 831 600 km^3 ($\approx$ 1016 m^3).

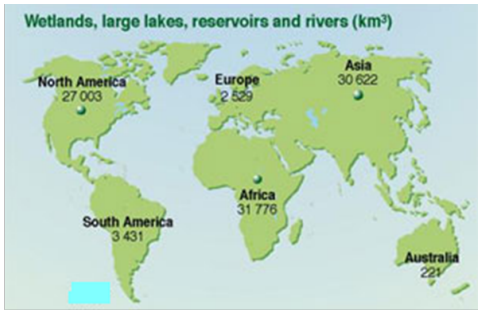
Percent shares:	
Groundwater	97,55 %
Vapour	0,15 %
Soil water	0,80 %
Lakes and rivers	1,50 %

**Gleick P. (1996): Water re-sources, in: Schneider S. H. (ed.), Encyclopedia of climate and weather 2, Oxford Univ. Press.*

2.2.9 Continental distribution of fresh water components



http://www.unep.org/dewa/vitalwater/article32.html



includes saline groundwater!

2.2.10 Volume and Mass Budget

Very basics of volume and mass budget - let us start with *budget.*

Budget = quantitative comparison of *growth* (or *production*) and *loss* in a system

Budgets can be put together for various quantities:

- energy
- mass ← needed to quantify transport of solutes in groundwater
- volume ← needed to quantify groundwater flow
- momentum
- electric charge
- number of inhabitants
- animal population
- money (bank account!)
- and many others

In this course we focus on *Mass Budget* and *Volume Budget.*

2.2.11 Volume Budget

As discussed in the last topic a *budget* represents the change (e.g., growth and loss). Thus, it is more suitable to quantify the **volume budget** in terms of a change, representing two different states (e.g., time (t)). More formally, the **volume budget** (ΔV) can be obtained from:

$$\Delta V = Q_{in} \cdot \Delta t - Qout \cdot \Delta t$$

Δt = time interval [T] ΔV = change of volume in the system [L^3] Q_{in} = volumetric rate of flow into the system [L^3/T] Q_{out} = volumetric rate of flow out of the system [L^3/T]

The following points have to be considered when using the above equation:

- Inflow and outflow may each consist of several individual components.
- $\Delta V = 0$ (no change in volume) is tantamount to steady-state or stationary (= time-independent) conditions.
- For steady-state conditions we have: $Q_{in} = Q_{out}$

Water Budget for a Catchment

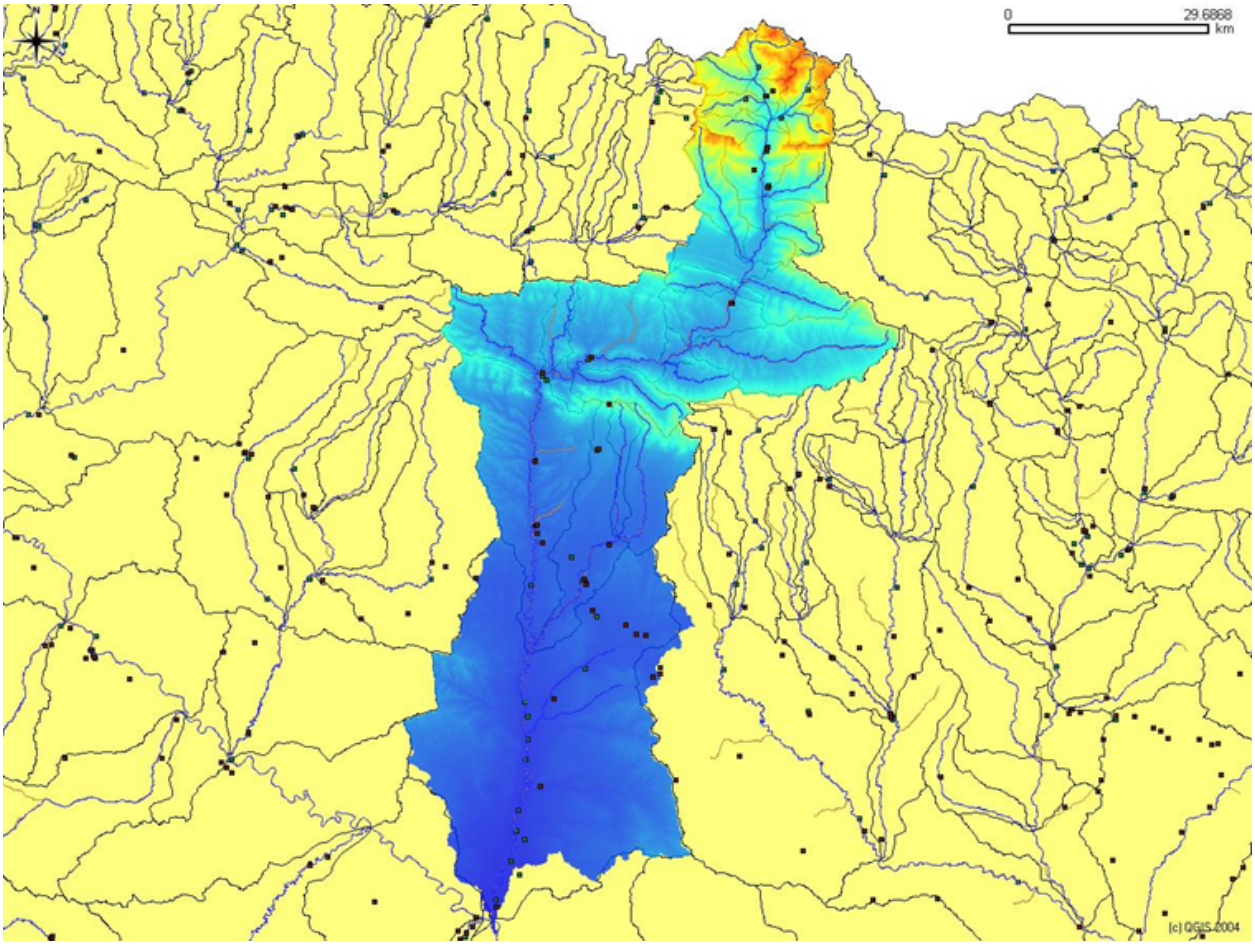
The equation provided for *volume budget* looks simple but in practice it is very complicated as several *inflow* and *outflow* components must be considered. Quantifying these components can be a challenging task.

For quantifying water budget of a catchment, one has to consider the following components:

To be considered:

- precipitation
- evapotranspiration
- surface runoff
- subsurface runoff

Among the above components, quantification of evapotranspiration and subsurface runoff have very high level of uncertainties.



2.2.12 Example: Estimation of Subsurface Runoff

Most numbers used in the example do not refer to the catchment shown before!

To calculate the following four-steps are to be followed:

- Step 1: determine rate of inflow in m^3/a
- step 2: determine rate of outflow due to evapotranspiration (ET.A) in m^3/a
- Step 3: express rate of outflow due to surface runoff in m^3/a
- step 4: determine rate of outflow due to subsurface runoff

An Example: For given data, determine the rate of outflow Qout,sub due to subsurface runoff for steady-state conditions

```
A = 4500 # km^2, catchment area
P = 550 # mm/a, precipitation
ET = 200 # mm/a, evapotranspiration
Qout_surf = 40 # m^3/s, surface runoff
Delta_V = 0 # m^3, change in volume = 0 Steady-state conditions

#Volume budget in this example: P·A = ET·A + Qout,surf + Qout,sub

#Step 1
```

(continues on next page)

(continued from previous page)

```
Qin = P*A*10**3  #m^3/a, 10^3 for unit conversion

#step 2:
ET_A = ET*A*10**3 #m^3/a, 10^3 for unit conversion

#Step 3:
Qout_surf = Qout_surf *365*24*3600 #  m^3/a

# step 4:
Qout_sub = Qin - ET_A - Qout_surf # m^3/a



print("The rate of inflow, Qin is {0:1.1E}".format(Qin),"m\u00b3/a \n"); print("The␣
↪outflow rate due to Evapotranspiration is {0:1.1E}".format(ET_A),"m\u00b3/a \n")
print("The surface outflow rate, Q_out_surf in m\u00b3/a is {0:1.1E}".format(Qout_
↪surf),"m\u00b3/a \n");print("The subsurface outflow rate, Qout_surf in m\u00b3/a is
↪{0:1.1E}".format(Qout_sub),"m\u00b3/a \n")
```

```
The rate of inflow, Qin is 2.5E+09 m³/a

The outflow rate due to Evapotranspiration is 9.0E+08 m³/a

The surface outflow rate, Q_out_surf in m³/a is 1.3E+09 m³/a

The subsurface outflow rate, Qout_surf in m³/a is 3.1E+08 m³/a
```

2.2.13 Mass Budget

The **mass budget** is quantified similar to the *volume budget.* Mathematically, the *mass budget* is:

$$\Delta M = J_{in} \cdot \Delta t - J_{out} \cdot \Delta t$$

with Δt = time interval [T] ΔM = change of mass in the system [M] J_{in} = rate of mass flow into the system [M/T] J_{out} = rate of mass flow out of the system [M/T]

Similar to *volume budget,* the following points have to be considered in quantifying mass budget:

- Inflow and outflow may each consist of several individual components.
- ΔM = 0 (no change in mass) is tantamount to steady-state or stationary (= time-independent) conditions.
- For steady-state conditions we have: J_{in}= J_{out}

2.2.14 Example of Mass Budget: Radioactive Decay

Consider a decay chain comprising of the three chemicals: **A**, **B** and **C**

- decay chain: A $\rightarrow$ B $\rightarrow$ C
- 30% of A and 20% of B decay each year.
- decay rate of A = production rate of B = $0.3 \cdot a^{-1} \cdot M_A$
- decay rate of B = production rate of C = $0.2 \cdot a^{-1} \cdot M_B$
- mass budgets for A, B and C:

$$\Delta M_A = 0.3\ \mathrm{a}^{-1} \cdot M_A \cdot \Delta t$$
$$\Delta M_B = 0.3\mathrm{a}^{-1} \cdot M_A \cdot \Delta t - 0.2\ \mathrm{a}^{-1} \cdot M_B \cdot \Delta t$$
$$\Delta M_C = 0.2\mathrm{a}^{-1} \cdot M_B \cdot \Delta t$$

- Similar equations hold for quantitative descriptions of some chemical reactions which correspond to the type A $\rightarrow$ B $\rightarrow$ C

```
def mass_bal(n_simulation, MA, MB, MC, R_A, R_B):

    A = np.zeros(n_simulation)
    B = np.zeros(n_simulation)
    C = np.zeros(n_simulation)
    time  = np.arange(n_simulation)

    for i in range(0,n_simulation-1):
        A[0] = MA
        B[0] = MB
        C[0] = MC
        A[i+1] = A[i]-R_A*A[i]
        B[i+1] = B[i]+R_A*A[i]-R_B*B[i]
        C[i+1] = C[i]+R_B*B[i]
        summ = A[i]+B[i]+C[i]

    d = {"Mass_A": A, "Mass_B": B, "Mass_C": C, "Total Mass": summ}
    df = pd.DataFrame(d) # Generating result table
    label = ["Mass A (g)", "Mass B (g)", "Mass C (g)"]
    fig = plt.figure(figsize=(6,4))
    plt.plot(time, A, time, B, time, C, linewidth=3);  # plotting the results
    plt.xlabel("Time [Time Unit]"); plt.ylabel("Mass [g]") # placing axis labels
    plt.legend(label, loc=0);plt.grid(); plt.xlim([0,20]); plt.ylim(bottom=0) #␣
↪legends, grids, x,y limits
    plt.show() # display plot

    df_pane = pn.pane.DataFrame(df)
    return print(df.round(2))

N = widgets.BoundedIntText(value=20,min=0,max=100,step=1,description= '&Delta; t (day)
↪',disabled=False)

A = widgets.BoundedFloatText(value=100,min=0,max=1000.0,step=1,description='M<sub>A</
↪sub> (kg)',disabled=False)

B = widgets.BoundedFloatText(value=5,min=0,max=1000.0,step=1,description='M<sub>B</
↪sub> (kg)',disabled=False)

C = widgets.BoundedFloatText(value=10,min=0,max=1000,step=0.1,description='M<sub>C</
↪sub> (kg)',disabled=False)

RA = widgets.BoundedFloatText(value=0.2,min=0,max=100,step=0.1,description='R<sub>A</
↪sub> (day<sup>-1 </sup>)',disabled=False)

RB = widgets.BoundedFloatText(value=0.2,min=0,max=100,step=0.1,description='R<sub>B</
↪sub> (day<sup>-1 </sup>)',disabled=False)


interactive_plot = widgets.interactive(mass_bal, n_simulation = N, MA=A, MB=B, MC=C,␣
↪R_A=RA, R_B=RB,)
output = interactive_plot.children[-1]
```

(continues on next page)

(continued from previous page)

```
#output.layout.height = '350px'
interactive_plot
```

```
interactive(children=(BoundedIntText(value=20, description='&Delta; t (day)'),␣
↪BoundedFloatText(value=100.0, d...
```

2.2.15 Comparison of Mass and Volume Budgets

mass budget: $\Delta M = J_{in} \cdot \Delta t - J_{out} \cdot \Delta t$

volume budget: $\Delta V = Q_{in} \cdot \Delta t - Q_{out} \cdot \Delta t$

- Mass and volume budgets are equivalent if there is no change of density ρ [M/L^3] with time. In this case the well known relationship $\Delta M = \rho \cdot \Delta V$ holds and each equation given above can be directly transformed into the other one.
- If density changes have to be considered (e.g. for gas flow), the mass budget equation remains valid but the volume budget equation must be modified because $\Delta M = \rho \cdot \Delta V + \Delta\rho \cdot V$ with $\Delta\rho$= change in density.
- Cases with changing density have proven to be more easily tractable if the mass budget equation is used.

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib widget
import warnings; warnings.simplefilter('ignore')
```

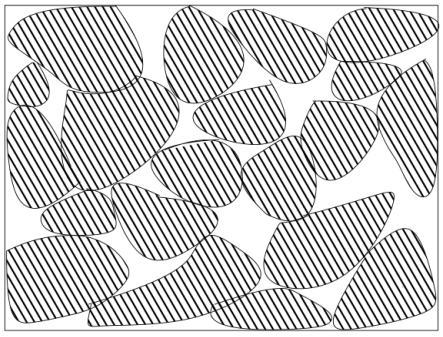
2.3 Lecture 2 - Subsurface Structure

(The contents presented in this section were re-developed principally by M.Sc. Hanieh Mehrdad and Dr. P. K. Yadav. The original contents are from Prof. Rudolf Liedl)

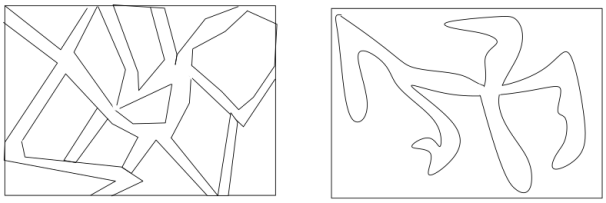
2.3.1 Porous Media

The general definition of the porous media is a **solid which contains voids**. This definition applies to the subsurface contains solid material plus voids which represent storage and transmission of the water. The voids may have various shapes and contain fluids (mostly air and/or water). Moreover, voids may be connected to or disconnected from each other. Generally voids and their properties are important to determine water storage (how much water is or could be available?) and water transmission (How fast the water can move?).

2.3.2 Types of porous media in the subsurface

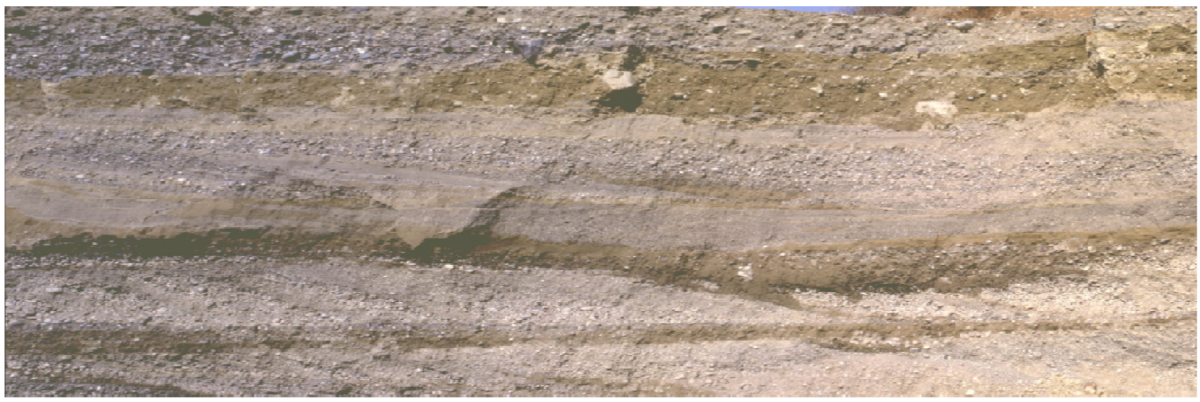


Unconsolidated media



Consolidated media

1. **unconsolidated porous medium (Sediments)**: it is non-cemented porous media and the grains can be taken away. The formation of such porous media is due to deposition of solid material mostly by water.



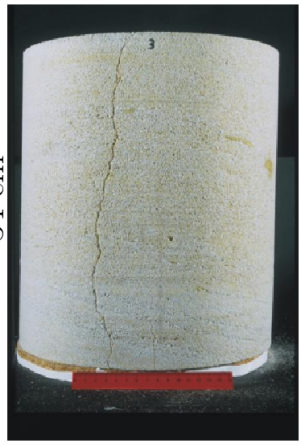
Sediments consisting of fluvial deposits of sands and gravels (Klingbeil, 1998)
(Rhine Basin – Southwest Germany / Northern Switzerland)

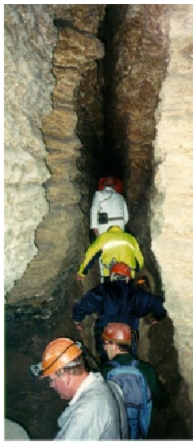
1. **Consolidated porous medium (Rocks)**: the formation is due to increased pressure acting together with thermal and chemical processes. It has two types:

   Fractured porous media

   Karstified porous media

Standstone
(South Germany)





Gypsum
(Western Ukraine)

Fractured porous medium:
Genesis of fractures or cracks due to mechanical or thermal processes (stress/strain or cooling/warming)

Krastified porous medium:
Genesis of fractures or tubes (conduits) due to chemical processes (i.e., dissolution of rock by water)

2.3.3 Porosity (Total porosity):

Is defined as the volumetric share of voids in a porous media. It is a number between 0 and 1 and can be expressed as percentage (0%: no voids, 100%:no solid)

$$n = \frac{V_v}{V_T}$$

- n= total porosity
- V_v= voids volume
- V_T= total volume

Example Problem

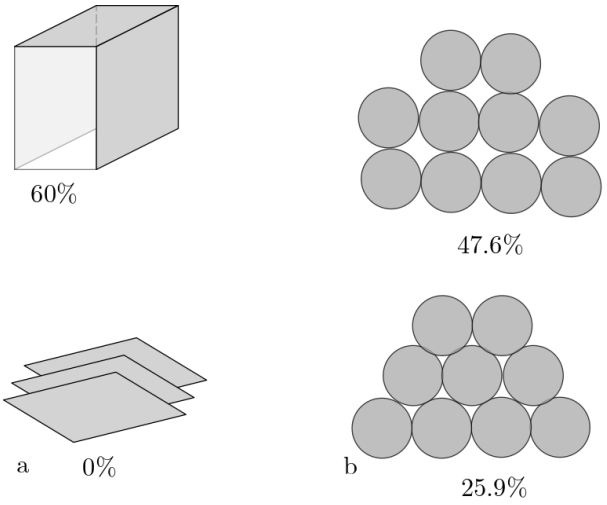
If the total volume of a media is 254 cubic meters, and the volume of the void is 27 cubic meters, what is the porosity (give as a percent)?

```
# input data
V_T= 254 #m^3 total volume
V_v=27 #m^3 voids volume

#calculations
n=(V_v/V_T)*100

# Output
print(" Total porosity is: {0:0.2f}%".format(n) )
```

```
 Total porosity is: 10.63%
```

2.3.4 Total porosity of artificial porous media:

If "grains" have identical shape and are regularly arranged, it is possible to exactly compute total porosity, the pores should have the same size.

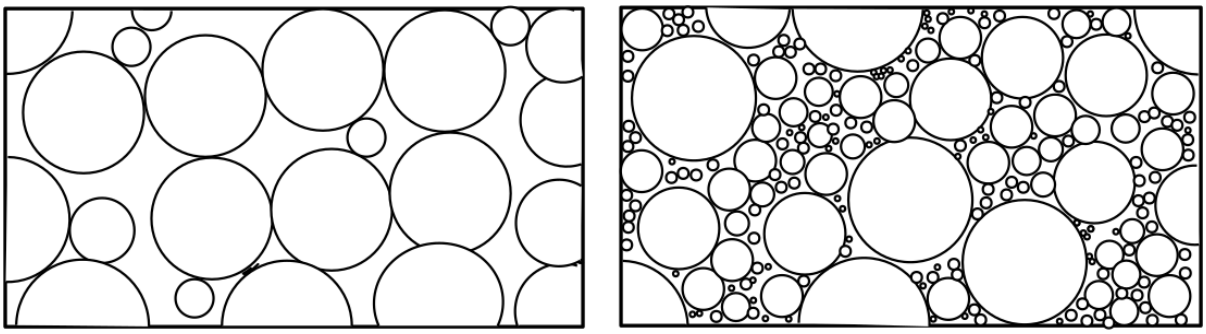


- Loose packing(first picture): each hole placed on top of the hole underneath
- Dense packing (second picture): each hole is placed at the deepest position possible

These schematics provides a practical range of porosity in the subsurface. The general range is between 25% to about 50%. In more extreme cases porosity higher than 60% is possible, e.g., cobbles, gravel. The other extreme, subsurface with no porosity (0%) is also encountered in the subsurface, e.g., in consolidated rocks.

2.3.5 Total porosity of natural (unconsolidated) porous media:

Natural unconsolidated porous media consist of grains of different size. Total porosity depends on the grain size distribution.



In general, well sorted unconsolidated porous media exhibit larger total poerosities than poorly sorted unconsolidated porous media. In the figure above on the left in which grain diameters cover a small range, i.e., **well-sorted**, the porosity can be approximated in the range 32%. Similary, in the (above) figure on the right in which the grain diameters cover a large range, i.e., **poorly sorted**, the porosity can be approximated in the range 17%.

2.3.6 Typical porosity values:

Table below provide the total porosity of unconsolidated and consolidated media.

Porosity - Unconolidated media

	grain diameter (mm)	Total Porosity (%)
Coarse gravel	20 - 60	24 - 36
Fine gravel	2 - 6	25 - 38
Coarse sand	0.6 - 2	31 - 46
Fine sand	0.006 - 0.2	26 - 53
Silt	0.002 - 0.06	34 - 61
Clay	< 0.002	34 - 60

Porosity - Conolidated media

	Total Porosity (%)
Siltstone	21 - 41
Sandstone	5 - 30
Basalt	3 - 35
Claystone	1 - 10
Limestone	1 - 10
Shale	< 10
Granite	< 1

Total porosity of consolidated porous media (rocks) is usually smaller than total porosity of unconsolidated porous media. However, weathering effect may lead to increase the value of porosity. only for unconsolidated porous media, total porosity tends to increase with decreasing grain size.

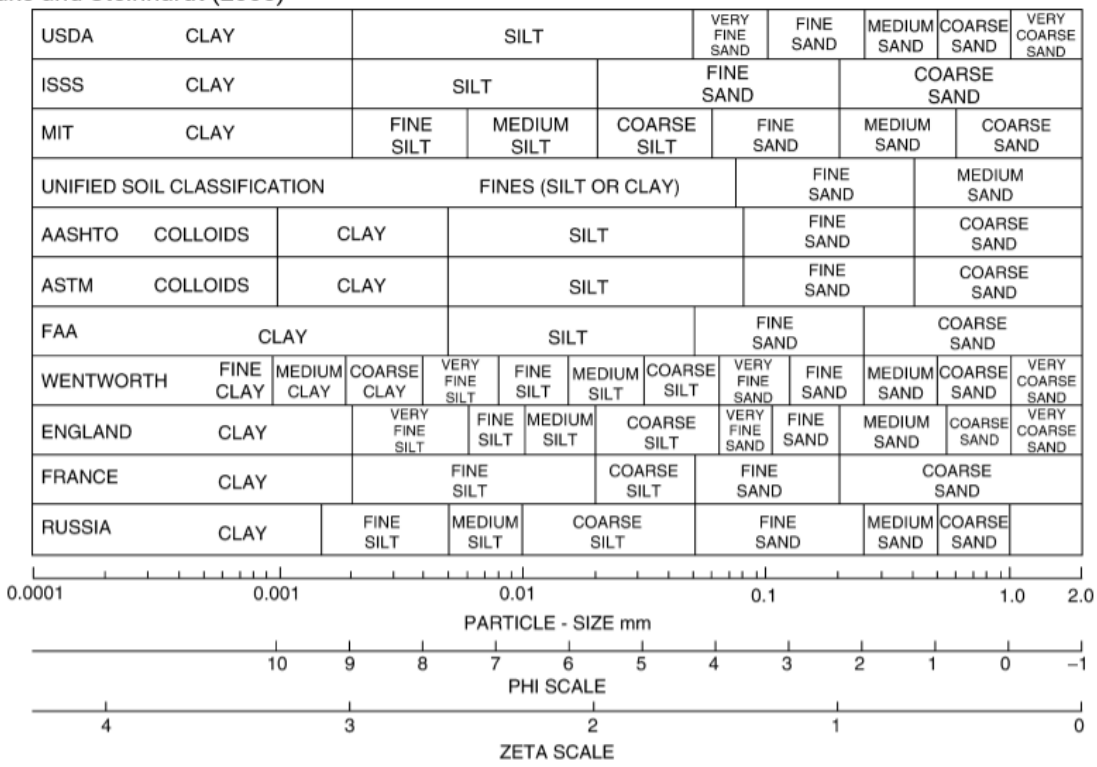
2.3.7 Grain size distribution of unconsolidated porous media

Unconsolidated porous media are able to store and transmit water that can be influenced by grain size distribution. Therefore, the grain size distribution is frequently determined in laboratory experiments in order to deduce important flow properties. There are five major grain size classes (observed by increasing diameter): clay, silt, sand, and gravel (or debris). The classes for silt, clay and gavel are usually subdivided by “fine”, “medium”, and “coarse” (or “very fine”, “fine”, “medium”, “coarse”, and “very coarse”). Different ranges for individual grain size classes have been defined by different authorities or regulations. However, the standard method to determine the grain size distribution of a sample is sieve analysis.

2.3.8 Classification schemes:

The diagrams below include a couple of classification schemes to define ranges of grain diameter for clay, silt, sand, and gravel:

Blake and Steinhardt (2008)



As can be observed that there exist several standards. These are often based on local requirements e.g., based on countries. In Germany the DIN standards are used.

Click for the abbreviation

USDA: United States Department of Agriculture, *ISSS: International Soil Science Society (ISSS)*, *MIT: Massachusetts Institute of Technology*, *ASTM: American Society for Testing and Materials*, *AASHTO: American Association of State Highway and Transportation Officials*, *FAA: Federal Aviation Administration*

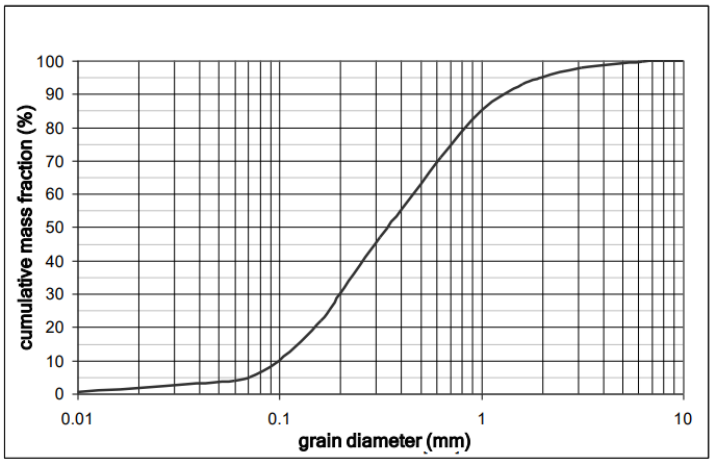
2.3.9 Sieve analysis:

The results from a sample consist of different grain size fractions should be transferred on granulometric curve. This curve provides cumulative information; vertical axis shows the mass fraction, and horizontal axis shows the grain diameter. For example, if 1mm grain diameter has 80% of cumulative mass fraction it means that 80% of this sample contains 1mm grain diameter or less than 1 mm (see the picture below).

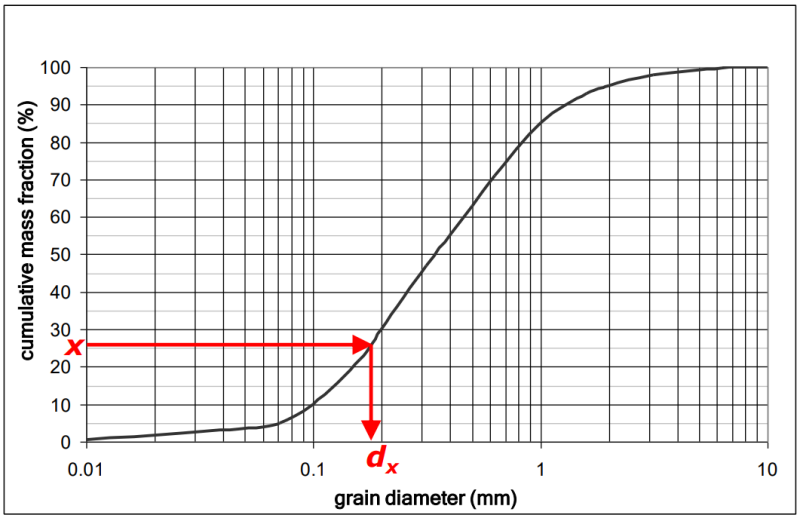
How to get granulometric curve? In order to perform sieve analysis we can use sieve machine. Sieve machine consist of sets of sieves from coarse sieve on top to fine sieve and a cup at the bottom. The mechanism is to shake the set. Finally, each sieve consists of grain sizes which are bigger than the sieve.

Sample consists of different grain size fractions

granulometric curve



dx and U: From the granulometric curve, several parameters can be determined in order to characterize the sample. d_x denotes the grain diameter for which x% (in mass or weight, not volume) of the sieve material is smaller than this diameter.



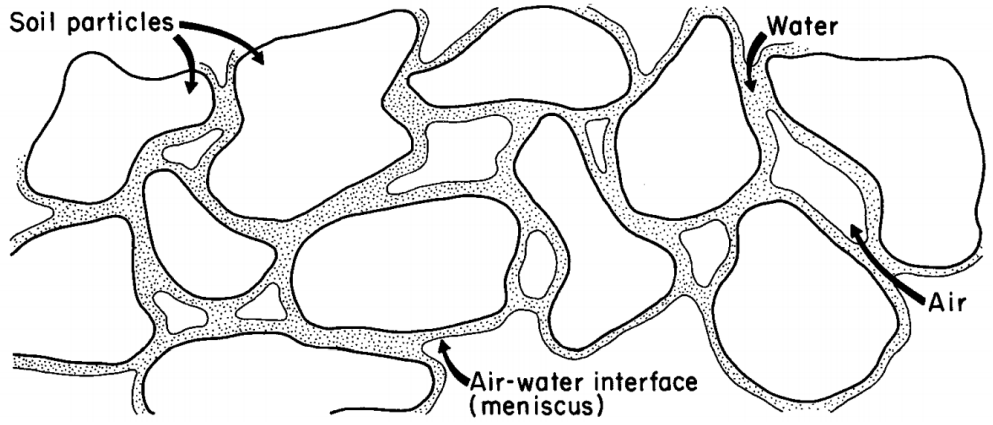
Grain diameters d_{10}, d_{60}, d_{75} are of practical importance with regard to groundwater flow properties. The ratio of d60 and d10 is called **coefficient of uniformity, U**:

$$U = \frac{d_{60}}{d_{10}}$$

d_{75} is specifically used for well construction purpose (not covered by this lecture)

2.3.10 Subterranean water

The subsurface can be regarded as a three-phase system consisting of a solid phase (soil particles), a water phase, and a gas phase. a schematic illustration for voids or pores in an unconsolidated porous medium is given in the figure below. Each phase has similar density and other properties. Sometimes it is possible for the fourth phase which is contamination.Voids are filled with water and gas. The volumetric ratio of water in voids can be calculated by water content.



2.3.11 Water content:

Water content is defined as the share of water in the porous medium:

$$\theta = \frac{V_w}{V_T}$$

- θ = water content
- V_w = water volume
- V_T = total volume

Water content cannot exceed the total porosity. i.e. n (total porosity is independent of the fluid content of porous medium).

2.3.12 Degree of saturation:

Another way to express the ratio of water in the porous medium is the degree of saturation, i.e. the ratio of water volume to void volume:

$$S = \frac{V_w}{V_v}$$

- S= degree of saturation
- V_w= water volume
- V_v= voids volume

The degree of saturation is equal to $\frac{}{n}$. S can vary between 0 to 1 (or between 0% to 100%), S=0 means no water in the voids, whereas S=100 means voids are completely filled with water.

Example Problem

The voids volume and the total air in the subsurface sample (0.2 m^3) was found to be 0.02 m^3 and 0.001 m^3, respectively. How much water does the sample contain and what is the degree of saturation?

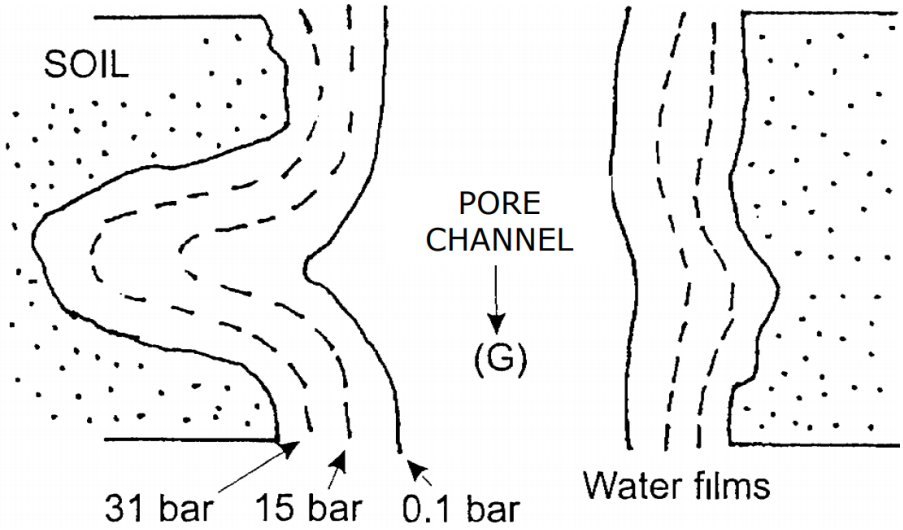
```
# solution

V_T = 0.2 # m^3, Total sample.
V_v = 0.06 # m^3, volume of voids
V_a = 0.004 # m^3, volume of air

# interim calculation
V_w = V_v - V_a # m^3, Vol. water,  the remaining volume

# calculation

Theta = V_w/V_T # -, water content
S = V_w/V_v # # -, degree of saturation

# print
print("The water content of the sample is: {0:.0%}".format(Theta),"\n")
print("The degree of saturation of the sample is: {0:.0%}".format(S))
```

```
The water content of the sample is: 28%

The degree of saturation of the sample is: 93%
```

2.3.13 Forces acting on subterranean water:

Subterranean water is subject to several forces. The most important ones are:

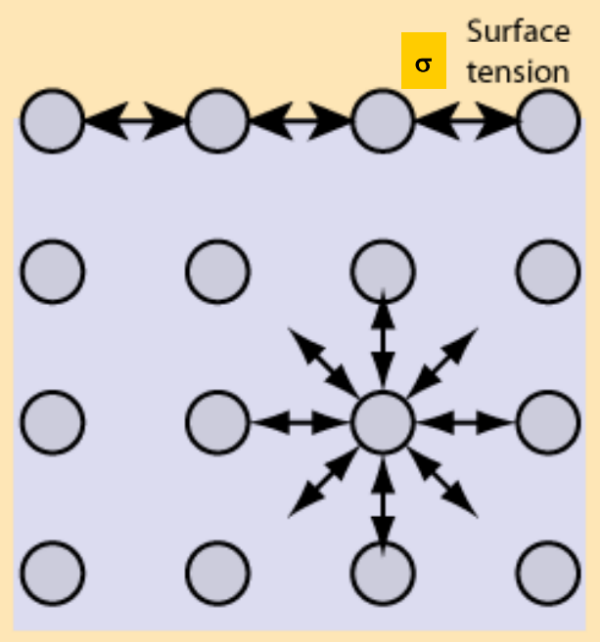
- gravity
- attractive forces between the water molecules (cohesion)
- attractive forces between water and solids (adhesion)



In the figure above, dotted area represent the solid phase. In the pore channel the dominant force is gravity, shown as G. getting closer to the solid surface, adhesive force become more important. The numbers indicate the required pressure to remove the corresponding layer of water from the solid surface. As an example, in order to remove the last layer of water from the solid surface, 31 bar pressure needs to be applied. Another easy way to remove the water is boiling the sample in the oven.

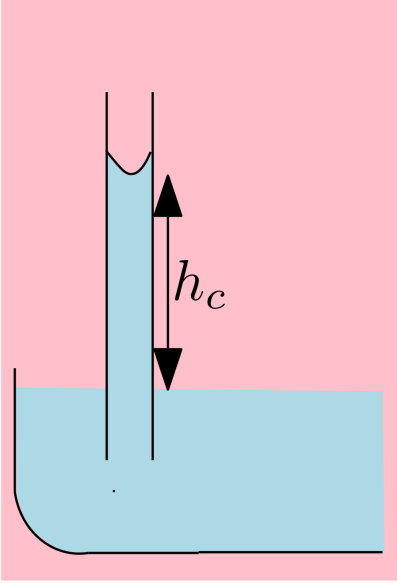
2.3.14 Surface tension:

Cohesive forces acting on water molecules compensate each other if the molecule is not located near water-air or water-solid interface. This is no longer true at an interface: cohesive interaction is reduced on one side. The resulting force tends to minimize the interface area. Macroscopically, this effect is parametrized by the “surface tension”, which is defined as the energy needed to increase the area of the interface by one unit.



Common units of the surface tension are $\frac{J}{m^2}$ or $\frac{N}{m}$ (Its dimension is $\frac{M}{T^2}$). The surface tension of water is about 7.5 . 10 -2 $\frac{N}{m}$ at 10 C.

2.3.15 Capillary action:



Water is subject to capillary action when adhesion is strongr than cohesion. The capillary rise of water in a tube depends on the surface tension and the tube redius. The maximum capillary rise is given by:

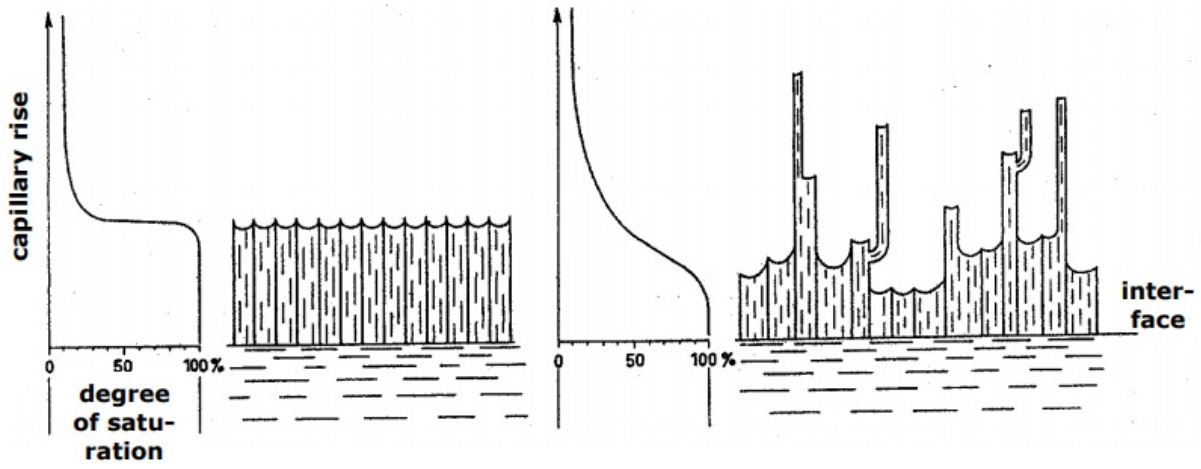
$$h_c = \frac{2\sigma w}{\rho_w g r}$$

- h_c= maximum capillary rise
- $sigmaw$= surface tension

- ρ_w= water density
- g= acceleration of gravity
- r= radius of the tube

2.3.16 Capillary action in the subsurface:

Capillary actions play a dominant role in the subsurface. The capillaries are given by individual pore channels. Poor channels in poorly sorted material may strongly differ in diameter, such that a certain variability in capillary rise is observed.



Left sketch shows the capillary rise in a perfectly sorted material which all the pores have the same size. So capillary rise is similar in every single pores. The right sketch, shows a real situation of subsurface. There are different grain size and then different pore channels, which **results** in various capillary rise.

Example Problem

For water at a tube with a radius *R*, the surface tension is 73 $\frac{g}{s^2}$, the density is 0.999 $\frac{g}{cm^3}$. Compute the rise of water in the capillary tube

```
# input data
sigma= 73 #g/s^2 surface tension
rho= 0.999 # g/cm^3 water density
g=980 #cm/s^2 acceleration of gravity

#calculation
h_c=(2*sigma)/(rho*g)

#output
print("The maximum water rise in this tube is: {0:0.2f} 1/R cm".format(h_c))
```

```
The maximum water rise in this tube is: 0.15 1/R cm
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import panel as pn
```

(continues on next page)

(continued from previous page)

```
pn.extension("katex", "mathjax")
```

2.4 Lecture 3- Groundwater as a reservoir

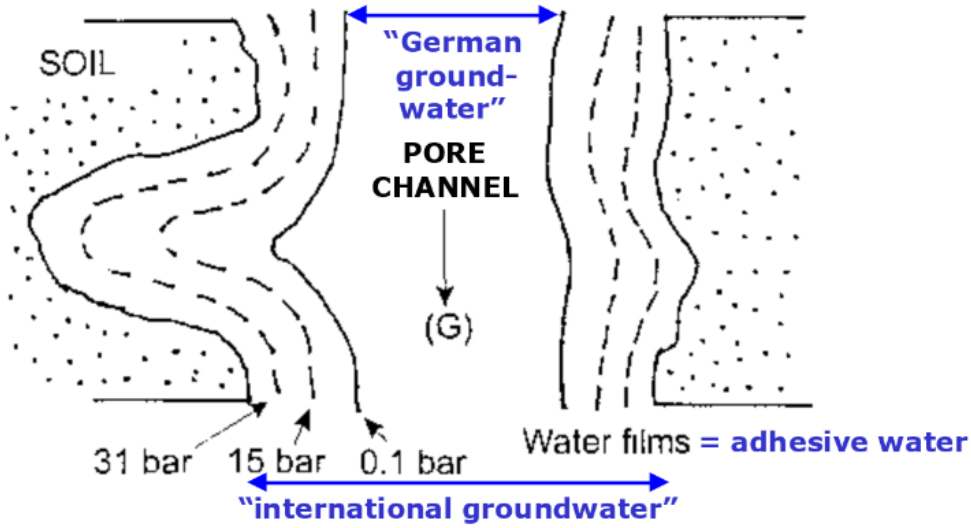
(The contents presented in this section were re-developed principally by Prof. Peter Dietrich and Dr. P. K. Yadav. The original contents are from Prof. Rudolf Liedl)

The content of the previous section was dedicated to very fundamental properties, such as aquifer and its types, solid and liquid (water) volumes in an aquifer, a of subsurface.

In this lecture, the subsurface will be considered from the perspective as a groundwater reservoir and some key definition and parameters will be introduced.

2.4.1 Groundwater and Aquifers

If the subterranean water completely fills the pore space we call it groundwater. In Germany a slightly different definition is in use. There, groundwater is only the subterranean water which is not subject to other forces than gravity (see Fig. below). That means, that the water adhesively bound to the grains is not part of the groundwater. Applicable forces in groundwater are sometime locally defined. For, e.g., In **Germany** the *gravity* is the only force acting on groundwater, whereas forces such adhesion, cohesion along with gravity are considered internationally.



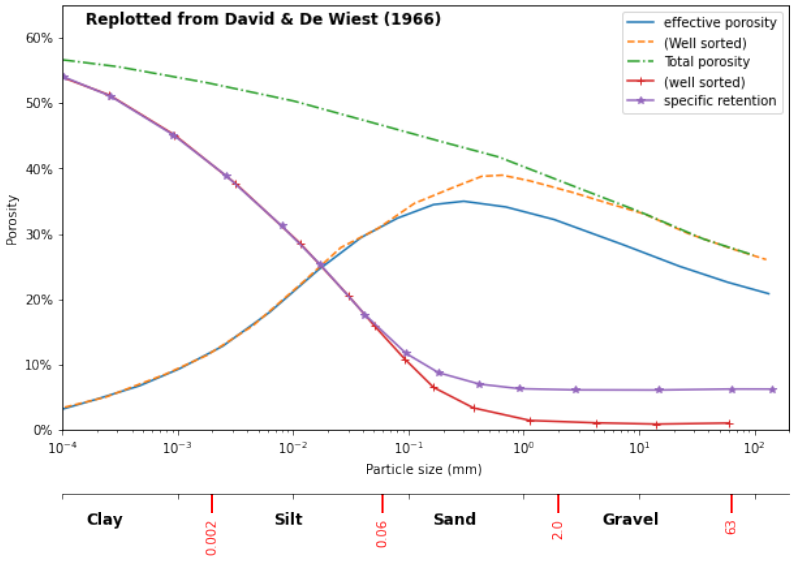
The difference between the international and the German definition of groundwater is the consideration of the adhesive water. Adhesive water does not participate in water movement. The same is true for water in isolated pores or in dead-end pores. All subterranean water not participating in water movement is summarized as immobile water. In contrast, the mobile water is the subterranean water participating in water movement.

```
Row
    [0] Column
        [0] Markdown(str, style={'text-align': 'justify'}, width=400)
        [1] LaTeX(str)
        [2] Markdown(str)
    [1] Spacer(width=50)
    [2] Video(str, height=400, sizing_mode='fixed', width=600)
```

The volumetric share of pore, which can be occupied by mobile water, is termed effective porosity n_e or flow-through porosity

$$n_e = \frac{V_{p,m}}{V_T}$$

The effective porosity is dimensionless and the pore volume $V_{p,m}$ available for mobile water as well as the total volume V_T has the dimension L^3. Effective porosity cannot exceed total porosity, i.e. $n_e \leq n$. The difference $(n \breve{} n_e)$ is termed **specific retention** or **field capacity**. Specific retention is the volumetric share of water which is retained in the porous medium after drainage due to gravitation. The reason for retention is the adhesive force which bounds water at the grain surfaces. Because the available grain surface in a medium depends on the grain size, the effective porosity is also different for various materials. As shown in Figure (below) clay has a high total porosity but only a low effective porosity whereas the total porosity of cobbles is not so significant different from the effective porosity of this material.



Example Problem

Moist sand specimen = 72.5 cm^3 and its weight = 152 g Oven dried sample = 71.2 cm^3 and its weight = 145 g

Other available information: Specific weight of particles (γ_s) = 2.65 g/cm^3 Specific weight of water (γ_w) = 1 g/cm^3

Find, total porosity, void ratio, water content, degree of saturation and effective porosity.

```
# solution

V_ms = 72.5 # cm^3, volume moist sand
W_ms = 152 # g, weight moist sand, also total volume
V_ds = 71.2 # cm^3, volume dry sand
W_ds = 145 # g, weight dey sand

# Other info
g_s = 2.65 # g/cm^3, sp. weight, particles
g_w = 1 # g/cm^3, sp. wt. water

# intermediate calculation
V_w = (W_ms-W_ds)/g_w # cm^3,  W_w/g_w; density = M/V
V_s = W_ds/g_s # cm^3,
```

(continues on next page)

(continued from previous page)

```
V_v = V_ms - V_s # cm^2, volume of voids
W_w = W_ms - W_ds # g, weight of water

# results calculation

n = V_v/V_ms*100 # %, Total porosity
e = V_v/V_s *100 # %,  void ratio
w = W_w/W_ds *100 # %, moisture content
S = V_w/V_v * 100 # %, degree of saturation

print("Total porosity is {0:0.2f}%".format(n))
print("Void ratio is {0:0.2f}%".format(e))
print("Moisture content is {0:0.2f}%".format(w))
print("Degree of saturation is {0:0.2f}%".format(S))
```

```
Total porosity is 24.53%
Void ratio is 32.50%
Moisture content is 4.83%
Degree of saturation is 39.36%
```

```
p5 = pn.pane.Markdown(""" ###Aquifer Classifications """)

p6 = pn.pane.Markdown("""
Subterranean formations can be classified by the capability to store and/or transmit␣
↪groundwater under natural conditions.

> An **aquifer** or a **groundwater reservoir** can store and transmit significant (=␣
↪exploitable) amounts of groundwater.

> An **aquitard** can store and transmit groundwater but to a much lesser extent than␣
↪an (adjacent) aquifer.

> An **aquiclude** can store groundwater but cannot transmit groundwater.

> An **aquifuge**can neither store nor transmit groundwater.

Aquifers usually appear as **layers**, i.e. their lateral extent is rather large as
compared to their vertical extent (maybe 10 - 100 km vs. 10 - 100 m). The upper␣
↪aquifer
boundary is called **aquifer top**, and the lower boundary is called **aquifer␣
↪bottom**.
The vertical distance between aquifer top and aquifer bottom is called **aquifer␣
↪thickness**.
Upper and lower aquifer boundaries do not have to be horizontal and the thickness may␣
↪be
spatially variable (see Fig. on the right).

 """, width=400, style={"text-align": "justify"})

p7 = pn.pane.Video("images/L03_f_6.mp4", width=600, height=200, loop=False)

p8 = pn.Column(p5, p6)
pn.Row(p8, pn.Spacer(width=25), p7)
```

```
Row
```

(continues on next page)

(continued from previous page)

```
    [0] Column
        [0] Markdown(str)
        [1] Markdown(str, style={'text-align': 'justify'}, width=400)
    [1] Spacer(width=25)
    [2] Video(str, height=200, sizing_mode='fixed', width=600)
```

```
p9 = pn.pane.Markdown(""" ### Unconfined Aquifer """)

p10 = pn.pane.Markdown(""" An aquifer can be completely or only partially filled with␣
↪groundwater (see Fig below).
Groundwater or an aquifer is termed **unconfined (phreatic)**, if the groundwater␣
↪does not
extend up to the aquifer top. The position of the groundwater table is therefore␣
↪changed
during water injection or extraction ("free" groundwater table). Water in a borehole␣
↪rises
up to the groundwater table.
""", width=600, style={"text-align": "justify"})

p11 = pn.pane.Video("images/L03_f_7.mp4", width=400, height=200, loop=False)
p12 = pn.pane.Markdown(""" ### Confined Aquifer """)

p13 = pn.pane.Markdown(""" Groundwater or an aquifer is termed **confined**, if the␣
↪aquifer
contains groundwater throughout its entire thickness. The pore space remains␣
↪completely
water filled during water injection or (moderate) extraction. This requires a low␣
↪permeable
cover layer. In addition, the groundwater _recharge area_ must be located at higher␣
↪altitude
than the aquifer top. The elevation of the _groundwater table_ in the recharge area␣
↪defines
the position of the confined aquifer's pressure line. Water in a borehole rises up to␣
↪the
pressure line (neglecting friction losses), i.e. higher than the elevation of the␣
↪aquifer top.
""", width=600, style={"text-align": "justify"})

p14 = pn.pane.Video("images/L03_f_8.mp4", width=400, height=200, loop=False)

p15 = pn.pane.Markdown(""" ### Artesian Aquifer """)
p16 = pn.pane.Markdown(""" If the pressure line is above ground surface the we have␣
↪an **artesian aquifer**
(see Fig. below). In this case, the water in a borehole would rises up to the ground
surface and then forms a fountain. _Artesian springs_ and _Artesian wells_ are based␣
↪on this principle.
The type of an aquifer can change along a cross section.""", width=600, style={"text-
↪align": "justify"})

p17 = pn.pane.Video("images/L03_f_9.mp4", width=400, height=200, loop=False)

pn.Column(p9, p10, p11, p12, p13, p14, p15, p16, p17)
```

```
Column
    [0] Markdown(str)
    [1] Markdown(str, style={'text-align': 'justify'}, width=600)
```

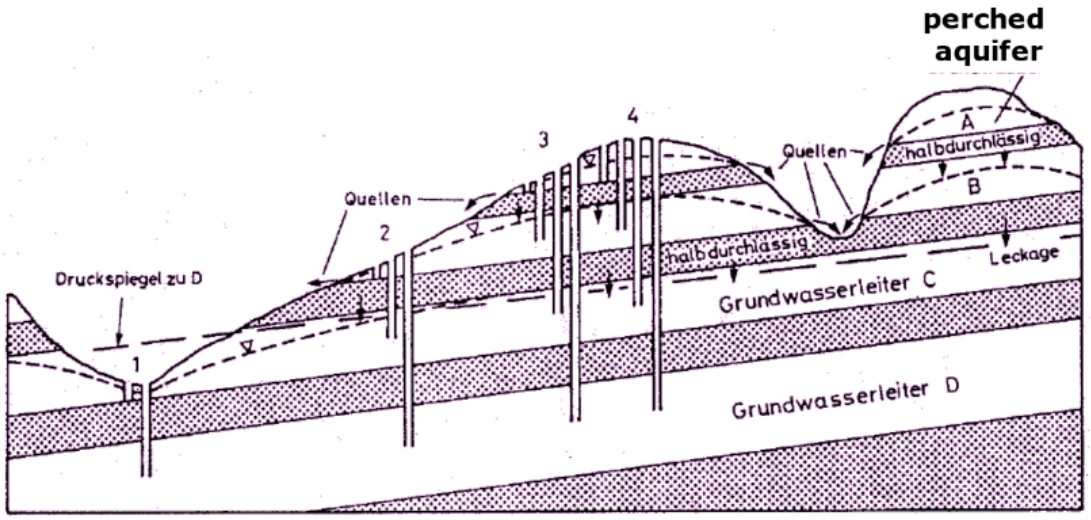
(continues on next page)

(continued from previous page)

```
    [2] Video(str, height=200, sizing_mode='fixed', width=400)
    [3] Markdown(str)
    [4] Markdown(str, style={'text-align': 'justify'}, width=600)
    [5] Video(str, height=200, sizing_mode='fixed', width=400)
    [6] Markdown(str)
    [7] Markdown(str, style={'text-align': 'justify'}, width=600)
    [8] Video(str, height=200, sizing_mode='fixed', width=400)
```

Example

Aquifer	Obs. Point 1	Obs. Point 2	Obs. Point 3	Obs. Point 4
A	—	—	Unconfined	Unconfined
B	—	Unconfined	Unconfined	Confined
C	Unconfined	Unconfined	Confined	Confined
D	Conf./Artesian	Confined	Confined	Confined



**perched aquifer*: Unconfined aquifer on top of another unconfined aquifer, separated from each other by a shallow aquitard

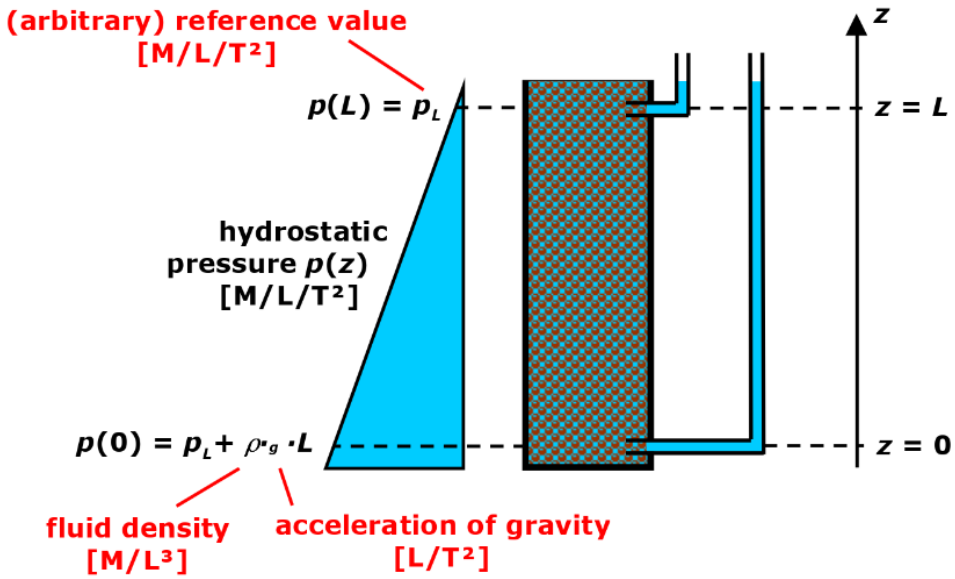
2.4.2 Pressure and pressure head

A reason for the movement of groundwater can be (hydrostatic) pressure difference. Let us consider a vertical column containing a porous medium and water filling the voids completely (Fig. below). We can assign to the top an arbitrary reference value pL for the pressure. Due to the load of the overlaying water column, the pressure increase if we go deeper into the column. This is the same as what we observe if we are diving in a lake. The increase of the pressure depends on the density of the fluid and the depth below the water surface. In the setup given in Fig., we have the (hydrostatic) pressure p [M/L/T^2] as a function of the height z [L] above the reference point

$$p(z) = p_L + \rho \cdot g \cdot (L - z)$$

An optional title

with, ρ [M/L^2] as the fluid density, g the acceleration of gravity [L/T^2], and L [L] the length of the water column



As shown in Fig., we can add two observation points, one at the bottom ($z = 0$) and the other at the top of the column ($z = L$). The pressure difference Δp between the observation points is

$$\Delta p = p(L) - p(0) = p_L - (p_L + \rho \cdot g \cdot L) = -\rho \cdot g \cdot L$$

Example: Compare the pressure difference for an experimental setup (pipe length 50 cm) in which water and diesel are the two liquids.

```
# solution
L_p = 50 # cm length of pipe
g = 981 # cm/s^2, accl. due to gravity


# Assume densities
rho_w = 1.0 # g/cm^3, density of water
rho_d = 0.830 # g/cm^3, density of diesel

# calculate
Dp_w = rho_w*g*L_p # g/cm.s^2, pressure difference due to water
Dp_d = rho_d*g*L_p# g/cm.s^2, pressure difference due to water
Dp_w
print("The pressure difference due to water is {0:2.2f} g/cm.s\u00b2,".format(Dp_w),
      "and that due to diesel is {0:0.2f} g/cm.s\u00b2.".format(Dp_d), )
```

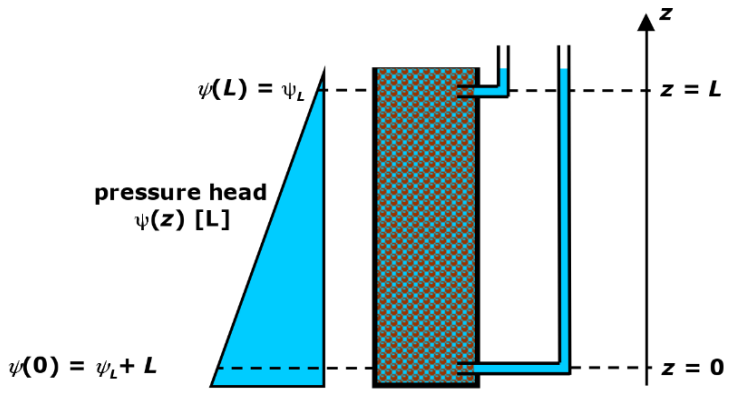
```
The pressure difference due to water is 49050.00 g/cm.s², and that due to diesel is␣
↪40711.50 g/cm.s².
```

2.4.3 Hydrostatic pressure

Mostly, pressure head is used instead of pressure when dealing with hydraulic properties or phenomena of the subsurface. The reason is that the pressure head can be easily measured with a tape whereas for a pressure measurement a more expensive manometer is necessary. The (hydrostatic) **pressure head** ψ [L] is defined as

$$\psi(z) = \psi_L + L - z$$

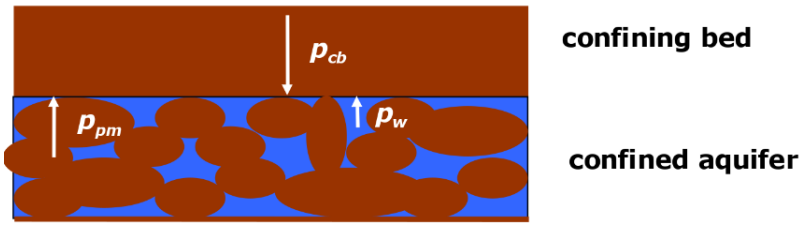
This expression makes it clear why we can measure the pressure head with a tape. Similar to pressure head, the hydrostatic pressure can be schematically represented as



2.4.4 Pressure in a Confined Aquifer

Let us now consider a *confined aquifer* (see Fig. below). The confining bed exerts a certain pressure p_{cb} on the aquifer. This pressure is compensated partly by the porous medium and partly by the groundwater (pressures p_{pm} and p_w, respectively). Therefore, we can write the following equation

$$p_{cb} + p_{pm} + p_w = 0$$



We can induce a *change in the hydrostatic pressure* Δp_w with an injection or release of groundwater. According to the above equation we can formulate

$$\Delta p_{cb} + \Delta p_{pm} + \Delta p_w = 0.$$

The hydrostatic pressure changes do not affect the weight of the confining bed and the exerted downward pressure remains unchanged. Therefore we have

$$\Delta p_{cb} = 0 \quad \text{and}$$

$$\Delta p_{pm} = -\Delta p_w$$

This implies that an increase of hydrostatic pressure automatically results in a decrease in the pressure exerted by the porous medium. In the case of a decrease of hydrostatic the pressure exerted by the porous medium would increase. The change in hydraulic pressure will have two effects with regard to water volume. First, the hydraulic

pressure change Δp_w directly leads to expansion/compression of the water and the water volume is accordingly increased/decreased. Secondly, the opposite change $\Delta p_{pm} = -\Delta p_w$ leads to compression/expansion of the porous medium as a whole (not the individual grains!). This, in turn, results in a reduced/an enlarged pore space such that the stored water volume is decreased/ increased. Both effects contribute to aquifer storage properties (see next section).

2.4.5 Aquifer storage properties

Storage properties of the aquifer and associated parameters can be understood by considering pressure changes. For this purpose, we consider the effect of a *change in water volume* $\Delta V_w'$ due to a *change in hydrostatic pressure*. The *relative* changes in water volume $\Delta V_w'/\Delta w$ [-] are proportional to change of pressure in groundwater Δp_w:

$$\frac{\Delta V_w'}{V_w} = \alpha_w \cdot \Delta p_w$$

with α_w as the **compressibility of water** [LT^2/M]. The compressibility of water is roughly $4.4 \cdot 10^{-10}$ m^2/N. Taking into account an incompressible behavior of the water, that means an increase in hydrostatic pressure results in an inflow of water. A decrease in hydrostatic pressure results would cause an outflow of water. The above equation can be rearranged to yield

$$\Delta V_w' = \alpha_w V_w \Delta p_w = \eta \alpha_w V_T \Delta p_w = \eta \alpha_w V_T \rho_w g \Delta \psi$$

With η [-] as the *total porosity* and $\Delta\psi$ [L] as the *change in pressure head.*

2.4.6 Change in total volume

The preceding considerations dealt with a change in storage by inflow or outflow. The change was invoked by a change of pressure in groundwater Δp_w. But a change could be also invoked by the change of the pressure exerted by the porous medium on the confining layer Δp_{pm}. A change Δp_{pm} in the pressure results in a decrease or an increase ΔV_T in total aquifer volume. Both quantities are proportional to each other via

$$\frac{\Delta V_T}{V_T} = -\alpha_{pm} \Delta p_{pm}$$

whereby the ratio is the relative change of the total volume and α_{pm} the compressibility of the porous medium [LT^2/M]. The compressibility of the porous medium is roughly $10^{-10} - 10^{-8}$ m^2/N for gravel, $10^{-9} - 10^{-7}$ m^2/N for sand, and $10^{-8} - 10^{-6}$ m^2/N for clay. Taking into account the relation between pressure and pressure head, the above equation can be rearranged to yield

$$\Delta V_T = -\alpha_{pm} V_T \Delta p_{pm} = \alpha_{pm} V_T \Delta p_{pw} = \alpha_{pm} V_T \rho_w g \Delta \psi$$

ΔV_T represents a *change in volume of the porous medium* as a whole. It is composed of a change in volume ΔV_s of the solids and another change $\Delta V_w''$ in water volume. Because the change in volume of the solid is negligible, we can write

$$\Delta V_T = \Delta V_s + \Delta V_w'' \approx \Delta V_w''$$

If we compare the last two equations we can immediately derive

$$\Delta V_w'' = \alpha_{pm} V_T \rho_w g \Delta \psi$$

With words that means a *decrease of pressure* in the porous medium leads to *an expansion of the porous medium* and an associated *increase in water volume* and enlarged pore space. An *increase in pressure* in the porous medium would lead to a *compression* of the porous medium and an associated *decrease* in water volume and *reduced* pore space.

The *total* change ΔV_w in water volume consists of both effects caused by pressure changes Δp_{pm} and Δp_w. Therefore we have

$$\Delta V_w = \Delta V_w' + \Delta V_w''$$

Using the results derived before, we can express how ΔV_w depends on changes $\Delta\psi$ in pressure head

$$\Delta V_w = \Delta V_w' + \Delta V_w'' = \eta\alpha_w V_T \rho_w g \Delta\psi + \alpha_{pm} V_T \rho_w g \Delta\psi$$

The *first term* of the sum is related to changes in hydrostatic pressure (Δp_w) and the *second term* to pressure changes in the porous medium (Δp_{pm}).

Example: The 45 m thick aquifer under the change of pressure 245 KPa compacts 0.20 m. What is the compressibility of porous media.

```
# Available information

dP = 245 # KPa= KN/m^2, Change pressure
m = 45 # m, aquifer thickness
dm = 0.20 # m, change in aquifer thickness

print("V = A*m and dV = A * dm \n")

# Calculate
al_pm = (dm/m)/dP # m^2/KN, compressibility of porous media.

print("The compressibility of porous media in aquifer {0:0.2e} m\u00b2/KN.".format(al_
↪pm))
```

```
V = A*m and dV = A * dm

The compressibility of porous media in aquifer 1.81e-05 m²/KN.
```

2.4.7 Specific storage

For the characterization of the storage properties of an aquifer, we use the term **specific storage** S_s. It is defined as the volume of water that is released from a unit aquifer volume if hydrostatic pressure head is reduced by one unit

$$S_s = \frac{\Delta V_w}{V_T \cdot \Delta\psi}$$

The dimension of *specific storage* is 1/L. Both impacts on water volume discussed before have to be considered in order to quantify ΔV_w in the above equation

$$\Delta V_w = \eta\alpha_w V_T \rho_w g \Delta\psi + \alpha_{pm} V_T \rho_w g \Delta\psi$$

The specific storage can therefore also be expressed as

$$S_s = (\eta\alpha_w + \alpha_{pm})\rho_w g$$

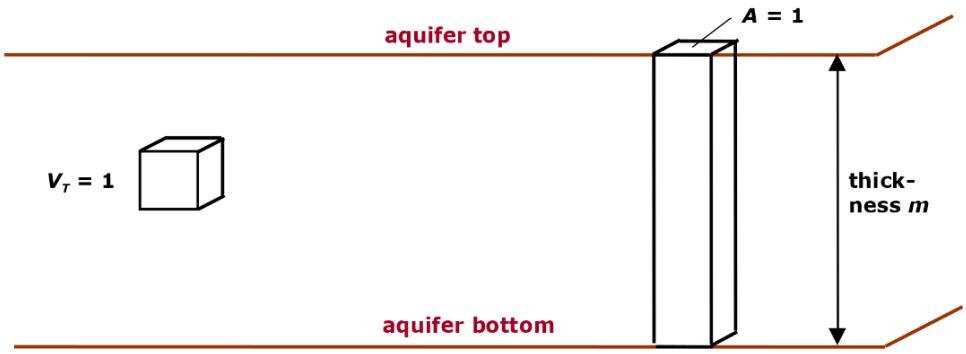
Typical values for specific storage range from 10^{-6} 1/m (e.g. gravel) to 10^{-2} 1/m (e.g. clay).

2.4.8 Storativity

Due to their relatively large lateral extent, aquifers are mostly considered as spatially two-dimensional (2D) systems. In this case, specific storage S_s is replaced by the **storativity** or **storage coefficient** S (Fig. below). Reference volume in a confined aquifer for defining specific storage S_s is a unit cube (e.g. $V_T = 1$ m^3), and for defining storativity S is a cuboid extending from the aquifer bottom to the aquifer top over a unit area (e.g. $A = 1$ m^2 and $V_T = A \cdot m$).

For confined aquifers S is simply obtained by multiplying S_s by the aquifer thickness m

$$S = S_s \cdot m$$



Storativity can be interpreted as the volume of water released from an aquifer volume extending from the aquifer bottom up to the aquifer top over a unit area if the hydrostatic pressure is reduced by one unit. *Storativity* is dimensionless.

Actually, **unconfined** aquifers are always treated as 2D systems. As a consequence, storativity is used to quantify water storage properties. The definition of storativity remains unchanged in principle but the considered aquifer volume now extends from the aquifer bottom up to the water table. For *unconfined aquifers*, storativity values correspond to *effective porosities*. This is explained by the free groundwater table. In this case a pressure changes simply lead to filling or emptying of voids. This is fundamentally different from the storage properties of confined aquifers discussed before.

In **confined** aquifers all voids remain filled with groundwater during pressure changes and storage properties depend on the compressibilities of water and the porous medium.

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

2.5 Lecture 4 - Darcy's Law and Conductivity

(The contents presented in this section were re-developed principally by Prof. B. R. Chahar and Dr. P. K. Yadav. The original contents are from Prof. Rudolf Liedl)

2.5.1 Energy and hydraulic head

In the last section we learned that *hydrostatic pressure difference* $p(z)$ will not allow the fully quantify water flow. In fact in addition to $p(z)$ other form of energy must also be considered.

The energy available for groundwater flow is given the name *hydraulic head* (h) or also called *piezometric head*:. It consists of three components, related to

elevation, pressure and velocity.

The total energy head is expressed by the equation

$$h = z + \frac{p}{\rho g} + \frac{v^2}{2g}$$

where, z is the *elevation* or *datum head* [L], p is the *pressure* exerted by water column [M L^{-1} T^{-2}], ρ is the density of fluid [M L^{-3}], g is the acceleration due to gravity [LT^{-2}], and v is velocity of flow [LT^{-1}].

Note that the h has the dimension of length [L]. In groundwater flow, the velocity is so low that the energy contained in velocity can be neglected when computing the total energy. Thus, the hydraulic head (see figure below) is written as

$$h = z + \frac{p}{\rho g}$$

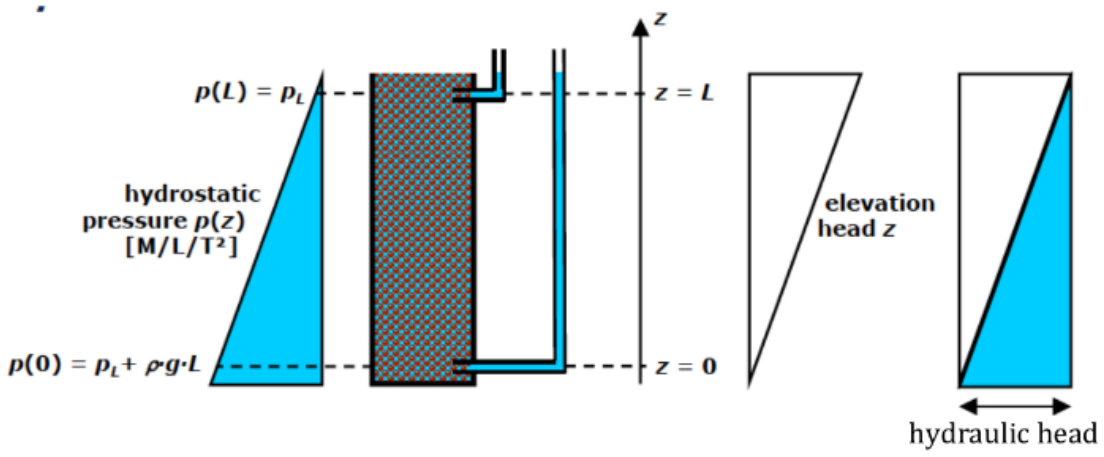
The above equation says that Water flow is governed by differences in hydraulic head and not by differences in pressure head alone.

It is to be noted that z depends on the orientation. In the above equation $+z$ is considered oriented upward (based on conventional sign convention). If $z-$axis is oriented downwards, we have

$$h = \frac{p}{\rho g} - z$$

Hydraulic head and discharge - when there is no discharge

Consider the figure below:

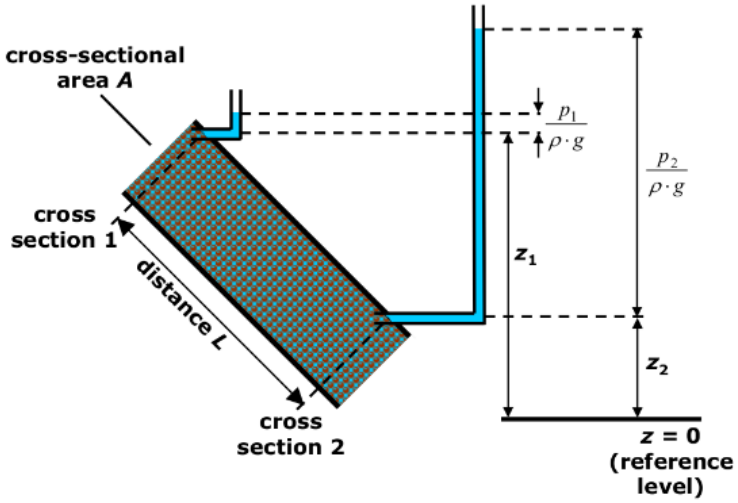


The pressure head: $p(z) = p_L + \rho \cdot g \cdot (L - z)$ The hydraulic head (piezometric head): $h(z) = \frac{p(z)}{\rho \cdot g} + z = \frac{p_L + \rho \cdot g \cdot (L-z)}{\rho \cdot g} + z = \frac{p_L}{\rho \cdot g} + L = \text{Const}$

In the figure above the hydraulic head difference between two points ($z = 0$ and $z = L$) are exactly equal, i.e., $\Delta h = 0$. This refers to the system with no energy gradient and hence a *no flow* system.

Hydraulic head and discharge - when there will be a discharge

Now consider the figure below



Here there is clear difference between the elevation head (z_1 and z_2), which is taken from a reference level ($z = 0$ in this case. Average Sea Level (ASL), is often use for this reference). Also differing are pressure heads (p_1 and p_2). Therefore, the $h(z)$ in this case are:

$$h_1 = \frac{p_1}{\rho \cdot g} + z_1 \tag{2.1}$$

$$h_2 = \frac{p_2}{\rho \cdot g} + z_2 \tag{2.2}$$

As can be observed from the figure, in this case $h_1 < h_2$. Thus, analogus to flow of energy - i.e., from higher energy level to the lower energy level, the flow in this example case is from cross-section 2 towards cross section 1.

It is to be noted that the *magnitude* of the hydraulic head (h) rather than the *orientation* of the set-up determines the direction of discharge.

Also, important to note is that the *differences of hydraulic head* (Δh) is independent of of the position of the origin of the reference axis (z-axis in the above case).

Example problem

Energy and hydraulic head

At a place where the fluid pressure is $1500 \frac{N}{m^2}$, the distance above the reference elevation is $0,8m$ and the fluid density is $1000 \frac{kg}{m^3}$. The fluid moves at a speed of $1 \cdot 10^{-6} \frac{m}{s}$. Find the total energy head.

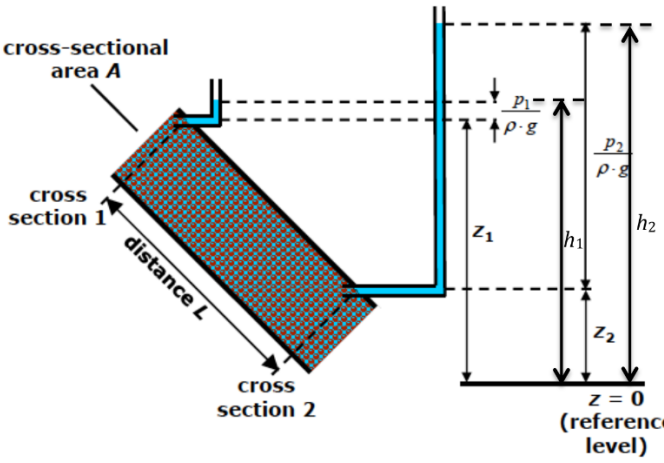
```
print("Let us find the total energy head.\n\nProvided are:")

z = 0.8 #elevation
g = 9.81 # [m/s^2]
p = 1500 # fluid pressure [N/m^2]
rho = 1000 # fluid density [kg/m^3]
v = 1e-6 #velocity [m/s]

#solution
h = z + p/(rho*g)+ v**2/(2*g) # m, head

print("elevation = {} m\nacceleration due to gravity = {} m\nfluid pressure = {} m\
↪nfluid density = {} m, and\nvelocity = {} m/s".format(z,g,p,rho,v),"\n")
print("The resulting total energy head is {0:0.2f} m".format(h))
```

```
Let us find the total energy head.

Provided are:
elevation = 0.8 m
acceleration due to gravity = 9.81 m
fluid pressure = 1500 m
fluid density = 1000 m, and
velocity = 1e-06 m/s

The resulting total energy head is 0.95 m
```

2.5.2 Darcy's Law

Groundwater in its natural state is invariably moving. This movement is governed by established hydraulic principles. Darcy's Law is a phenomenologically derived constitutive equation that describes the flow of a fluid through a porous medium. Darcy's Law along with the equation of conservation of mass is equivalent to the groundwater flow equation, one of the basic relationships of hydrogeology. It is also used to describe oil, water, and gas flows through petroleum reservoirs.

The law was formulated by Henry Darcy (Darcy, 1856) based on the results of experiments on the flow of water through beds of sand (Figure below shows a schematic setup). In the experiments, *area of cross section* (A [L^2] = Const) was kept constant. Constant discharge (Q = Const) was applied and the sand medium was fully saturated, i.e., voids between sand grains were completely filled with water.



Darcy's law is a simple mathematical statement which neatly summarizes several familiar properties that groundwater flowing in aquifers exhibits, including:

1. if there is no hydraulic gradient (difference in hydraulic head over a distance), no flow occurs (this is hydrostatic conditions),
1. if there is a hydraulic gradient, flow will occur from a high head towards a low head (opposite the direction of increasing gradient, hence the negative sign in Darcy's law),
1. the greater the hydraulic gradient (through the same aquifer material), the greater the discharge, and
1. the discharge may be different through different aquifer materials (or even through the same material, in a different direction) even if the same hydraulic gradient exists.

From the experiments Darcy observed that the *volume of water per unit time* passing through a porous medium

- is *directly proportional* to the A [L^2] and the head difference between inlet and outlet $(h_1 \text{˘} h_2)$ [L], and
- is inversely proportional to the *length of the medium* L [L]

i.e.,

$$\frac{\text{Vol}}{t} = Q \propto A(h_1 - h_2)\frac{1}{L}$$

which in terms of specific discharge q, or discharge velocity or Darcy velocity v [LT^{-1}] is

$$q = v = \frac{Q}{A} = -K\frac{\partial h}{\partial L} = -K\,i$$

where constant of proportionality K = *hydraulic conductivity* [LT^{-1}]; and $i = \partial h/\partial L$ = *hydraulic gradient* = rate of head loss per unit length of medium [ ]. The *negative sign* indicates that the total head is decreasing in the direction of flow because of friction or resistance

Example problem

Darcy's Law

Calculate the specific discharge and the flow rate passing through the surface with the given parameters.

```
print("\nProvided are:\n")

K = 5e-4 # m/s, conductivity
A = 10 # m^2, surface
h_in = 10 # m, hydraulic head at the inlet
h_out = 2 # m, hydraulic head at the outlet
L = 5 #m, lenght of the column

#intermediate calculation
I = (h_in-h_out)/L

#solution
q = K*I
Q = K*I*A

print("Conductivity = {} m/s\nSurface = {} m^2\nHydraulic head at the inlet = {} m\
↪\nHydraulic head at the outlet = {} m\nLenght of the column = {} m".format(K, A, h_
↪in, h_out, L), "\n")
print("Solution:\nThe resulting specific discharge is {0:0.0e} m/s".format(q), "\nand␣
↪the flow rate is {0:0.0e} m^3/s".format(Q))
```

```
Provided are:

Conductivity = 0.0005 m/s
Surface = 10 m^2
Hydraulic head at the inlet = 10 m
Hydraulic head at the outlet = 2 m
Lenght of the column = 5 m

Solution:
The resulting specific discharge is 8e-04 m/s
and the flow rate is 8e-03 m^3/s
```

Darcy's law and analogous physical systems

Darcy's law is analogous to pipe flow in which energy is dissipated over the distance to overcome frictional loss resulting from fluid viscosity.

It also forms the scientific basis of permeability used in the earth sciences.

It may be noted Darcy's law is analogous to Fourier's law in the field of heat conduction, Ohm's law in the field of electrical networks, or Fick's law in diffusion theory.

2.5.3 Hydraulic Conductivity and Intrinsic Permeability

Hydraulic Conductivity (K) appeared in the Darcy's law as a constant of proportionality, i.e., it is the fundamental quantity that is required to describe groundwater flow. Therefore we illustrate this further,

A medium has a *unit hydraulic conductivity* if it will transmit in *unit time* a *unit volume of groundwater* at the prevailing kinematic viscosity through a cross section of *unit area* measured at *right angles* to the direction of flow, under a *unit hydraulic gradient*.

The hydraulic conductivity of a soil or rock depends on a variety of physical factors, important ones are:

- porosity,
- particle size and distribution,
- shape of particles,
- arrangement of particles.

Also, hydraulic conductivity is depends on the property of the fluid e.g., density, viscosity

In general for unconsolidated porous media,K varies with *square* of particle size; clayey materials exhibit low values of K, whereas sands and gravels display high values

Typical values for hydraulic conductivity (see figure XX for more comprehensive listing):

Media Type	hydraulic conductivity (m/s)
Gravel	10^{-2}˘10^{-1}
coarse sand	$\approx 10^{-3}$
medium sand	10^{-4}˘10^{-3}
fine sand	10^{-5}˘10^{-4}
Silt	10^{-9}˘10^{-6}
Clay	$< 10^{-9}$

Obtaining Hydraulic Conductivities

The hydraulic conductivity depends on properties of the fluid (density, viscosity, temperature) and on properties of the porous medium (effective porosity, grain size distribution). It can be obtained by calculation from formulas, laboratory methods, or field tests

The laboratory method can be indirect method or direct method. For example, determination of the hydraulic conductivity based on the evaluation of sieve analysis data is the indirect laboratory method. On the other hand, the direct method of determination of the hydraulic conductivity (e.g. permeameter) is based on some version of Darcy's experiment. Advantages of laboratory methods include controlled conditions, small sample size (easy handling), lower costs, larger number of experiments, etc. While, the disadvantages of laboratory methods are disturbed samples, additional pathways at column walls, small sample size (randomly high or low K), flushing of fine material, etc.

Hydraulic Conductivities estimations from Sieve analysis

Sieve analysis data can be evaluated to estimate hydraulic conductivity of unconsolidated media. There are several *empirical methods*. The simplest one dates back to Hazen (1892):

$$K = 0.0116 \cdot d_{10}^2$$

where, d_{10} = grain diameter (mm) corresponding to 10% of cumulative mass fraction. It can be generalized by including temperature (θ in °C). Thus the Hazen formula becomes

$$K = 0.0116 \cdot d_{10}^2 \cdot (0.7 + 0.03 \cdot \theta)$$

Hazen's formula is only valid for the indicated units, i.e., conversion of the *unit* may be required before using the formula.

Example problem

Hydraulic Conductivity from sieve data

An Aquifer with fine to medium sand was investigated with an sieve analysis. At a temperature of $20C$ a d_{10} of 0.13 mm was measured. Determine the hydraulic conductivity (using Hazen's formula) and how it changes when the temperature rises by $5C$.

```
print("Let us find the hydraulic conductivities.\n\nProvided are:")

d10 = 0.13 # mm,  grain diameter corresponding to 10% of cumulative mass fraction
T1 = 10 # °C, Temperature
deltaT = 5 # °C, temperature change

#intermediate calculation
T2 = T1 + deltaT

#solution based on Hazen's formula
K1 = 0.0116 * d10**2 * (0.7 + 0.03*T1)
K2 = 0.0116 * d10**2 * (0.7 + 0.03*T2)

print("grain diameter corresponding to 10% of cumulative mass fraction = {} mm\
 ↪\ntemperature = {} °C\ntemperature change = {} K".format(d10, T1, deltaT),"\n")
print("The resulting hydraulic Conductivity at 20°C is {0:0.2e} m/s".format(K1),
      "\nand the resulting hydraulic conductivity at 25°C is {0:0.2e} m/s".format(K2))
```

```
Let us find the hydraulic conductivities.

Provided are:
grain diameter corresponding to 10% of cumulative mass fraction = 0.13 mm
Temperature = 10 °C
temperature change = 5 K

The resulting hydraulic Conductivity at 20°C is 1.96e-04 m/s
and the resulting hydraulic conductivity at 25°C is 2.25e-04 m/s
```

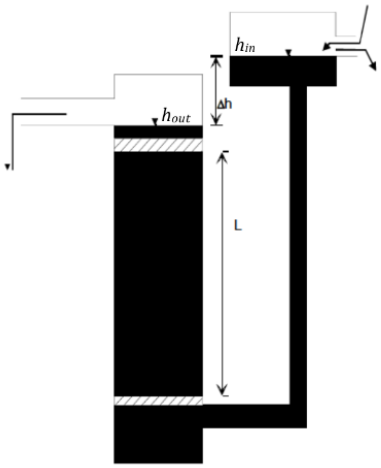
Hydraulic Conductivities estimations from Darcy's Law

Permeameter is an instrument used to determine hydraulic conductivity of soil samples in the laboratory as *direct method*. The design of permeameters is based on Darcy's experiment.

There are mainly two types of permeameters

1. Constant-head permeameter, and
2. Falling-head permeameter.

In **constant-head permeameters** as shown in Figure the hydraulic heads at inflow and outflow of the Darcy column are constant in time. As a consequence, the discharge is not changing with time.



The hydraulic conductivity can be obtained by observing discharge and heads and then substituting in the below formula that is rearranged form of Darcy's law from:

$$K = \frac{QL}{A(h_{in} - h_{out}}$$

where Q =discharge[L^3T^{-1}]; L = length of sample [L]; A = cross-sectional area of sample [L^2]; h_{in} = hydraulic head at column inlet [L]; h_{out} = hydraulic head at column outlet [L]; $\Delta h = h_{in} - h_{out}$

h_{out} can be set equal to zero as only head differences are important.

Example problem

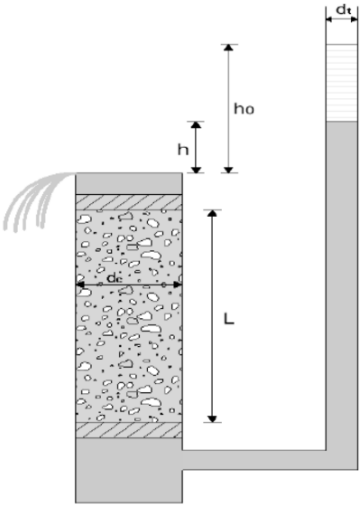
Hydraulic Conductivity from Constant head-permeameter

A constant-head permeameter has a length of 15 cm and a cross-sectional area of 25 cm^2. With a head of 5 cm, a total Volume of 100 mL of water is collected in 12 min. Determine the hydraulic conductivity.

```
print("Let us find the hydraulic conductivity with a constant-head permeameter.\n\
↪nProvided are:")

L = 15 # Length of the permeameter [cm]
A = 25 # cross-sectional area [cm^2]
h = 5 # hydraulic head [cm]
V = 100 # Volume of collected water [mL = cm^3]
t = 12 # time [min]

#solution
K1 = (V * L)/(A * t * h)
K2 = K1/(60*100)

print("Length of the permeameter = {} cm \ncross-sectional area = {} cm\u00b2\
↪nhydraulic head = {} cm \nVolume = {} mL \ntime = {} min".format(L,A,h,V,t),"\n")
print("The resulting hydraulic Conductivity is {0:2.0e} cm/min".format(K1),
      "\nand which is {:02.0e} m/s".format(K2))
```

```
Let us find the hydraulic conductivity with a constant-head permeameter.

Provided are:
Length of the permeameter = 15 cm
cross-sectional area = 25 cm²
hydraulic head = 5 cm
Volume = 100 mL
time = 12 min

The resulting hydraulic Conductivity is 1e+00 cm/min
and which is 2e-04 m/s
```

In falling-head permeameter as shown in Figure the hydraulic head at the outflow of the Darcy column is not changing, but the hydraulic head at the inflow is decreasing with time.



As a result, the discharge also decreases with time. Rewriting Darcy's law for a small time interval

$$K\frac{\pi d_c^2}{4}\left(h_{in}-h_{out}\right)\frac{1}{L}\mathrm{d}t=\frac{\pi d_t^2}{4}\mathrm{d}h$$

K can be obtained from the above equation by separating the variables and then integrating. The resulting expression for K will be

$$K=\frac{d_t^2 L}{d_c^2}\ln\left(\frac{h_{in}(0)-h_{out}}{h_{in}(t)-h_{out}}\right)$$

where, L = length of sample [L]; d_c = diameter of sample cylinder [L]; d_t = diameter of piezometer tube [L]; $h_{in}(0)$ = initial hydraulic head at column inlet [L]; $h_{in}(t)$ = final hydraulic head at column inlet [L]; h_{out} = hydraulic head at column outlet [L]; $h_0 = h_{in}(0) - h_{out}$; $h = h_{in}(t) - h_{out}$

and h_{out} can be set equal to zero as only head differences are important. The hydraulic conductivity can be obtained from the above equation using observed time and corresponding heads. Larger experimental time periods are needed for the falling-head permeameter, in particular if hydraulic conductivity is low. On the other hand, no measurement of discharge or water volume is required.

Field methods of determination of the hydraulic conductivity include tracer tests, auger hole tests, pumping tests of wells, etc. Field experiments are much more complicated and expensive than laboratory tests. Resulting hydraulic conductivities represent averages over an aquifer volume, which is covered by the experiment. The size of this volume depends on subsurface properties and on the experimental method used.

Intrinsic Permeability

A convenient alternative is to write Darcy's equation in a form of **intrinsic permeability** where the properties of the medium and the fluid are represented explicitly

$$v=\frac{-k\cdot\rho\cdot g}{\mu}\frac{\partial h}{\partial L}$$

where, k is the intrinsic permeability [L^2], and η is the dynamic viscosity of fluid [$ML^{-1}T^{-1}$] e.g., (Pa-S). The relation between *hydraulic conductivity intrinsic permeability*, therefore, is

$$K=k\cdot\frac{\rho\cdot g}{\eta}$$

In terms of Kinematic viscosity [L^2T^{-1}}], $\eta = \rho \cdot \nu$, the above relation becomes

$$K = k \cdot \frac{g}{\nu}$$

Both density and viscosity are temperature dependent quantites. Their values in field conditions $\approx 10°C$ and in the laboratory conditions $\approx 20°C$ are:

	10℃	20℃
density (kg/m^3)	999.7	999.7
kinematic viscosity (m^2/s)	$1.3101 \cdot 10^{-6}$	$1.0105 \cdot 100^{-6}$
ynamic viscosity (Pa·s)	$1.3097 \cdot 10^{-3}$	$1.3097 \cdot 100^{-3}$

The *intrinsic permeability* can be written in terms of specific weight or weight density [$ML^{-2}T^{-2}$] (or in metric unit-N/m^3), $\gamma = \rho \cdot g$ as

$$k = \frac{\eta}{\gamma} K$$

Example problem

Instrinsic Permeability

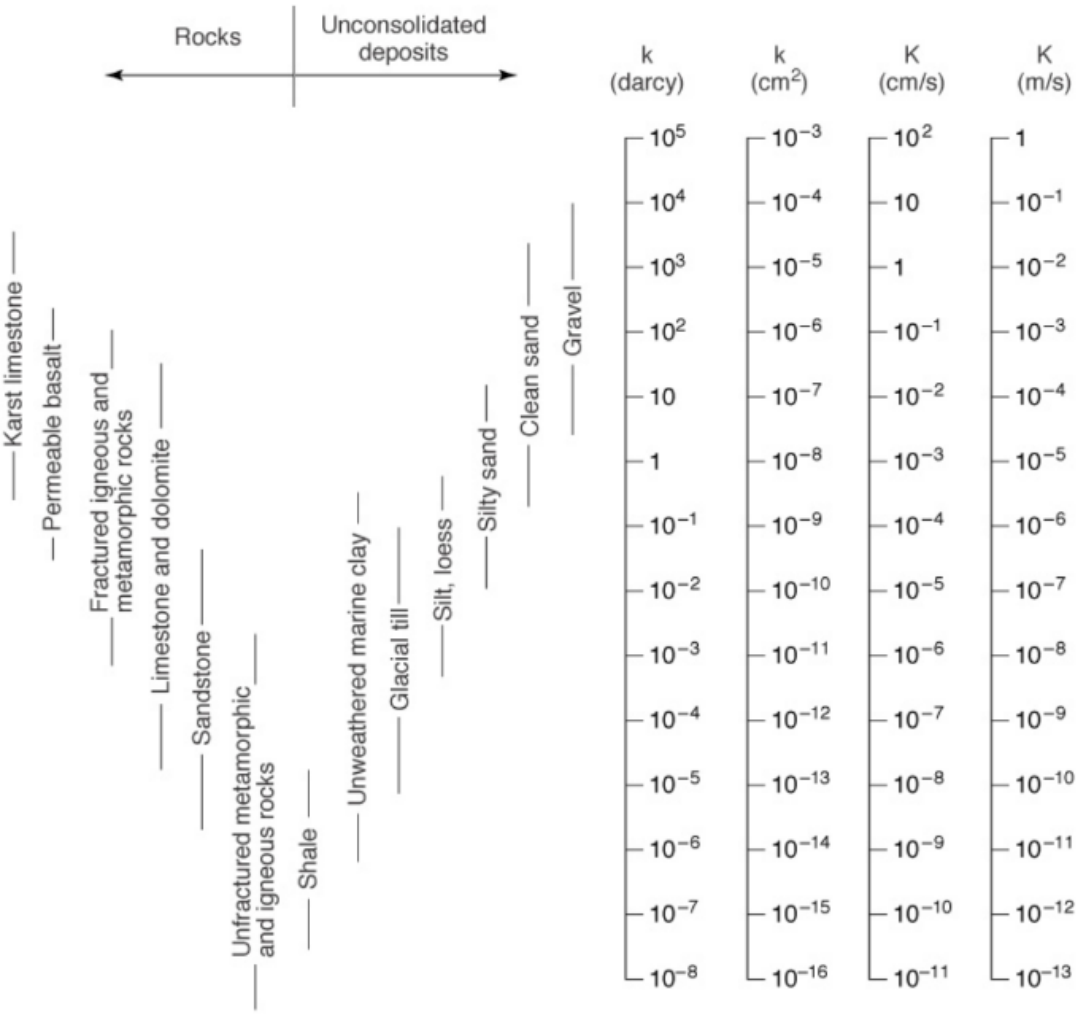
The intrinsic permeability of a consolidated rock is $2,7 \cdot 10^{-11} cm^2$. What is the hydraulic conductivity for water at 20°C

```
print("Let us find the hydraulic conductivity.")

k = 2.7e-15 # intrinsic permeability [m^2]
rho = 999.7 # density at 20°C [kg/m^3]
eta = 0.013097 # dynamic viscosity at 20°C [Pa*s]
g = 9.81 # [m/s^2]

# solution
K = k * ((rho * g)/eta)

print("intrinsic permeability = {} m\u00b2\ndensity = {} kg/m\u00b3\ndynamic␣
↪viscosity = {} Pa*s".format(k, rho, eta),"\n")
print("The resulting hydraulic conductivity at 20°C is {0:0.1e} m/s".format(K))
```

```
Let us find the hydraulic conductivity.
intrinsic permeability = 2.7e-15 m²
density = 999.7 kg/m³
dynamic viscosity = 0.013097 Pa*s

The resulting hydraulic conductivity at 20°C is 2.0e-09 m/s
```

Properties of Intrinsic Permeability

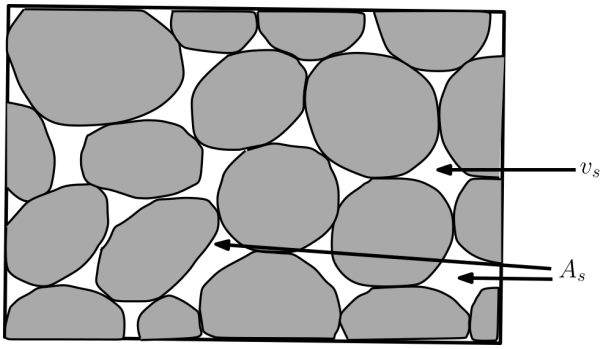
- The value of intrinsic permeability of a porous medium equals 1 m^2 if a fluid with dynamic viscosity of 1 Pa·s can pass through the porous medium at a Darcy velocity of 1 m/s under a hydrostatic pressure gradient of 1 Pa/m (horizontal flow).
- The intrinsic permeability is *independent* of the fluid moving through the medium and depends only upon the medium properties. Intrinsic permeability of unconsolidated porous media is roughly proportional to the square of the pore diameter.
- The intrinsic permeability is used primarily when the density or the viscosity of the fluid varies with position.
- The dimension of k is [L^2], but when expressed in m^2 is so small that square micrometers (μ m)$^2 = 10^{-12}$ m^2 is used. In the petroleum industry it is expressed in **Darcy** (symbol: D) with conversion factor to SI units is: 1 D $= 0.987 \cdot 10^{-12}$ m^2.
- Intrinsic permeability for a weakly , well and highly permeable aquifers vary in the range 10^{-4} to 10^{-1} D, 10^{-1} to 10^2 D, and $> 10^2$ D respectively
- Typical value of conductivity and intrinsic permeability is provided in the fig (from Todd and Mays, 2004) below.



2.5.4 Darcy velocity and Interstitial velocity

Darcy velocity v is the *apparent velocity* or *fictitious velocity* or *Darcy flux* (discharge per unit area). This value of velocity, often referred to as the *apparent velocity*, is not the velocity which the water traveling through the pores is experiencing. The velocity v is referred to as the Darcy velocity because it assumes that flow occurs through the entire cross section of the material without regard to solids and pores. Actually water can flow though pores only and the pore spaces vary continuously with location within the medium. Therefore the actual velocity is nonuniform, involving endless accelerations, deceleration, and changes in direction. To define the *actual flow velocity* or **interstitial velocity**, one must consider the microstructure of the rock material.

For naturally occurring geologic materials, the microstructure cannot be specified three- dimensionally; hence, actual velocities can only be quantified statistically.



Actually, the flow is limited to the pores (white spaces in Figure) only so that the *average interstitial velocity* or *actual velocity* or *seepage velocity* (v_s) through pore space can be determined by applying continuity equation

$$Q = A_s \cdot v_s = Av$$

Leading to

$$v_s = v\frac{A}{A_s} = \frac{v}{\nu_e}$$

where A = total area of soil specimen, and A_s = area of pores only (see Figure). The velocity is divided by effective porosity (ν_e) to account for the fact that only a fraction of the total aquifer volume is available for flow. This indicates that for a sand with a porosity of 33% the $v_s = 3v$. Thus the average interstitial velocity or seepage velocity or linear velocity through pore space is never smaller than Darcy velocity. Sometimes, the average flow velocity of water in the pore space is termed *linear velocity*.

Example problem

Interstitial velocity

From the data below obtain the average interstitial velocity in the Darcy's column.

```
print("Provided data are:\n")

Q = 0.005 # Flow rate [m^3/s]
A = 1000 # total area of soil specimen [m^2]
ne = 0.4 # effective porosity [-] =
```

(continues on next page)

(continued from previous page)

```
#solution
vs = Q / (ne * A)*3600*24

print("Flow rate = {} m\u00b3/s\ntotal area = {} m\u00b2\neffective porosity = {} ".
↪format(Q, A, ne),"\n")
print("The resulting average interstitial velocity is {} m/d".format(vs))
```

```
Provided data are:

Flow rate = 0.005 m^3/s
total area = 1000 m^2
effective porosity = 0.4

The resulting average interstitial velocity is 1.08 m/d
```

Typical values of linear velocities

Typical values for average interstitial velocities or linear velocities in unconsolidated aquifers are 0.5 m/d to 1 m/d and 30 m/d to 300 m/d in sand and gravel respectively.

Linear velocities in fractured or karstified aquifers can be rather high along fractures or conduits e.g. 200 m/d to 1.2 km/d along fractures and 3 km/d to 14 km/d in karst conduits.

On the contrary, the linear velocities are very low in the rock matrix of consolidated aquifers (1 cm/d or even less).

Travel time and Pore volume

The average linear/pore velocity is the velocity a conservative tracer/dye experiences if carried by water through the aquifer.

The travel time of water through a column of length L is given by

$$t = L/v_s$$

It is to be noted that the linear/seepage velocity v_s has to be used in travel time computation, not the Darcy velocity. The term *residence time* is can also be found to be used for referring to *travel time* .

Yet another important term that is often found in standard texts is *pore volume*. The _pore volume is the travel time through a column. It can be understood as the *time* needed to replace the water in the column. In this sense, the pore volume is not a *volume* but a *time* (i.e., 1 PV corresponds to the ratio L/v_s). The pore volume (PV) is frequently used for *normalisation* purposes in order to better compare column experiments conducted under different flow velocities. This is mostly done for studying the transport behaviour of chemicals dissolved in water and their arrivals at the column outlets

Example problem

Travel time and pore volume

In a tracer test, the breaktrought was measured after 100 h at a distance of 200 m. Determine the linear velocity and the pore volume. what is the darcy velocity, if there is an effective porosity of 0.25.

```
print("Provided are:\n")

t = 100 # travel time of water [h]
L = 200 # Distance from injection to measurement [m]
ne = 0.25 # effective porosity [-]


#solution
vs = L / t
PV = L / vs
v = vs * ne

print("travel time of water = {} s\nLength = {} m\u00b2\neffective porosity = {} ".
↪format(t, L, ne),"\n")
print("The linear velocity is {} m/h \nthe pore volume is {} s, and \nthe darcy␣
↪velocity is {} m/h".format(vs, PV, v))
```

```
Provided are:

travel time of water = 100 s
Length = 200 m²
effective porosity = 0.25

The linear velocity is 2.0 m/h
the pore volume is 100.0 s, and
the darcy velocity is 0.5 m/h
```

2.6 Lecture 5. Aquifer Heterogeneity and Anisotropy

(The contents presented in this section were re-developed principally by Prof. B. R. Chahar and Dr. P. K. Yadav. The original contents are from Prof. Rudolf Liedl)

2.6.1 Motivation

The last lecture introduces aquifer properties such hydraulic conductivity, storativity, porosity. The key assumption in that lecture was that the aquifer is an 1D unit, e.g., the Darcy column, and that its properties do not vary in space. In contrast to these, an aquifer is more accurately represented by a 3D system and its properties vary both in space and directions. In fact variations of hydraulic conductivity (K), a most critical aquifer quantity, are dominant in most cases. Variations of K can be observed at very small spatial scales and directions. Thus, aquifer properties that depends on K also varies. Consequently, aquifer properties such as K takes a tensor form- a quantity whose magnitude is space and direction dependent.

In the picture below (Fig. **??**) the 3-D spatial variability of aquifer properties is illustrated by a 2-D vertical cross-section. The picture of the outcrop show some form of a layered system, with each layer likely possessing individual subsurface properties.

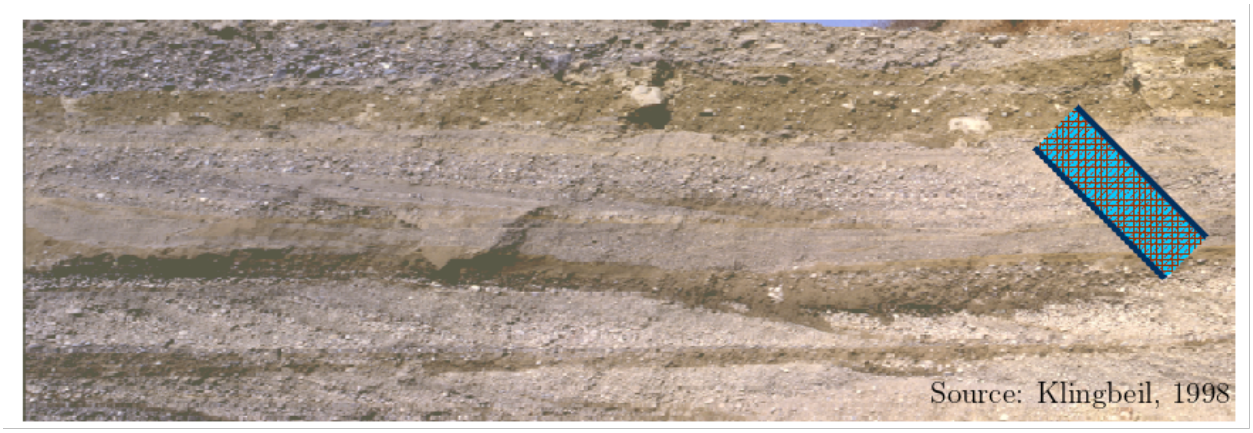


Fig. 2.1: Aquifer Heterogeneity

2.6.2 Heterogeneity

A solid or a porous medium is **homogeneous** if its property do not vary in space. In contrast, the porous medium is **heterogeneous**, or also sometime termed inhomogeneous, if at least one of its properties varies in space. In groundwater studies, heterogeneity or homogeneity is generally associated with hydraulic conductivity (K) of the aquifer. In many practical applications, properties such as strativity and porosity are treated as spatially constant or homogeneous. This is usually done for two reasons:

- the spatial variability of hydraulic conductivity is much more pronounced than that of storativity and porosity, and
- very limited data are available of the spatial variability of storativity and porosity.

Thus, an aquifer is:

homogeneous if $K(x, y, z) = K$, and

heterogeneous if $K(x, y, z) \neq K$

For heterogeneous aquifer it is not required that K is varying in all directions, i.e., varying of K can be restricted to one or two spatial coordinates.

Heterogeneity in aquifer can exist in many configurations. But they are broadly categorized to the following three classes:

layered heterogeneity: common in sedimentary systems and unconsolidated deposits.

discontinuous heterogeneity: due to presence of faults, e.g., at the contact between overburden-bedrock.

trending heterogeneity: Common in the aquifers with active sedimentation processes

Aquifer properties such as storativity and porosity are affected by heterogeneity but they are often considered spatially constant in practical applications. This is because

- The spatial variability of hydraulic conductivity is much more pronounced than any other aquifer property, and
- Only limited spatial variability data are available for storativity or porosity.

Heterogeneity play a significant role in controlling the flow of groundwater, but its quantification is more relevant to the transport of chemicals in groundwater.

Effective Hydraulic Conductivity

It is possible to represent the spatial distribution of hydraulic conductivity in a heterogeneous aquifer by an *average* value such that steady-state groundwater discharge remains the same as in the heterogeneous case. Such obtained *average* K value is termed as **effective hydraulic conductivity**. In other words, it can be mentioned that the effective hydraulic conductivity characterizes a fictitious homogeneous aquifer with the same discharge and the same overall head difference under steady-state conditions as some heterogeneous aquifer to be investigated.

The quantification of *effective hydraulic conductivity* is straight-forward for perfectly layered systems, which can be seen as an idealized representation of natural layering. In more natural systems with complex heterogeneous configurations, cumbersome mathematical derivations are required to obtain effective K.

In the next few sections, effective hydraulic conductivity for ideal layered system is derived.

2.6.3 Layered Systems

Case I - Flow Parallel to Layering

A confined aquifer consisting of n layers is considered. In the set-up (see Fig. **??**), the layer boundaries are parallel to each other and groundwater flow is parallel to the layering.

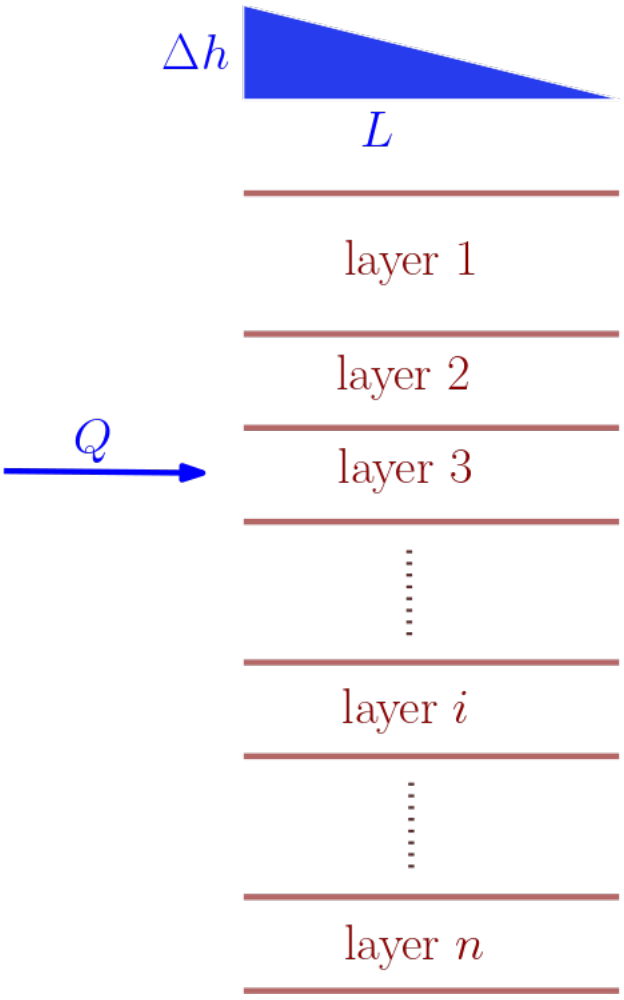


Fig. 2.2: Flow parallel to layering

The analysis is required to consider the aquifer as a *single entire aquifer unit* and a set of *layered aquifer unit*. Data of both units are to be considered.

Data for a single entire aquifer unit are:

- W = width (extension perpendicular to cross-section, see figure)
- L = flow distance along the layer
- Δh = total head loss corresponding to flow length
- Q = the total discharge.

Data for set of layered aquifer unit are:

- i = layer numbers $(i = 1, 2, 3..., n)$
- m_i = thickness of i^{th} layer
- K_i = conductivity of i^{th} layer

With these informations available, using volumetric budget and Darcy's law the *effective hydraulic conductivity* K can be quantified from the following steps:

I. Total thickness: $m = \sum_{i=1}^{n} m_i$

II. Volumetric budget: $Q = \sum_{i=1}^{n} Q_i$

III. Head loss in i^{th} layer: $\Delta h_i = \Delta h$

IV. Darcy's law for i^{th} layer: $Q_i = -wm_iK_i\frac{\Delta h}{L}$

V. Darcy's law for the homogeneous aquifer to replace the layered system: $Q = -wmK\frac{\Delta h}{L}$

Note: As flow is parallel to the layering, the head loss in individual layer equals the total head loss (step. 3)

Summing the discharge in each layer (step 4) will result to the discharge of the homogeneous aquifer (step 5), i.e., we can equate step 4 and step 5 as:

$$-w \cdot m \cdot K \cdot \frac{\Delta h}{L} = \sum_{i=1}^{n} \left(- w \cdot m_i \cdot K_i \cdot \frac{\Delta h}{L} \right)$$

Constant quantities from the right-hand side can be taken out of summation and equated with the left side. This leads to:

$$-w \cdot m \cdot K \cdot \frac{\Delta h}{L} = -w \cdot \frac{\Delta h}{L} \cdot \sum_{i=1}^{n} (m_i K_i)$$

providing

$$m \cdot K = \sum_{i=1}^{n} (m_i K_i)$$

As a result, the effective hydraulic conductivity of a layered system when the flow is parallel to the layering equals

$$K = \frac{\sum_{i=1}^{n} (m_i K_i)}{m}$$

In the above equation, effective hydraulic conductivity K is obtained as the *weighted arithmetic average* of layer conductivities K_i. Weights correspond to relative thickness m_i/m. Thus, the *largest term* in the sum contributes most to the arithmetic average. Thus, the effective hydraulic conductivity K can be approximated from

$$K \approx \frac{\max(m_i \cdot K_i)}{m}$$

Example problem

Flow parallel to layering

Calculate the effective hydraulic conductivity of the layer system consisting of 3 layers if the flow is parallel to the stratification.

```
import numpy as np

print("\033[1m Provided are: \033[0m \n")

#Thickness of i-th layer [m]
m1 = 3
m2 = 2.5
m3 = 1.75

#conductivity of i-th layer [m/s]
K1 = 3.5e-3
K2 = 2e-2
K3 = 5e-4


#intermediate calculation
m = m1+m2+m3

#solution
K = (m1*K1+m2*K2+m3*K3)/m
print("thickness of layer 1 = {}".format(m1), "m\nthickness of layer 2 = {}".
↪format(m2),"m\nthickness of layer 3 = {}".format(m3), "m\n\nconductivity of layer 1␣
↪= {:02.1e}".format(K1),
      "m/s\nconductivity of layer 2 = {:02.1e}".format(K2), "m/s\nconductivity of␣
↪layer 3 {:02.1e}".format(K3), "m/s")
print("\n\033[1mSolution:\033[0m\nThe resulting hydraulic conductivity of the layer␣
↪system is \033[1m{:02.1e} m/s\033[0m.".format(K))
```

```
 Provided are:

thickness of layer 1 = 3 m
thickness of layer 2 = 2.5 m
thickness of layer 3 = 1.75 m

conductivity of layer 1 = 3.5e-03 m/s
conductivity of layer 2 = 2.0e-02 m/s
conductivity of layer 3 5.0e-04 m/s

Solution:
The resulting hydraulic conductivity of the layer system is 8.5e-03 m/s.
```

Case 2 - Flow Perpendicular to Layering

In the second case the the flow *perpendicular* to the layering is considered. Just as in the *parallel* flow case, the aquifer in this case also is a confined aquifer with n layers (see Fig. **??**).

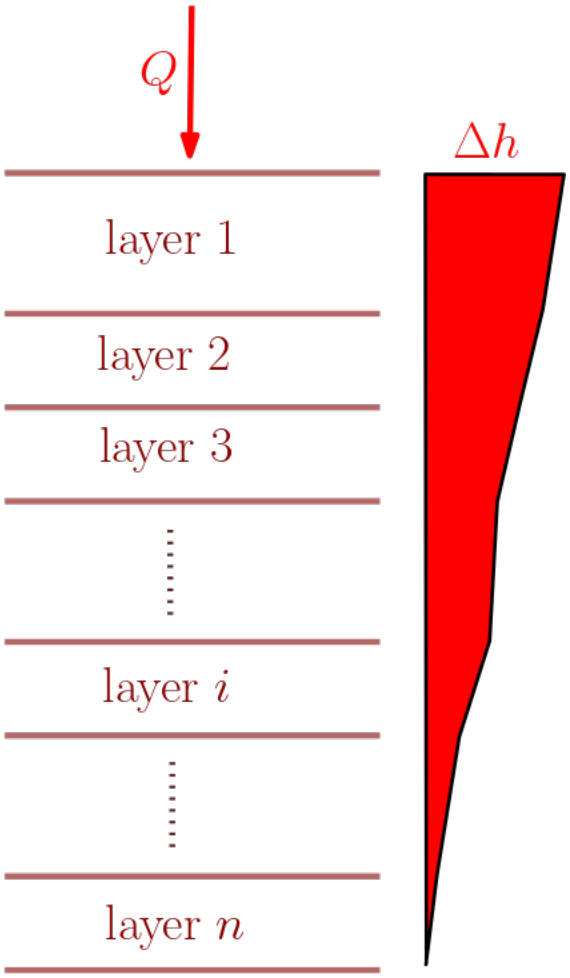


Fig. 2.3: Flow perpendicular to layering

Data for a single entire aquifer unit are:

- A = total cross-sectional area
- Δh = total head loss
- Q = total discharge

And, the available data for the *layered units* are:

- i = layer number i $(i = 1, 2, 3, \ldots, n)$
- m_i = thickness of i^{th} layer
- K_i = Hydraulic conductivity of i^{th} layer

These data combined with *equation of continuity* and *Darcy's Law* can be used to obtain the effective hydraulic conductivity of the system in which flow is perpendicular to the layers. The following steps are required:

I. Total thickness: $m = \sum_{i=1}^{n} m_i$

II. Equation of continuity: $Q_i = Q$

II. Cross-sectional area for layer: $A_i = A$

IV. Decomposition of head loss: $\Delta h = \sum_{i=1}^{n} \Delta h_i$

V. Darcy's law for layer i:

$$Q_i = -A_i K_i \frac{\Delta h_i}{m_i}$$

$$\Delta h_i = -\frac{Q_i m_i}{A_i K_i} = -\frac{Q m_i}{A K_i}$$

VI. Similarly the Darcy's law for the homogeneous aquifer to replace the layered system:

$$Q = -AK\frac{\Delta h}{m}$$

$$\Delta h = -\frac{Qm}{AK}$$

VII. The head loss from step 4 can be equated with the sum of head loss of each layered unit (from step 5), i.e.,

$$\frac{Q \cdot m}{A \cdot K} = \sum_{i=1}^{n} \frac{Q \cdot m}{A \cdot K}$$

The constants Q and A can be taken out of the summation, this leads to

$$\frac{-Q \cdot m}{A \cdot K} = \frac{-Q}{A} \sum_{i=1}^{n} \frac{m_i}{K_i}$$

As a result, the effective hydraulic conductivity of a layered system for a flow perpendicular to the layering equals

$$K = \frac{m}{\sum_{i=1}^{n} \frac{m_i}{K_i}}$$

In the above equation, effective hydraulic conductivity K is obtained as the *weighted harmonic average* of layer conductivities K_i. Weights correspond to relative thicknesses m_i/m. Thus, the largest term in the sum contributes most to the harmonic average and therefore, the effective hydraulic conductivity can be approximated from

$$K \approx \frac{m}{\max\left(\frac{m_i}{K_i}\right)}$$

Example problem

Flow perpendicular to layering

Calculate the effective hydraulic conductivity of the layer system consisting of 3 layers if the flow is perpendicular to the layering.

```
print("\033[1m Provided are:\033[0m")

#Thickness of i-th layer [m]
m1 = 3
m2 = 2.5
m3 = 1.75

#conductivity of i-th layer [m/s]
K1 = 3.5e-3
K2 = 2e-2
K3 = 5e-4

#intermediate calculation
m = m1+m2+m3

#solution
K = m/(m1/K1+m2/K2+m3/K3)

print("thickness of layer 1 = {}".format(m1), "m\nthickness of layer 2 = {}".
↪format(m2),"m\nthickness of layer 3 = {}".format(m3), "m\n\nconductivity of layer 1␣
↪= {:02.1e}".format(K1),
      "m/s\nconductivity of layer 2 = {:02.1e}".format(K2), "m/s\nconductivity of␣
↪layer 3 {:02.1e}".format(K3), "m/s")
print("\n\033[1mSolution:\033[0m\nThe resulting hydraulic conductivity of the layer␣
↪system is \033[1m{:02.1e} m/s\033[0m.".format(K))
```

```
 Provided are:
thickness of layer 1 = 3 m
thickness of layer 2 = 2.5 m
thickness of layer 3 = 1.75 m

conductivity of layer 1 = 3.5e-03 m/s
conductivity of layer 2 = 2.0e-02 m/s
conductivity of layer 3 5.0e-04 m/s

Solution:
The resulting hydraulic conductivity of the layer system is 1.6e-03 m/s.
```

Summary: Effective Conductivity of Layered Aquifers

- **For flow parallel to layering:** Effective hydraulic conductivity equals the *weighted arithmetic mean* of layer conductivities.
- **Flow perpendicular to layering:** Effective hydraulic conductivity equals the *weighted harmonic mean* of layer conductivities.
- **Weights** in both cases are given by relative layer thicknesses m_i/m
- It can be mathematically shown that the harmonic mean of a set of positive numbers cannot exceed the arithmetic mean of the same set. Both means are identical only if all numbers in the set are identical. Apart from this very special case, we have

Note:

harmonic mean < arithmetic mean.

This implies that the flow direction perpendicular to the layering is associated with a smaller effective hydraulic conductivity than the flow direction parallel to the layering.

2.6.4 Hydraulic Resistance

The *Hydraulic Resistance* (R) provide an alternative approach to express conductivity. It is defined as a reciprocal to hydraulic conductivity (K), i.e.,

$$R = \frac{1}{K}$$

This implies that large K corresponds to small R and vice-versa. Considerations about effective hydraulic conductivities of layered aquifers can be transferred to hydraulic resistances by recalling that the arithmetic mean of positive numbers coincides with the harmonic mean of reciprocal numbers and vice versa. This can be used in the following manner:

- **For *flow parallel to layering*:**

As Effective hydraulic conductivity equals the weighted arithmetic mean of layer conductivities, the reciprocal of this, i.e., the effective hydraulic resistance, will equal the weighted harmonic mean of layer resistances.

Furthermore, if all layer thicknesses are identical ($m_i = \text{const.}$) and flow is parallel to layering, the largest discharge passes through the layer with highest hydraulic conductivity (smallest hydraulic resistance). In this case, the discharge through each layer is proportional to layer conductivity or inversely proportional to layer resistance.

- **For *flow perpendicular to layering*:**

In this case the effective hydraulic conductivity equals the weighted harmonic mean of layer conductivities. This leads to Effective hydraulic resistance equaling the weighted arithmetic mean of layer resistances.

Similar to the *flow parallel to the layering,* if all layer thicknesses are identical ($m_i = \text{const.}$) and flow is perpendicular to layering, the largest hydraulic gradient is across the layer with lowest hydraulic conductivity (highest hydraulic resistance). In this case, the *head gradient* across each layer is proportional to layer resistance or inversely proportional to layer conductivity.

Example problem

Hydraulic Resistance

Find the hydraulic resistance with the given hydraulic counductivity

---

```
print("\033[1m Provided are:\033[0m \n")

K = 5e-4 # m/s, hydraulic conductivity

#solution
R = 1/K

print("Hydraulic conductivity = {:02.1e} m/s".format(K))
print("\n\033[1mSolution:\033[0m\n\nThe resulting hydraulic resistance is \033[1m{:02.
↪1e} s/m\033[0m.".format(R))
```

```
 Provided are:

Hydraulic conductivity = 5.0e-04 m/s
```

(continues on next page)

(continued from previous page)

```
Solution:

The resulting hydraulic resistance is 2.0e+03 s/m.
```

2.6.5 Aquifer Anisotropy

A solid or a porous medium is **isotropic** if its (all) properties are independent of *direction.* Conversely, a solid or a porous medium is **anisotropic** if at least one of its property is dependent on direction. Thus, isotropic (or anisotropic) property of porous media refines the concept of homogeneity (or heterogeneity). The key difference being that anisotropy of aquifer is associated only with hydraulic conductivity as other aquifer properties like storativity or porosity cannot depend on direction. Groundwater flow (and more so solute transport) is affected by anisotropy. However, in unconsolidated aquifers the impact of heterogeneity is usually more important.

Figure below (Fig. **??**) explains the concept of isotropy more clearly. The direction dependency of K is represented by the arrows diagram.

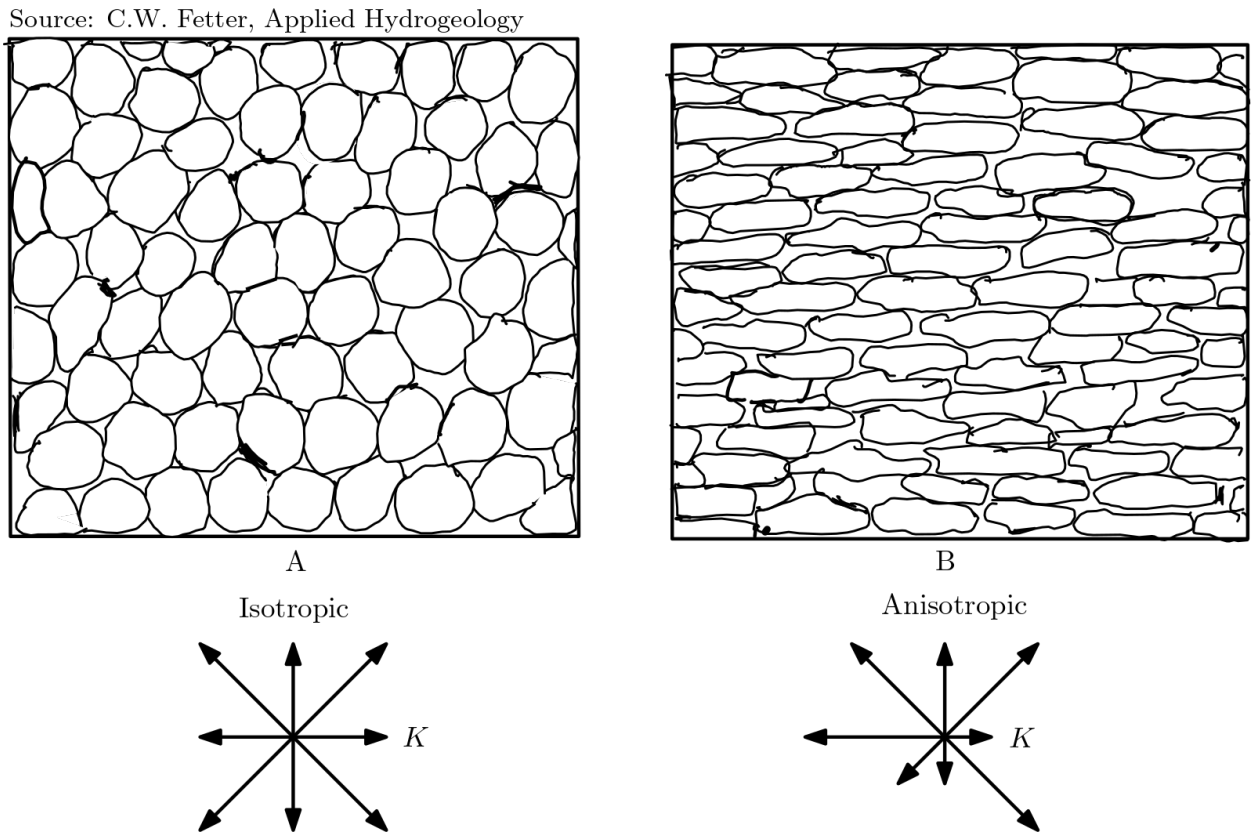


Fig. 2.4: Isotropy and Anisotropy in aquifers

Anisotropy and scale effects

The effective hydraulic conductivity of layered aquifers was shown earlier to depend on the orientation of the flow direction relative to layering, i.e., parallel versus perpendicular. On a larger scale, it may not be possible to identify or resolve heterogeneities associated with example thin layers, small lenses, shape and orientation of grains (see figure above). Nevertheless, K is found to be direction-dependent when groundwater flow is quantified at the larger scale. In these case, small-scale heterogeneity (e.g., due to layering) expresses itself as anisotropy of hydraulic conductivity at the larger scale.

Indices and functional arguments to describe anisotropy

One could consider heterogeneity as a property of entirety. While function arguments point towards heterogeneity, *indices* are used to express that hydraulic conductivity depends on direction. Considering a 3-D Cartesian coordinate system, the directional dependent hydraulic conductivity can be represented by K_x, K_y and K_z in parallel with the $x-$, $y-$ and $z-$axis, respectively. Within these, one could distinguish between horizontal conductivities (K_x and K_y) with the vertical conductivity K_z. More often, anisotropy is observed only between horizontal and vertical directions (i.e., along x or y and z directions), while isotropy is observed along horizontal directions (x and y direction). Thus, symbols K_h representing $K_x = K_y$ and K_v representing K_z can be used to denote horizontal and vertical hydraulic conductivity, respectively.

Concept of the hydraulic conductivity ellipse

Let us consider an aquifer with horizontal conductivity K_h and vertical hydraulic conductivity K_v. Let us assume that the flow in the aquifer is in some arbitrary direction characterized by the angle θ between the flow direction and horizontal plane (see Fig. **??**).

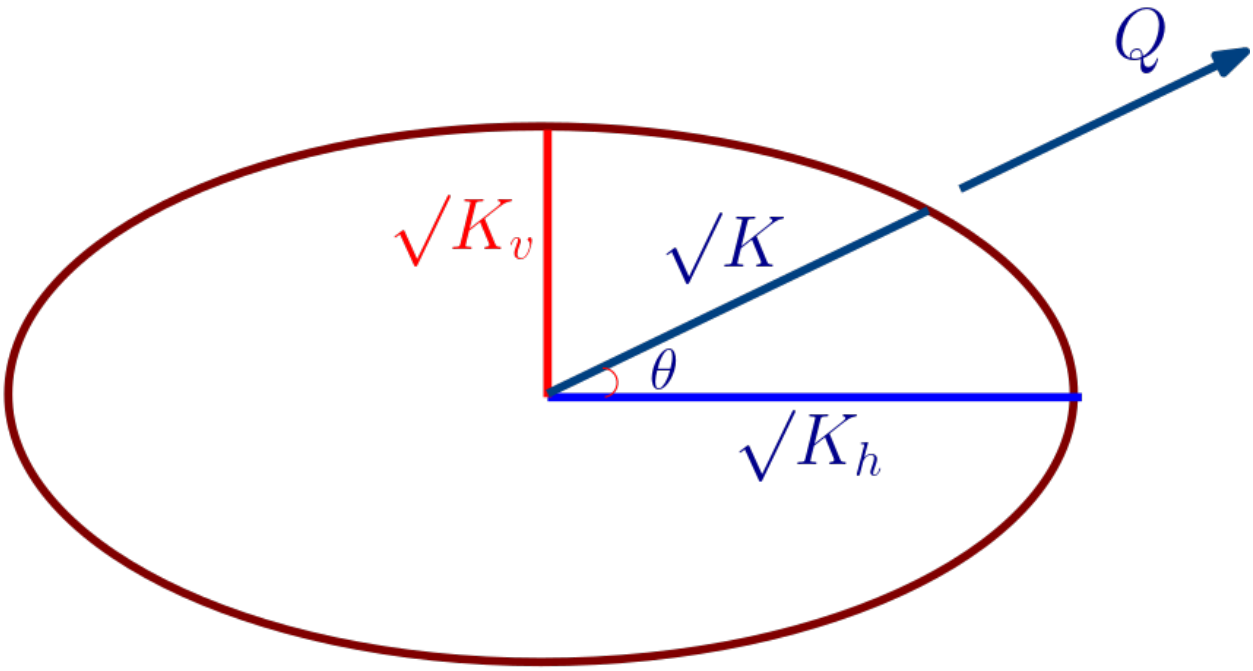


Fig. 2.5: hydraulic conductivity ellipse

In this case, it can be shown that the effective hydraulic conductivity K is

$$K = \frac{1}{\frac{\cos^2\theta}{K_h} + \frac{\sin^2\theta}{K_v}}$$

If the angle θ is varied, the above equation defines an ellipse (also called "hydraulic conductivity ellipse") with semi-axes equal to $\sqrt{K_h}$ and $\sqrt{K_v}$, respectively. The square root of K can be visualised by the length of a line segment parallel to the direction of flow. This line segment extends from the centre to the perimeter of the ellipse.

Example problem

Hydraulic Resistance

Find resulting hydraulic conductivity from provided horizontal and vertical conductivities.

```
print("\n\033[1mProvided are:\033[0m\n")

Kh =  1e-3 #horizontal hydraulic conductivity [m/s]
Kv =  1e-4 #vertical hydraulic conductivity [m/s]
theta = 50 #angle between flow direction ans horizontal plane [°]

#solution
K = 1 /((np.cos(theta)**2/Kh)+(np.sin(theta)**2/Kv))

print("horizontal hydraulic conductivity = {:02.1e}".format(Kh), "m/s\n" "vertical␣
↪hydraulic conductivity = {:02.1e}".format(Kv), "m/s\n"
      "angle = {}°\n".format(theta))
print("\033[1mSolution:\033[0m\n\nThe resulting hydraulic conductivity is \033[1m{:02.
↪1e} m/s\033[0m.".format(K))
```

```
Provided are:

horizontal hydraulic conductivity = 1.0e-03 m/s
vertical hydraulic conductivity = 1.0e-04 m/s
angle = 50°

Solution:

The resulting hydraulic conductivity is 6.2e-04 m/s.
```

2.6.6 Combining Heterogeneity and Anisotropy

The distinction between_Homogeneous, Heterogeneous, isotropic_ and *anisotropic* aquifer or any combination of them can be more clearly understood from the figure (for vertical $x - z$ plane).

2.7 Lecture 6: Steady-State Groundwater flow in 3D

2.7.1 Motivation

This lecture shows how Darcy's law can be used for variety of groundwater problems, e.g., in isotropic and anisotropic aquifers, multiple dimension problems - 2D and 3D. The lecture will also introduce the basic concepts on visualizing groundwater flow, e.g., using streamlines, flow-nets, and present their uses for example using isochrones and delineating protection zones.

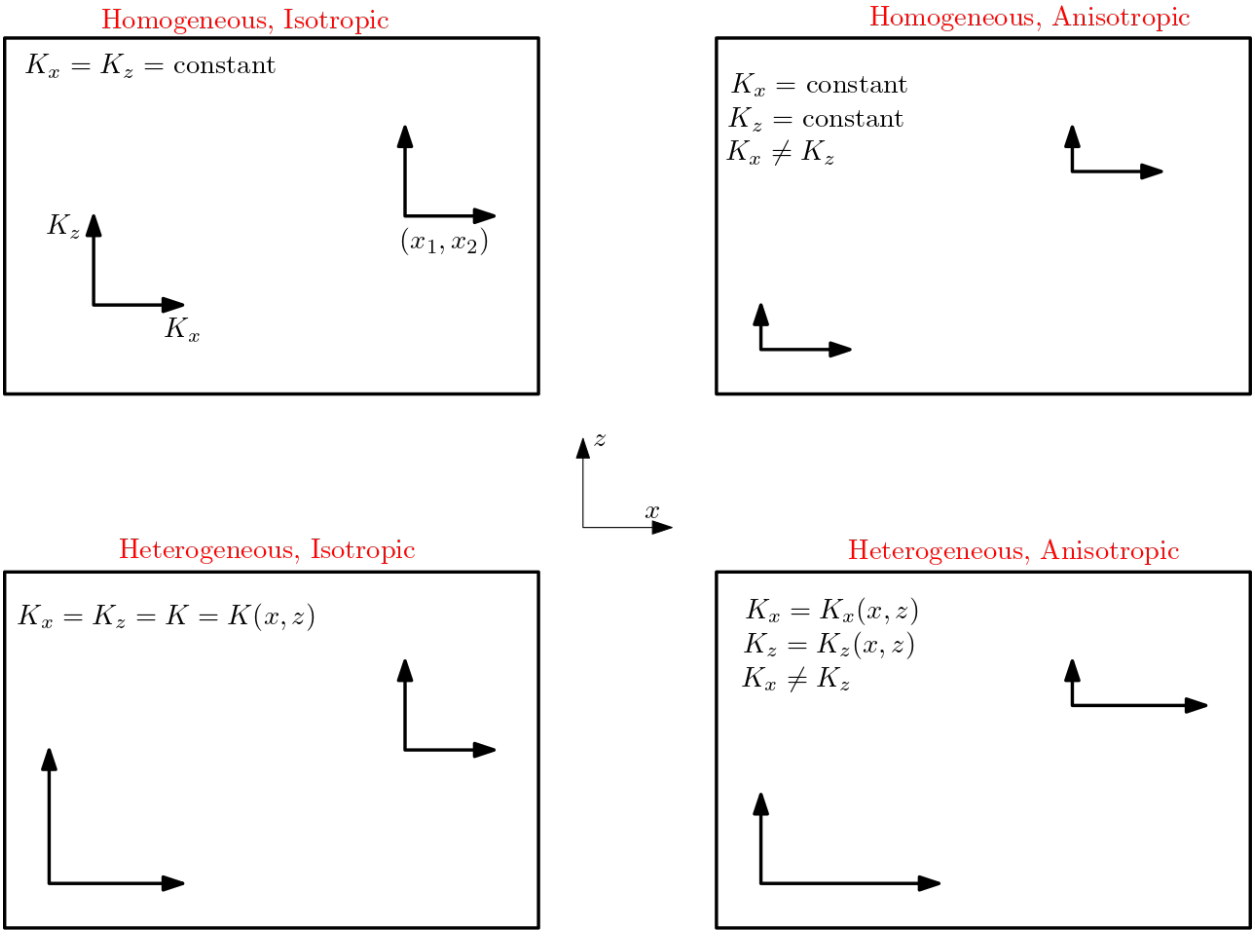


Fig. 2.6: Heterogeneity and Anisotropy in aquifer

2.7.2 Darcy's Law in Isotropic Aquifers

Summarizing Hydraulic Head

The Darcy's column experiment is associated with one-dimensional (1D) steady-state or time independent flow system. In addition, the experiments as we discussed in lecture 4, dealt with homogeneous (space dependency) and isotropic (direction dependency) porous media. The consequence of these is that the hydraulic head h in a Darcy column therefore depends on a single space coordinate, say x that is oriented along the column length. Therefore $h = h(x)$ in this case.

Now considering the real/natural aquifer, which is inherently three dimensional (3D), the hydraulic head is then a function of three space coordinates, i.e., $h = h(x, y, z)$. But the 3D natural problem can be simplified to 2D problems based on the fact that vertical flow components (e.g., the aquifer depth) is very much smaller than horizontal flow components (the aquifer width or its length), i.e., $h = h(x, y)$ considering z as vertical components. The 2D approach, instead of 3D, has been found to appropriately quantify groundwater flow systems. The application of 2D or 1D for quantifying in groundwater flow system have be justified for each case. For example the layered aquifers, that was discussed in lecture 5, can be easily treated as a 2D system.

Note: The hydraulic head (h) in any case (1D, 2D or 3D), is the sum of hydrostatic pressure head (ψ) and the elevation head (z) and has a dimension of length [L], i.e.,

$$h = \psi + z$$

2.7.3 Isolines and Isosurfaces

We make a slight detour from the Darcy's law and quantifying groundwater flow system to **visualizing** the groundwater flow. As can be expected, visualizing groundwater is a challenging task. This is largely because sub-surface, compared to surface, cannot be mapped with very high resolution. However, it is possible to connect aquifer properties such as h hydraulic head to visualize groundwater flow system. Particularly used to visualize groundwater are:

> **Isolines**: The curves (in 2D space) where a physical quantity (e.g., h) assumes a certain value. Isolines are thus suitable for visualizing 2D problems.
>
> **Isosurface**: The surfaces (in 3D space) where a physical quantity assumes a certain value. Isosurface addresses the 3D groundwater problems.

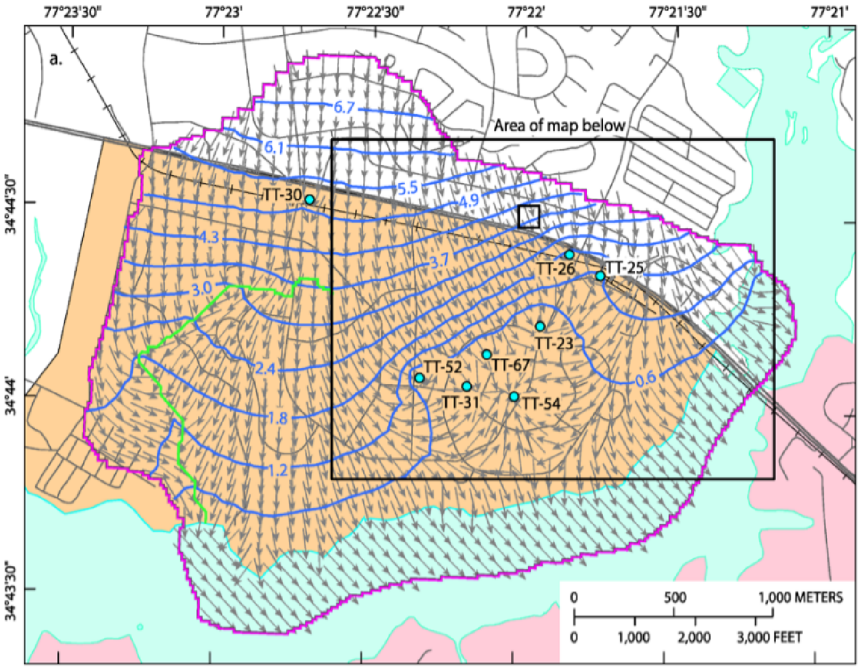
The representation of *isolines* yields a *map*. Isolines are therefore frequently superimposed on an already existing geographical map, e.g., the topographical map (see Fig. **??**). The simultaneous representation of several isolines/isosurfaces can be confusing. Thus intervals between values represented by these representations are constant.

Isosurfaces are in general confusing to interpret due to presence of the third dimensions. As such often 2D cross-section through isosurfaces are depicted.

2.7.4 Hydrologic Triangle

The **hydrologic triangle** is a method that is used to approximate head isolines when head values are available at some distinct locations. As the name suggests, at least three head data are required to be known for approximating isolines. The process used in approximating is as follows:

- First the known head data points are connected with straight lines (dashed lines in the figure Fig. **??**).
- Linear interpolation is then performed with pre-defined head intervals ($\Delta h = 1m$ in the figure).
- Finally, points with identical head values are connected to obtain the isolines (solid lines in the figure).



—1.8—

Simulated potentiometric contour. The contour intervals are 0.6 m and 0.7 m.

Simulated direction of groundwater flow

Site: U.S. Marine Corps Base Camp Lejeune, North Carolina

Source: Maslia et al. (2009)

Fig. 2.7: Head Isolines

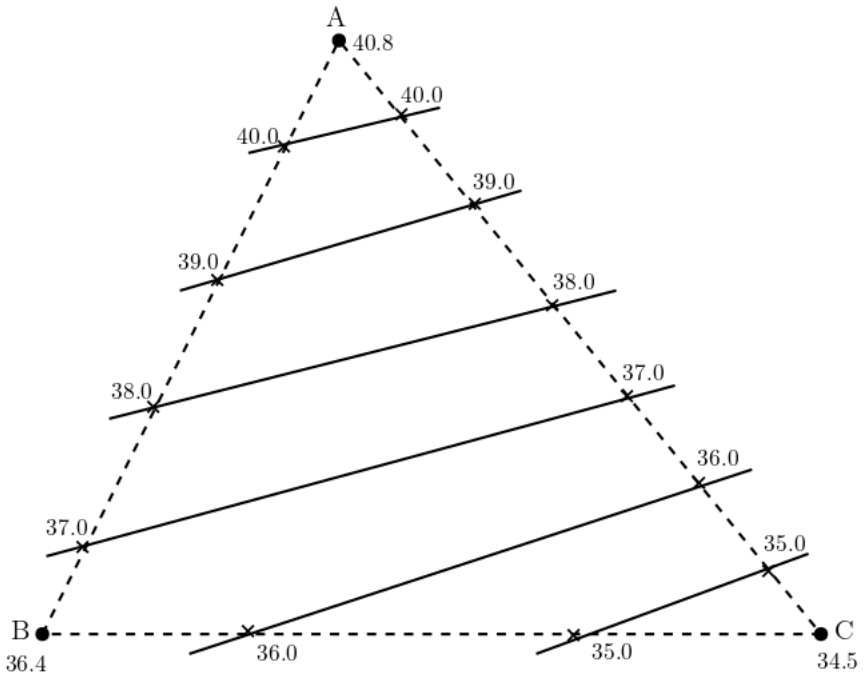


Fig. 2.8: Hydrologic Triangle

Applying the Hydrologic Triangle

In practical cases more than the just three observation points are available. This implies a triangulation of the investigation area as shown in the figure (Fig. **??**). Linear interpolation of heads is again performed along the dashed lines. Points with identical head values are connected to obtain the isolines. The connection lines do not have to be strictly a straight line as was the case in Fig. **??**. In addition to straight line segments, the connected line can be *smooth* curves as shown in the Fig. **??**.

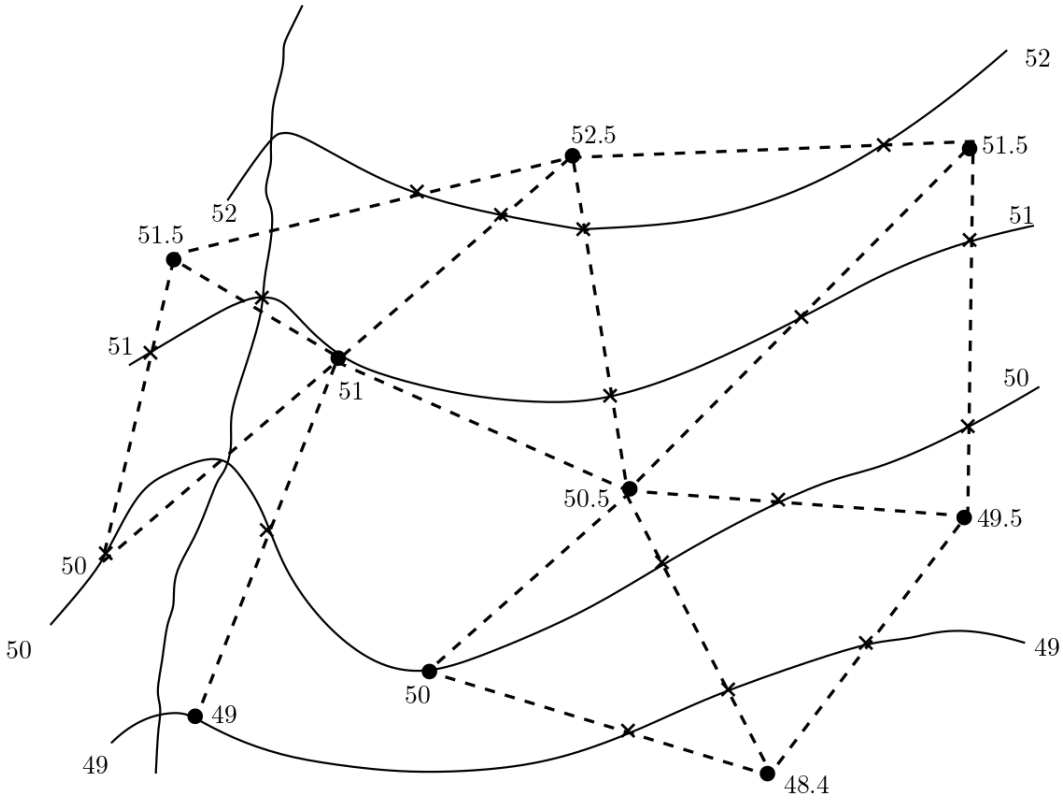


Fig. 2.9: Applying the Hydrologic Triangle

Flow Direction

Based on Darcy's law it is known that the flow is directed from the higher head towards the lower head. Though Darcy's law is based on 1D, the same concept can be extended to the 2D and 3D flow. Thus in the figure, (Fig. **??**), the flow direction can be assumed to be directed from the top towards the base of the triangle (red arrow in the figure). But this remains assumptions so far.

Attention: What is the correct extension of Darcy's law- in 2D and 3D systems?

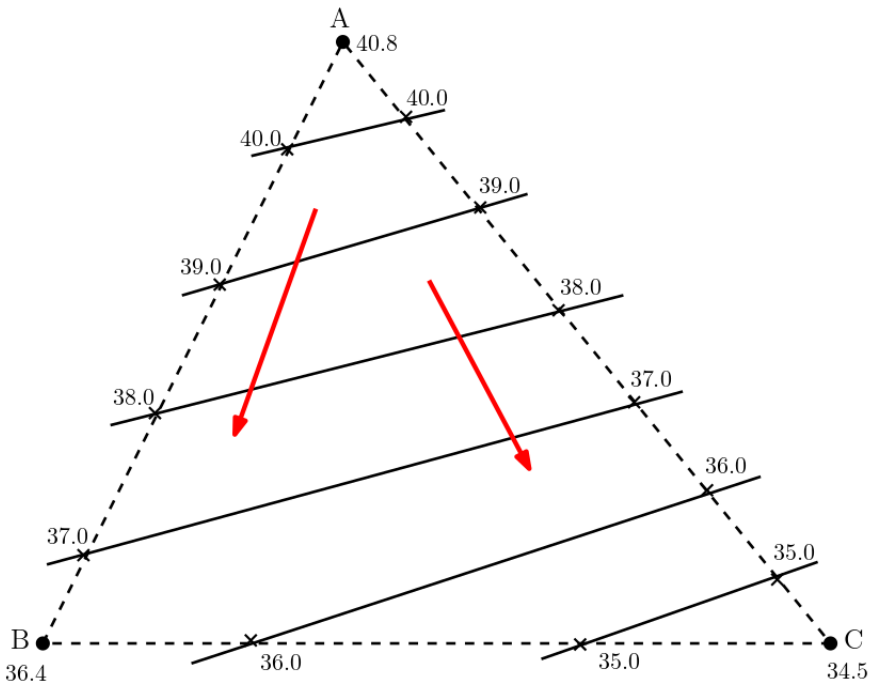


Fig. 2.10: Flow Direction in Hydrologic Triangle

2.7.5 Hydraulic Gradient in 2D and 3D systems

The hydraulic gradient appears as a major constituent of Darcy's law for 1D flow. In the 1D case the hydraulic gradient is given as the ratio of difference of heads at two points in the space and the distance between those points. This distance measures also the flow length. Since head is direction oriented, the hydraulic gradient in the 2D or the 3D flow must be a vector quantity. This combines both magnitude and direction that fits the definition of hydraulic gradient. Therefore hydraulic gradient in 2D or 3D is achieved by employing partial derivatives of hydraulic head with respect to Cartesian coordinates. Mathematically, this can be represented as:

$$\text{grad}h = \nabla h = \begin{bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{bmatrix}$$

grad$h$ in the equation above is a vector pointing in the direction of the steepest **increase** of h. In groundwater/hydrogeology study, in contrast to the mathematical definition, grad$h$ refer to the direction of steepest **decrease** of hydraulic head. This can be observed in Fig. **??**

Darcy's Law (Isotropic Aquifer)

The concept used for hydraulic gradient for 2D/3D discussed above can be extended to specific discharge or Darcy velocity v_f. Similar to hydraulic gradient, v_f is a vector with two or three components for 2D and 3D systems, respectively. Thus the 3D Darcy velocity vector is

$$v_f = \begin{bmatrix} v_{fx} \\ v_{fy} \\ v_{fz} \end{bmatrix}$$

Therefore in higher dimension systems, the Darcy law can be stated as

$$v_f = -K \cdot \text{grad}h$$

Based on this definition, the hydraulic conductivity (K) is

Heterogeneous aquifer: $K = K(x, y, z$, and

Homogeneous aquifer: $K =$ constant

For **isotropic** aquifers the Darcy velocity is oriented opposite to the hydraulic gradient vector (due to minus sign in Darcy's law). This is explained further below.

Example problem

Hydraulic gradient in 2D

Obtain the hydraulic gradients of the following 2D system shown in the figure below

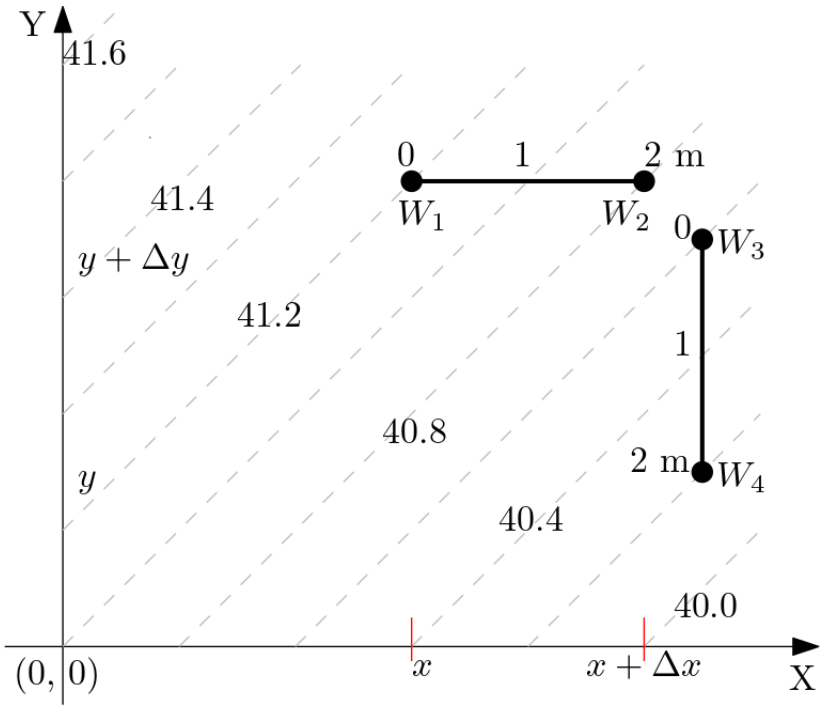


Fig. 2.11: Example 1

Solution

Known for the 2D system is:

$$\nabla h = \begin{bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \end{bmatrix}$$

In which for fixed y,

$$\frac{\partial h}{\partial x} \approx \frac{h(x + \Delta x, y) - h(x, y)}{\Delta x}$$

and for fixed x,

$$\frac{\partial h}{\partial y} \approx \frac{h(x, y + \Delta y) - h(x, y)}{\Delta y}$$

Then from the given figure:

```
# solution

import numpy as np

h_xdelx = 40.8 # m, head at W2
h_x = 41.2 # m, head at W1
Lx = 2 # m, length between W1 and W2

h_xdely = 40.6 # m, head at W2
h_y = 40.2 # m, head at W1
Ly = 2 # m, length between W1 and W2

# calculate
Delh_x = (h_xdelx - h_x)/Lx # (-), hydraulic head along x-axis
Delh_y = (h_xdely - h_y)/Ly # (-), hydraulic head along y-axis

#print
print("Hydraulic gradient along x-axis is {0:0.3f}\n".format(Delh_x))
print("Hydraulic gradient along y-axis is {0:0.3f}\n".format(Delh_y))
```

```
Hydraulic gradient along x-axis is -0.200

Hydraulic gradient along y-axis is 0.200
```

Attention: In the above example the orientation of co-ordinate axis is important. The change in orientation of axis, e.g., y-axis pointing downward will make also hydraulic gradient along y negative.

2.7.6 Streamlines and Flow Nets

Basics

Streamlines or **flowlines** are curves which are in each point tangential to the flow direction. There is no flow component perpendicular to the streamlines. As a consequence, streamlines are perpendicular to head iso-surface of head isolines if the aquifer is **isotropic**

Flow nets consist of a set of isolines (or isosurfaces) and a set of streamlines. Flow nets are ususally employed for 2D flow scenarios only. The flow behaviour is illustrated by covering the investigation area with a mesh of isolines and streamlines.

Radial Flow Near a Well

In this example steady-state water abstraction from a pumping well, e.g., for drinking water supply, is considered. The hydraulic conductivity is assumed to be spatially constant, i.e., the aquifer is *homogeneous*. Now assuming that there are no other hydraulic impacts in the system, the pumping of water from the groundwater will lead to a radially symmetric **drawdown** of *hydraulic heads* (this will be further discussed in future lecture 8). In this case the *streamlines* radially approaches the pumping well, and the *head isolines* are circles with the well in the centre.

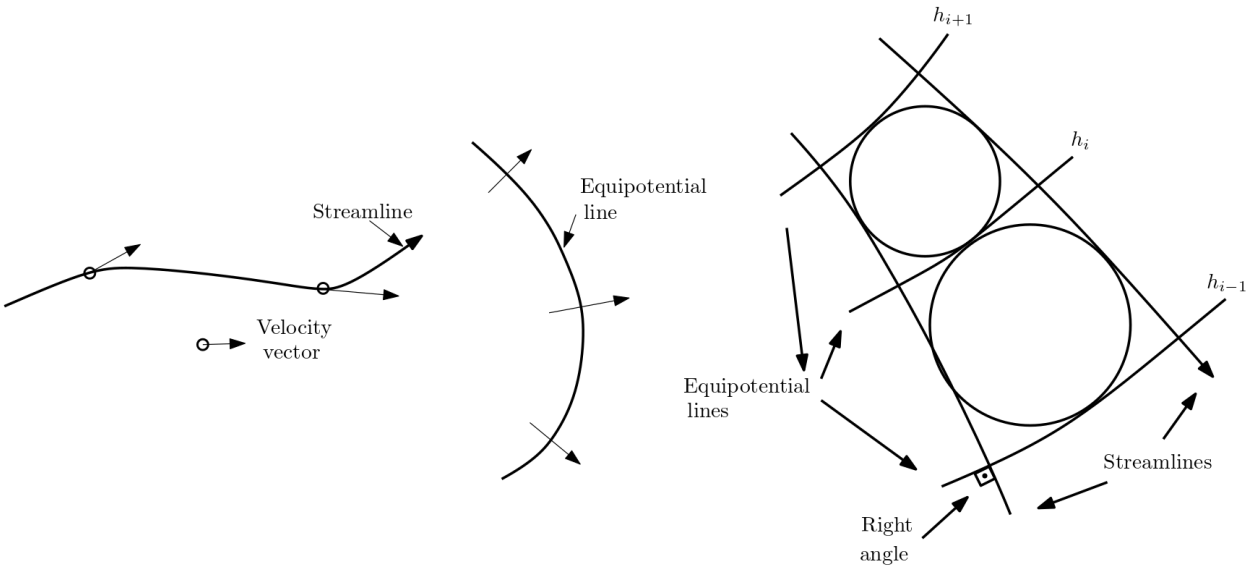


Fig. 2.12: Streamlines and equipotential lines

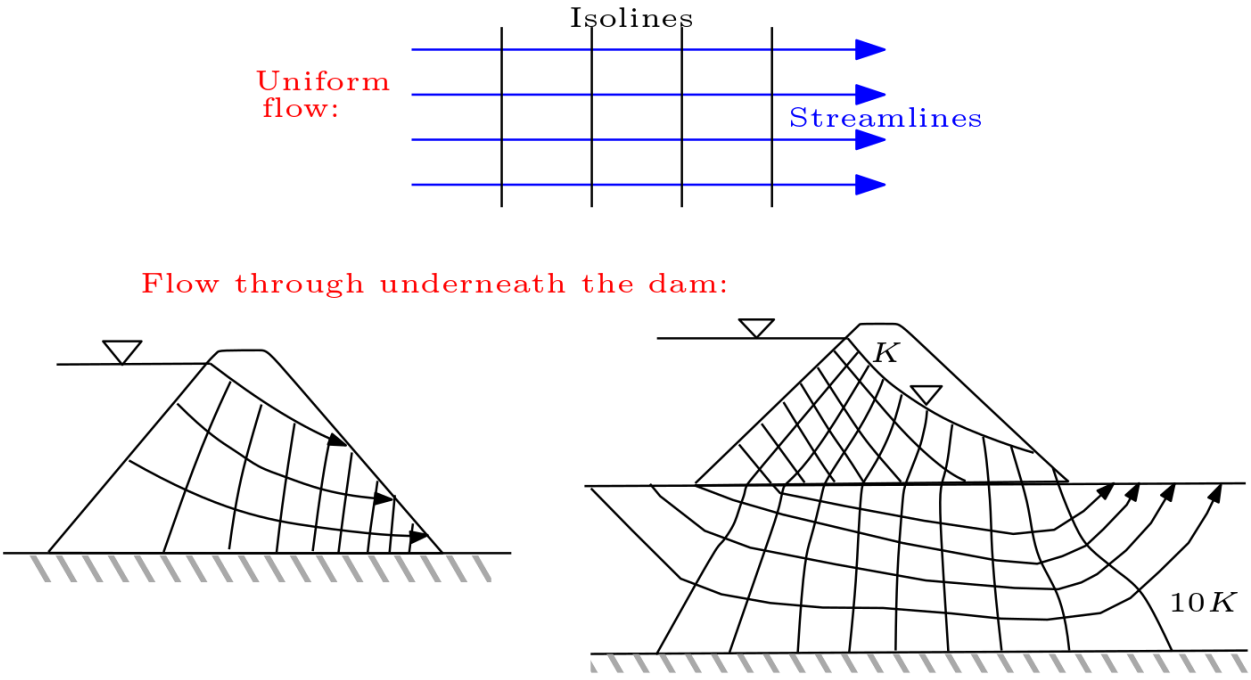


Fig. 2.13: Flow nets

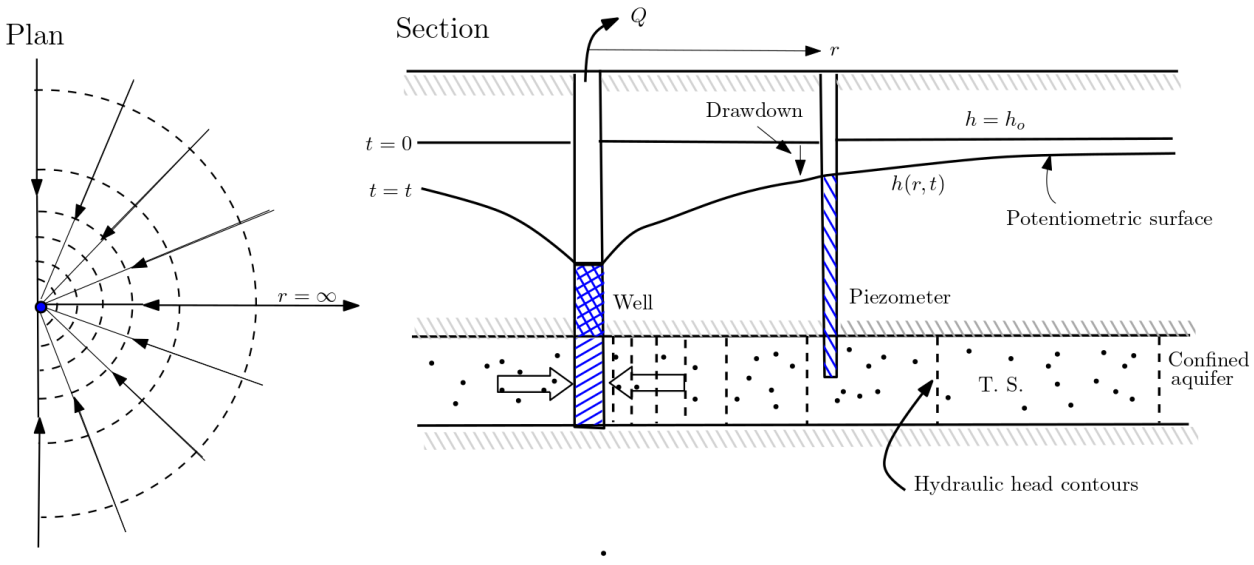


Fig. 2.14: Radial flow near a well

Superposition of Uniform and Radial Flow

The **dividing streamline** (solid black line) represents the boundary of the **well capture zone**. The flow velocities of the uniform flow and the radial flow exactly compensate each other at the **stagnation point.** The resulting flow velocity equal zero.

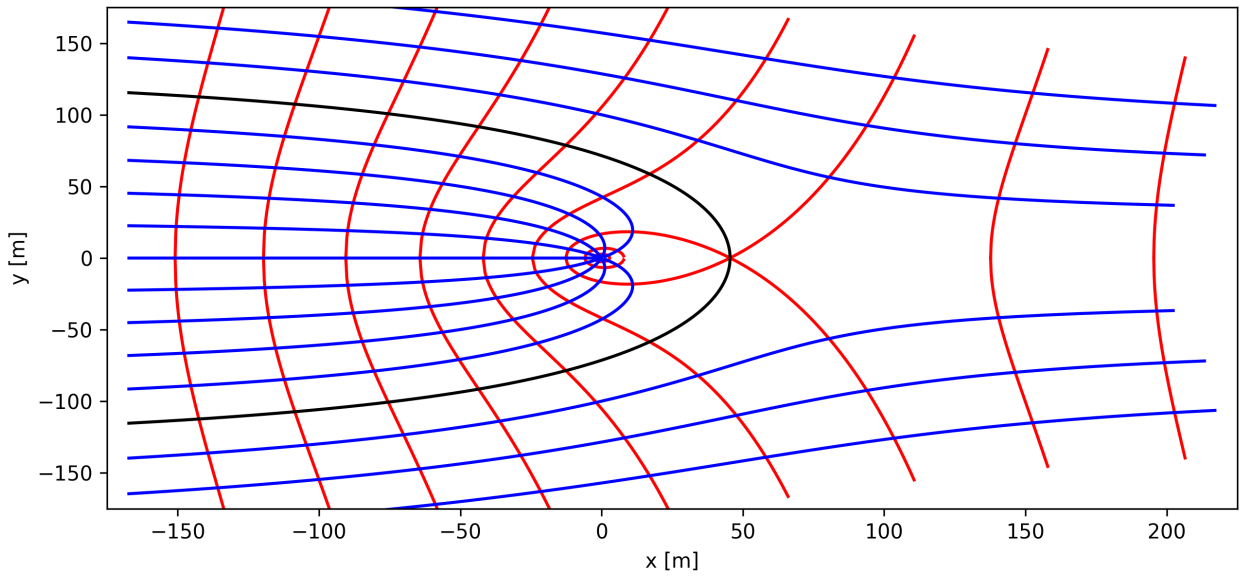


Fig. 2.15: Superposition of Radial flow near a well

Some Rules for Drawing Flow Nets

Drawing a flow net for a certain domain requires information about domain boundaries.

Impermeable boundaries represents a *streamline.*

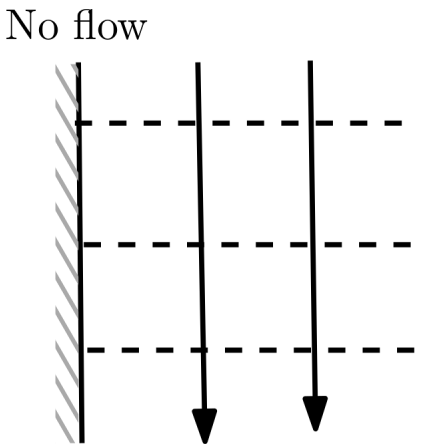


Fig. 2.16: No flow boundary

Constant head boundaries with given head values represents *isolines.*

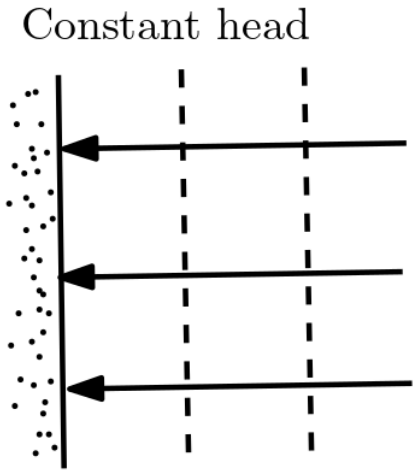


Fig. 2.17: Constant head boundary

Water table with and without evapotranspiration and recharge: Isolines and streamlines are drawn only for the saturated zone, i.e., they do not cross the water table.

Isolines do not intersect each other. Streamlines do not intersect each other.

Streamlines are never closed (**circular**). They start at an inflow boundary and end at an outflow boundary.

Adjacent isolines and streamlines should form **curvilinear squares**.

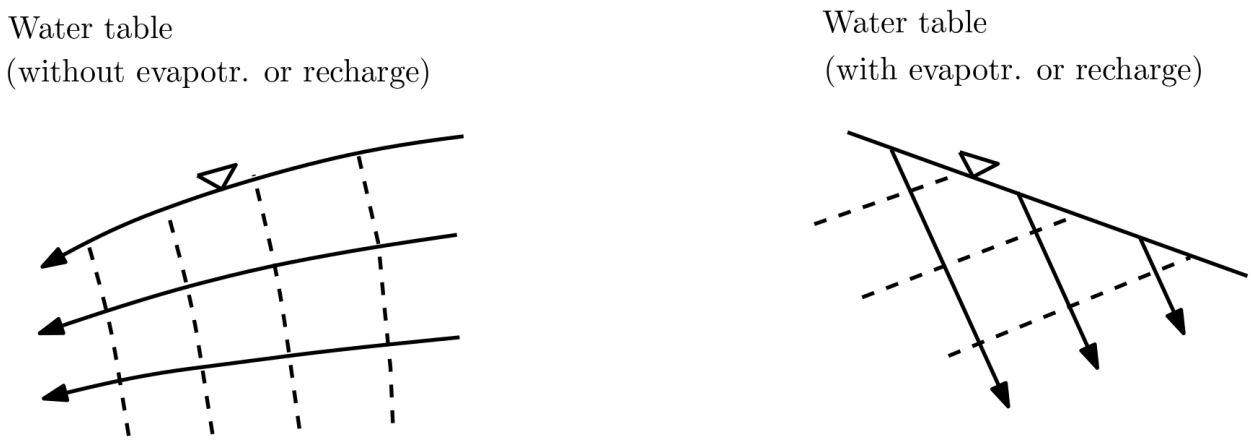


Fig. 2.18: Water table as boundary

Example problem

Flow parallel to layering

Considering $K = 6$ cm/s and $w = 50$cm, the width normal to plane, find the total discharge from the flow net provided in figure below: (source: https://gw-project.org/)

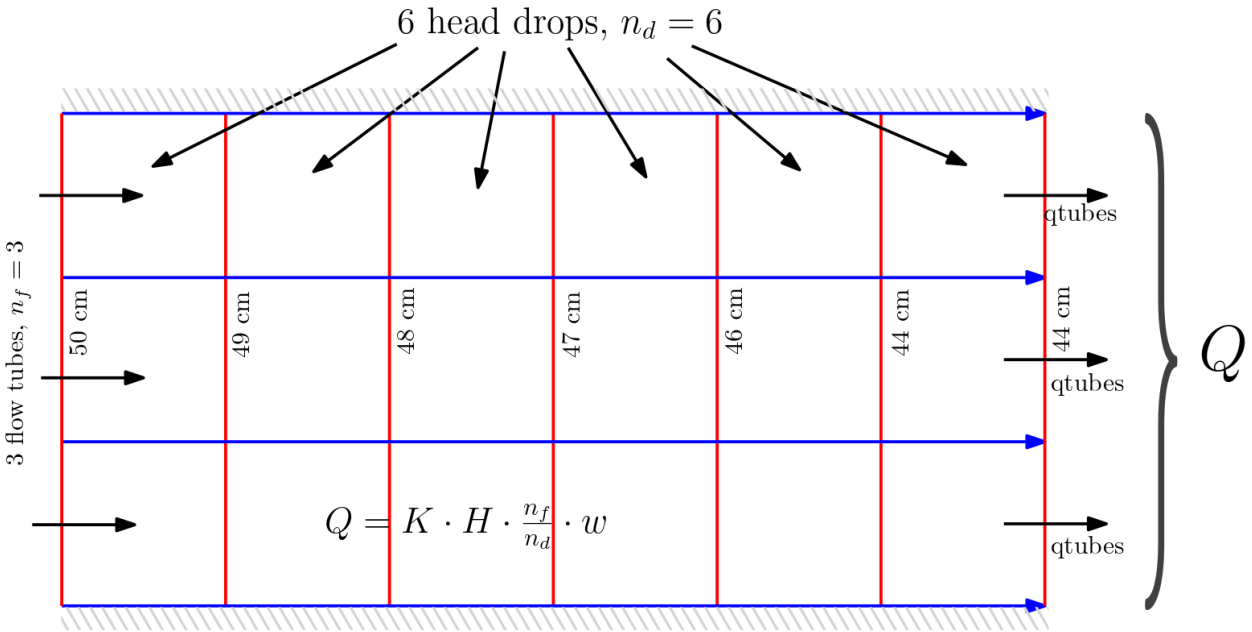


Fig. 2.19: Example problem 2

```
### solution

n_f = 3 # numer of flow tubes
n_d = 6 # number of head drops in the flow net
```

(continues on next page)

(continued from previous page)

```
w = 50 # cm, width lateral to the plane
K = 0.4 # cm/s, conductivity
h_in = 50 # cm, head inlet
h_out = 44 # cm, head outlet

#interim calculation
H = h_in-h_out

Q_t = K*H*n_f/n_d*w # cm^3/s, the total discharge

print("The total discharge out of domain is {0:0.0f}".format(Q_t), "cm\u00b3/s")
```

```
The total discharge out of domain is 60 cm³/s
```

2.7.7 Isochrones and Protection Zones.

Ischrones

The **isochrones** are curves of identical travel times. This is analogous to head isolines, in which the curves provides point of identical head. Similar to isolines, isochrones and streamlines do not *necessarily* intersect each other at an angle of 90°, i.e., curvilinear intersection between these curves are also possible.Isochrones are mostly used for delineating water protection zones. These zones are mostly part of the local water-use regulations.

```
import numpy as np
from ipywidgets import *
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

#definition of the function
def uniform_flow(K, ne, m, v, Q, t1, t2, t3):

    #intermediate results
    K_1 = K*86400                       #hydraulic conductivity [m/d]
    capture_width = Q/m/v               #capture width [m]
    L_ref = Q/(2*np.pi*np.exp(1)*m*v)   #[m]
    h_ref = v/K_1*L_ref                 #[m]
    t_ref = 0.5*ne*Q/np.pi/m/v**2       #[d]
    stagnation_x = np.exp(1)*L_ref      #stagnation point (x) [m]
    stagnation_y = 0                    #stagnation point (y) [m]

    #isolines; Syntax: isolines_[X or Y]_plot_[h_ref]_[optional: 1 or 2]
    isolines_x_plot_n5_1=[]
    isolines_y_plot_n5_1=[]
    isolines_x_plot_n5_2 = []
    isolines_y_plot_n5_2 = []
    isolines_x_plot_n2_5_1 = []
    isolines_y_plot_n2_5_1 = []
    isolines_x_plot_n2_5_2 = []
    isolines_y_plot_n2_5_2 = []
    isolines_x_plot_0_1 = []
    isolines_y_plot_0_1 = []
    isolines_x_plot_0_2 = []
```

(continues on next page)

(continued from previous page)

```
    isolines_y_plot_0_2 = []
    isolines_x_plot_2_5 = []
    isolines_y_plot_2_5 = []
    isolines_x_plot_5 = []
    isolines_y_plot_5 = []
    isolines_x_plot_7_5 = []
    isolines_y_plot_7_5 = []
    isolines_x_plot_10 = []
    isolines_y_plot_10 = []
    isolines_x_plot_12_5 = []
    isolines_y_plot_12_5 = []
    isolines_x_plot_15 = []
    isolines_y_plot_15 = []


    for x in range(0, 100):
        isolines_x_n5_1=L_ref*((x*0.169103048517306+(100-x)*-0.150360933444141)/100)
        isolines_x_n5_2=L_ref*((x*12.35+(100-x)*11.6815653622516)/100)
        isolines_x_n2_5_1=L_ref*((x*0.474722923528955+(100-x)*-0.350418198256065)/100)
        isolines_x_n2_5_2=L_ref*((x*9.45+(100-x)*8.22933315122817)/100)
        isolines_x_0_1=L_ref*((x*np.exp(1)+(100-x)*-0.75695357132717)/100)
        isolines_x_0_2=L_ref*((x*6.65+(100-x)*np.exp(1))/100)
        isolines_x_2_5=L_ref*((x*4+(100-x)*-1.46395392968976)/100)
        isolines_x_5=L_ref*((x*1.5+(100-x)*-2.50444142220744)/100)
        isolines_x_7_5=L_ref*((x*-0.875+(100-x)*-3.84154019983304)/100)
        isolines_x_10=L_ref*((x*-3.15+(100-x)*-5.4105773228373)/100)
        isolines_x_12_5=L_ref*((x*-5.4+(100-x)*-7.15205676143326)/100)
        isolines_x_15=L_ref*((x*-7.65+(100-x)*-9.02099613666581)/100)

        if x == 0:
            isolines_y_n5_1 = 0
            isolines_y_n5_2 = 0
            isolines_y_n2_5_1 = 0
            isolines_y_n2_5_2 = 0
            isolines_y_0_1 = 0
            isolines_y_0_2 = 0
            isolines_y_2_5 = 0
            isolines_y_5 = 0
            isolines_y_7_5 = 0
            isolines_y_10 = 0
            isolines_y_12_5 = 0
            isolines_y_15 = 0

        else:
            isolines_y_n5_1 = np.sqrt((L_ref*np.exp(-5/np.exp(1)+isolines_x_n5_1/L_
↪ref/np.exp(1)))**2-isolines_x_n5_1**2)
            isolines_y_n5_2 = np.sqrt((L_ref*np.exp(-5/np.exp(1)+isolines_x_n5_2/L_
↪ref/np.exp(1)))**2-isolines_x_n5_2**2)
            isolines_y_n2_5_1 = np.sqrt((L_ref*np.exp(-2.5/np.exp(1)+isolines_x_n2_5_
↪1/L_ref/np.exp(1)))**2-isolines_x_n2_5_1**2)
            isolines_y_n2_5_2 = np.sqrt((L_ref*np.exp(-2.5/np.exp(1)+isolines_x_n2_5_
↪2/L_ref/np.exp(1)))**2-isolines_x_n2_5_2**2)
            isolines_y_0_1 = np.sqrt((L_ref*np.exp(0/np.exp(1)+isolines_x_0_1/L_ref/
↪np.exp(1)))**2-isolines_x_0_1**2)
            isolines_y_0_2 = np.sqrt((L_ref*np.exp(0/np.exp(1)+isolines_x_0_2/L_ref/
↪np.exp(1)))**2-isolines_x_0_2**2)
            isolines_y_2_5 = np.sqrt((L_ref*np.exp(2.5/np.exp(1)+isolines_x_2_5/L_ref/
↪np.exp(1)))**2-isolines_x_2_5**2)
```

(continues on next page)

(continued from previous page)

```
            isolines_y_5 = np.sqrt((L_ref*np.exp(5/np.exp(1)+isolines_x_5/L_ref/np.
↪exp(1)))**2-isolines_x_5**2)
            isolines_y_7_5 = np.sqrt((L_ref*np.exp(7.5/np.exp(1)+isolines_x_7_5/L_ref/
↪np.exp(1)))**2-isolines_x_7_5**2)
            isolines_y_10 = np.sqrt((L_ref*np.exp(10/np.exp(1)+isolines_x_10/L_ref/np.
↪exp(1)))**2-isolines_x_10**2)
            isolines_y_12_5 = np.sqrt((L_ref*np.exp(12.5/np.exp(1)+isolines_x_12_5/L_
↪ref/np.exp(1)))**2-isolines_x_12_5**2)
            isolines_y_15 = np.sqrt((L_ref*np.exp(15/np.exp(1)+isolines_x_15/L_ref/np.
↪exp(1)))**2-isolines_x_15**2)

        isolines_x_plot_n5_1.append(isolines_x_n5_1)
        isolines_y_plot_n5_1.append(isolines_y_n5_1)
        isolines_x_plot_n5_2.append(isolines_x_n5_2)
        isolines_y_plot_n5_2.append(isolines_y_n5_2)
        isolines_x_plot_n2_5_1.append(isolines_x_n2_5_1)
        isolines_y_plot_n2_5_1.append(isolines_y_n2_5_1)
        isolines_x_plot_n2_5_2.append(isolines_x_n2_5_2)
        isolines_y_plot_n2_5_2.append(isolines_y_n2_5_2)
        isolines_x_plot_0_1.append(isolines_x_0_1)
        isolines_y_plot_0_1.append(isolines_y_0_1)
        isolines_x_plot_0_2.append(isolines_x_0_2)
        isolines_y_plot_0_2.append(isolines_y_0_2)
        isolines_x_plot_2_5.append(isolines_x_2_5)
        isolines_y_plot_2_5.append(isolines_y_2_5)
        isolines_x_plot_5.append(isolines_x_5)
        isolines_y_plot_5.append(isolines_y_5)
        isolines_x_plot_7_5.append(isolines_x_7_5)
        isolines_y_plot_7_5.append(isolines_y_7_5)
        isolines_x_plot_10.append(isolines_x_10)
        isolines_y_plot_10.append(isolines_y_10)
        isolines_x_plot_12_5.append(isolines_x_12_5)
        isolines_y_plot_12_5.append(isolines_y_12_5)
        isolines_x_plot_15.append(isolines_x_15)
        isolines_y_plot_15.append(isolines_y_15)

    #streamlines; synthax: streamlines_[X or Y]_plot_[psi]
    streamlines_x_plot_0 = [0, (L_ref*-10)]
    streamlines_y_plot_0 = [0, 0]
    streamlines_x_plot_0_2 = []
    streamlines_y_plot_0_2 = []
    streamlines_x_plot_0_4 = []
    streamlines_y_plot_0_4 = []
    streamlines_x_plot_0_6 = []
    streamlines_y_plot_0_6 = []
    streamlines_x_plot_0_8 = []
    streamlines_y_plot_0_8 = []
    streamlines_x_plot_1 = []
    streamlines_y_plot_1 = []
    streamlines_x_plot_1_2 = []
    streamlines_y_plot_1_2 = []
    streamlines_x_plot_1_4 = []
    streamlines_y_plot_1_4 = []
    streamlines_x_plot_1_6 = []
    streamlines_y_plot_1_6 = []

    for x in range(0,100):
```

(continues on next page)

(continued from previous page)

```
        streamlines_y_0_2 = L_ref*((x*0+(100-x)*1.34462005667342)/100)
        streamlines_y_0_4 = L_ref*((x*0+(100-x)*2.6992421745751)/100)
        streamlines_y_0_6 = L_ref*((x*0+(100-x)*4.07255559164565)/100)
        streamlines_y_0_8 = L_ref*((x*0+(100-x)*5.47097889806004)/100)
        streamlines_y_1 = L_ref*((x*0+(100-x)*6.89826117541355)/100)
        streamlines_y_1_2 = L_ref*((x*2.13413758353342+(100-x)*8.35561532789609)/100)
        streamlines_y_1_4 = L_ref*((x*4.24381382643103+(100-x)*9.8422946888304)/100)
        streamlines_y_1_6 = L_ref*((x*6.31283612436048+(100-x)*11.3562442221618)/100)

        streamlines_x_0_2 = streamlines_y_0_2/np.tan(streamlines_y_0_2/L_ref/np.
↪exp(1)-np.pi*0.2)
        streamlines_x_0_4 = streamlines_y_0_4/np.tan(streamlines_y_0_4/L_ref/np.
↪exp(1)-np.pi*0.4)
        streamlines_x_0_6 = streamlines_y_0_6/np.tan(streamlines_y_0_6/L_ref/np.
↪exp(1)-np.pi*0.6)
        streamlines_x_0_8 = streamlines_y_0_8/np.tan(streamlines_y_0_8/L_ref/np.
↪exp(1)-np.pi*0.8)
        streamlines_x_1 = streamlines_y_1/np.tan(streamlines_y_1/L_ref/np.exp(1)-np.
↪pi*1)
        streamlines_x_1_2 = streamlines_y_1_2/np.tan(streamlines_y_1_2/L_ref/np.
↪exp(1)-np.pi*1.2)
        streamlines_x_1_4 = streamlines_y_1_4/np.tan(streamlines_y_1_4/L_ref/np.
↪exp(1)-np.pi*1.4)
        streamlines_x_1_6 = streamlines_y_1_6/np.tan(streamlines_y_1_6/L_ref/np.
↪exp(1)-np.pi*1.6)

        streamlines_x_plot_0_2.append(streamlines_x_0_2)
        streamlines_y_plot_0_2.append(streamlines_y_0_2)
        streamlines_x_plot_0_4.append(streamlines_x_0_4)
        streamlines_y_plot_0_4.append(streamlines_y_0_4)
        streamlines_x_plot_0_6.append(streamlines_x_0_6)
        streamlines_y_plot_0_6.append(streamlines_y_0_6)
        streamlines_x_plot_0_8.append(streamlines_x_0_8)
        streamlines_y_plot_0_8.append(streamlines_y_0_8)
        streamlines_x_plot_1.append(streamlines_x_1)
        streamlines_y_plot_1.append(streamlines_y_1)
        streamlines_x_plot_1_2.append(streamlines_x_1_2)
        streamlines_y_plot_1_2.append(streamlines_y_1_2)
        streamlines_x_plot_1_4.append(streamlines_x_1_4)
        streamlines_y_plot_1_4.append(streamlines_y_1_4)
        streamlines_x_plot_1_6.append(streamlines_x_1_6)
        streamlines_y_plot_1_6.append(streamlines_y_1_6)

    #isochrones
    isochrones_x_plot_t1 = []
    isochrones_y_plot_t1 = []
    isochrones_x_plot_t2 = []
    isochrones_y_plot_t2 = []
    isochrones_x_plot_t3 = []
    isochrones_y_plot_t3 = []

    #iterate 5 times with start value t_xmin1/ t_xmax1

    t1_xmin1=-np.exp(1)*np.sqrt(np.exp(2*(t1/t_ref))-1)
    t1_xmin6= t1_xmin1+(np.exp(1)-t1_xmin1)*(1+np.exp(1)/t1_xmin1*(np.log(1-t1_xmin1/
↪np.exp(1))+(t1/t_ref)))
    t1_xmax1= np.exp(1)*np.sqrt(1-np.exp(-2*(t1/t_ref)))
```

(continues on next page)

(continued from previous page)

```
    t1_xmax6= t1_xmax1+(np.exp(1)-t1_xmax1)*(1+np.exp(1)/t1_xmax1*(np.log(1-t1_xmax1/
↪np.exp(1))+(t1/t_ref)))
    t2_xmin1=-np.exp(1)*np.sqrt(np.exp(2*(t2/t_ref))-1)
    t2_xmin6= t2_xmin1+(np.exp(1)-t2_xmin1)*(1+np.exp(1)/t2_xmin1*(np.log(1-t2_xmin1/
↪np.exp(1))+(t2/t_ref)))
    t2_xmax1=np.exp(1)*np.sqrt(1-np.exp(-2*(t2/t_ref)))
    t2_xmax6= t2_xmax1+(np.exp(1)-t2_xmax1)*(1+np.exp(1)/t2_xmax1*(np.log(1-t2_xmax1/
↪np.exp(1))+(t2/t_ref)))
    t3_xmin1=-np.exp(1)*np.sqrt(np.exp(2*(t3/t_ref))-1)
    t3_xmin6= t3_xmin1+(np.exp(1)-t3_xmin1)*(1+np.exp(1)/t3_xmin1*(np.log(1-t3_xmin1/
↪np.exp(1))+(t3/t_ref)))
    t3_xmax1=np.exp(1)*np.sqrt(1-np.exp(-2*(t3/t_ref)))
    t3_xmax6= t3_xmax1+(np.exp(1)-t3_xmax1)*(1+np.exp(1)/t3_xmax1*(np.log(1-t3_xmax1/
↪np.exp(1))+(t3/t_ref)))
    for i in range(4):
        t1_xmin6= t1_xmin6+(np.exp(1)-t1_xmin6)*(1+np.exp(1)/t1_xmin6*(np.log(1-t1_
↪xmin6/np.exp(1))+(t1/t_ref)))
        t1_xmax6= t1_xmax6+(np.exp(1)-t1_xmax6)*(1+np.exp(1)/t1_xmax6*(np.log(1-t1_
↪xmax6/np.exp(1))+(t1/t_ref)))
        t2_xmin6= t2_xmin6+(np.exp(1)-t2_xmin6)*(1+np.exp(1)/t2_xmin6*(np.log(1-t2_
↪xmin6/np.exp(1))+(t2/t_ref)))
        t2_xmax6= t2_xmax6+(np.exp(1)-t2_xmax6)*(1+np.exp(1)/t2_xmax6*(np.log(1-t2_
↪xmax6/np.exp(1))+(t2/t_ref)))
        t3_xmin6= t3_xmin6+(np.exp(1)-t3_xmin6)*(1+np.exp(1)/t3_xmin6*(np.log(1-t3_
↪xmin6/np.exp(1))+(t3/t_ref)))
        t3_xmax6= t3_xmax6+(np.exp(1)-t3_xmax6)*(1+np.exp(1)/t3_xmax6*(np.log(1-t3_
↪xmax6/np.exp(1))+(t3/t_ref)))

    for x in range (0,100):

        isochrones_x_t1 = 0.5*L_ref*(t1_xmin6+t1_xmax6+(t1_xmax6-t1_xmin6)*np.cos(np.
↪pi*(100-x)/100))
        isochrones_x_t2 = 0.5*L_ref*(t2_xmin6+t2_xmax6+(t2_xmax6-t2_xmin6)*np.cos(np.
↪pi*(100-x)/100))
        isochrones_x_t3 = 0.5*L_ref*(t3_xmin6+t3_xmax6+(t3_xmax6-t3_xmin6)*np.cos(np.
↪pi*(100-x)/100))

        if x == 0:
            isochrones_y_t1 = 0
            isochrones_y_t2 = 0
            isochrones_y_t3 = 0
        else:

            isochrones_y1_t1 = L_ref*np.exp(1)*np.arccos((0.5*isochrones_x_t1/np.
↪exp(1)/L_ref+np.exp(-(t1/t_ref)-isochrones_x_t1/np.exp(1)/L_ref))/(1-0.5*isochrones_
↪x_t1/np.exp(1)/L_ref))
            isochrones_y_t1 = L_ref*np.exp(1)*np.arccos((isochrones_x_t1/L_ref*(np.
↪sin(isochrones_y1_t1/np.exp(1)/L_ref)/(isochrones_y1_t1/L_ref)-0.5*np.
↪cos(isochrones_y1_t1/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t1/t_ref)-isochrones_x_t1/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t1/np.exp(1)/L_ref))

            isochrones_y1_t2 = L_ref*np.exp(1)*np.arccos((0.5*isochrones_x_t2/np.
↪exp(1)/L_ref+np.exp(-(t2/t_ref)-isochrones_x_t2/np.exp(1)/L_ref))/(1-0.5*isochrones_
↪x_t2/np.exp(1)/L_ref))
            isochrones_y_t2 = L_ref*np.exp(1)*np.arccos((isochrones_x_t2/L_ref*(np.
↪sin(isochrones_y1_t2/np.exp(1)/L_ref)/(isochrones_y1_t2/L_ref)-0.5*np.
↪cos(isochrones_y1_t2/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t2/t_ref)-isochrones_x_t2/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t2/np.exp(1)/L_ref))
```

(continues on next page)

(continued from previous page)

```
            isochrones_y1_t3 = L_ref*np.exp(1)*np.arccos((0.5*isochrones_x_t3/np.
↪exp(1)/L_ref+np.exp(-(t3/t_ref)-isochrones_x_t3/np.exp(1)/L_ref))/(1-0.5*isochrones_
↪x_t3/np.exp(1)/L_ref))
            isochrones_y_t3 = L_ref*np.exp(1)*np.arccos((isochrones_x_t3/L_ref*(np.
↪sin(isochrones_y1_t3/np.exp(1)/L_ref)/(isochrones_y_t3/L_ref)-0.5*np.cos(isochrones_
↪y1_t3/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t3/t_ref)-isochrones_x_t3/np.exp(1)/L_
↪ref))/(1-0.5*isochrones_x_t3/np.exp(1)/L_ref))

            for i in range(4):
                isochrones_y_t1 = L_ref*np.exp(1)*np.arccos((isochrones_x_t1/L_
↪ref*(np.sin(isochrones_y_t1/np.exp(1)/L_ref)/(isochrones_y_t1/L_ref)-0.5*np.
↪cos(isochrones_y_t1/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t1/t_ref)-isochrones_x_t1/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t1/np.exp(1)/L_ref))

                isochrones_y_t2 = L_ref*np.exp(1)*np.arccos((isochrones_x_t2/L_
↪ref*(np.sin(isochrones_y_t2/np.exp(1)/L_ref)/(isochrones_y_t2/L_ref)-0.5*np.
↪cos(isochrones_y_t2/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t2/t_ref)-isochrones_x_t2/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t2/np.exp(1)/L_ref))

                isochrones_y_t3 = L_ref*np.exp(1)*np.arccos((isochrones_x_t3/L_
↪ref*(np.sin(isochrones_y_t3/np.exp(1)/L_ref)/(isochrones_y_t3/L_ref)-0.5*np.
↪cos(isochrones_y_t3/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t3/t_ref)-isochrones_x_t3/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t3/np.exp(1)/L_ref))


        isochrones_x_plot_t1.append(isochrones_x_t1)
        isochrones_y_plot_t1.append(isochrones_y_t1)
        isochrones_x_plot_t2.append(isochrones_x_t2)
        isochrones_y_plot_t2.append(isochrones_y_t2)
        isochrones_x_plot_t3.append(isochrones_x_t3)
        isochrones_y_plot_t3.append(isochrones_y_t3)

    #still necessary: mirror on x-axis
    isolines_y_plot_n5_1_mirror = -1*(np.asarray(isolines_y_plot_n5_1))
    isolines_y_plot_n5_2_mirror = -1*(np.asarray(isolines_y_plot_n5_2))
    isolines_y_plot_n2_5_1_mirror = -1*(np.asarray(isolines_y_plot_n2_5_1))
    isolines_y_plot_n2_5_2_mirror = -1*(np.asarray(isolines_y_plot_n2_5_2))
    isolines_y_plot_0_1_mirror = -1*(np.asarray(isolines_y_plot_0_1))
    isolines_y_plot_0_2_mirror = -1*(np.asarray(isolines_y_plot_0_2))
    isolines_y_plot_2_5_mirror = -1*(np.asarray(isolines_y_plot_2_5))
    isolines_y_plot_5_mirror = -1*(np.asarray(isolines_y_plot_5))
    isolines_y_plot_7_5_mirror = -1*(np.asarray(isolines_y_plot_7_5))
    isolines_y_plot_10_mirror = -1*(np.asarray(isolines_y_plot_10))
    isolines_y_plot_12_5_mirror = -1*(np.asarray(isolines_y_plot_12_5))
    isolines_y_plot_15_mirror = -1*(np.asarray(isolines_y_plot_15))

    streamlines_y_plot_0_2_mirror = -1*(np.asarray(streamlines_y_plot_0_2))
    streamlines_y_plot_0_4_mirror = -1*(np.asarray(streamlines_y_plot_0_4))
    streamlines_y_plot_0_6_mirror = -1*(np.asarray(streamlines_y_plot_0_6))
    streamlines_y_plot_0_8_mirror = -1*(np.asarray(streamlines_y_plot_0_8))
    streamlines_y_plot_1_mirror = -1*(np.asarray(streamlines_y_plot_1))
    streamlines_y_plot_1_2_mirror = -1*(np.asarray(streamlines_y_plot_1_2))
    streamlines_y_plot_1_4_mirror = -1*(np.asarray(streamlines_y_plot_1_4))
    streamlines_y_plot_1_6_mirror = -1*(np.asarray(streamlines_y_plot_1_6))
```

(continues on next page)

(continued from previous page)

```
    isochrones_y_plot_t1_mirror = -1*(np.asarray(isochrones_y_plot_t1))
    isochrones_y_plot_t2_mirror = -1*(np.asarray(isochrones_y_plot_t2))
    isochrones_y_plot_t3_mirror = -1*(np.asarray(isochrones_y_plot_t3))


    fig, (ax1, ax2) = plt.subplots(2, figsize=(10, 10))

    #plotten incl. mirror on x-axis

    ax1.plot(isolines_x_plot_n5_1, isolines_y_plot_n5_1, 'r')
    ax1.plot(isolines_x_plot_n5_2, isolines_y_plot_n5_2, 'r')
    ax1.plot(isolines_x_plot_n2_5_1, isolines_y_plot_n2_5_1, 'r')
    ax1.plot(isolines_x_plot_n2_5_2, isolines_y_plot_n2_5_2, 'r')
    ax1.plot(isolines_x_plot_0_1, isolines_y_plot_0_1, 'r')
    ax1.plot(isolines_x_plot_0_2, isolines_y_plot_0_2, 'r')
    ax1.plot(isolines_x_plot_2_5, isolines_y_plot_2_5, 'r')
    ax1.plot(isolines_x_plot_5, isolines_y_plot_5, 'r')
    ax1.plot(isolines_x_plot_7_5, isolines_y_plot_7_5, 'r')
    ax1.plot(isolines_x_plot_10, isolines_y_plot_10, 'r')
    ax1.plot(isolines_x_plot_12_5, isolines_y_plot_12_5, 'r')
    ax1.plot(isolines_x_plot_15, isolines_y_plot_15, 'r')

    ax1.plot(isolines_x_plot_n5_1, isolines_y_plot_n5_1_mirror, 'r')
    ax1.plot(isolines_x_plot_n5_2, isolines_y_plot_n5_2_mirror, 'r')
    ax1.plot(isolines_x_plot_n2_5_1, isolines_y_plot_n2_5_1_mirror, 'r')
    ax1.plot(isolines_x_plot_n2_5_2, isolines_y_plot_n2_5_2_mirror, 'r')
    ax1.plot(isolines_x_plot_0_1, isolines_y_plot_0_1_mirror, 'r')
    ax1.plot(isolines_x_plot_0_2, isolines_y_plot_0_2_mirror, 'r')
    ax1.plot(isolines_x_plot_2_5, isolines_y_plot_2_5_mirror, 'r')
    ax1.plot(isolines_x_plot_5, isolines_y_plot_5_mirror, 'r')
    ax1.plot(isolines_x_plot_7_5, isolines_y_plot_7_5_mirror, 'r')
    ax1.plot(isolines_x_plot_10, isolines_y_plot_10_mirror, 'r')
    ax1.plot(isolines_x_plot_12_5, isolines_y_plot_12_5_mirror, 'r')
    ax1.plot(isolines_x_plot_15, isolines_y_plot_15_mirror, 'r')

    ax1.plot(streamlines_x_plot_0,streamlines_y_plot_0, 'b')
    ax1.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2, 'b')
    ax1.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4, 'b')
    ax1.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6, 'b')
    ax1.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8, 'b')
    ax1.plot(streamlines_x_plot_1,streamlines_y_plot_1, color = 'black')
    ax1.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2, 'b')
    ax1.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4, 'b')
    ax1.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6, 'b')


    ax1.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2_mirror, 'b')
    ax1.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4_mirror, 'b')
    ax1.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6_mirror, 'b')
    ax1.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8_mirror, 'b')
    ax1.plot(streamlines_x_plot_1,streamlines_y_plot_1_mirror, color = 'black')
    ax1.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2_mirror, 'b')
    ax1.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4_mirror, 'b')
    ax1.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6_mirror, 'b')


    ax1.set(xlabel='x [m]', ylabel ='y [m]', xlim = [-175, 225], ylim = [-175,175])
```

(continues on next page)

(continued from previous page)

```
    fig.savefig("isolines.png", dpi=300)

    ax2.plot(isochrones_x_plot_t1, isochrones_y_plot_t1, 'g')
    ax2.plot(isochrones_x_plot_t2, isochrones_y_plot_t2, 'g')
    ax2.plot(isochrones_x_plot_t3, isochrones_y_plot_t3, 'g')

    ax2.plot(isochrones_x_plot_t1, isochrones_y_plot_t1_mirror, 'g')
    ax2.plot(isochrones_x_plot_t2, isochrones_y_plot_t2_mirror, 'g')
    ax2.plot(isochrones_x_plot_t3, isochrones_y_plot_t3_mirror, 'g')

    ax2.plot(streamlines_x_plot_0,streamlines_y_plot_0, 'b')
    ax2.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2, 'b')
    ax2.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4, 'b')
    ax2.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6, 'b')
    ax2.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8, 'b')
    ax2.plot(streamlines_x_plot_1,streamlines_y_plot_1, color = 'black')
    ax2.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2, 'b')
    ax2.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4, 'b')
    ax2.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6, 'b')


    ax2.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2_mirror, 'b')
    ax2.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4_mirror, 'b')
    ax2.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6_mirror, 'b')
    ax2.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8_mirror, 'b')
    ax2.plot(streamlines_x_plot_1,streamlines_y_plot_1_mirror, color = 'black')
    ax2.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2_mirror, 'b')
    ax2.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4_mirror, 'b')
    ax2.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6_mirror, 'b')



    ax2.set(xlabel='x [m]', ylabel ='y [m]', xlim = [-175, 225], ylim = [-175,175])
    fig.savefig("isochrones.png", dpi=300)

interact(uniform_flow,
         K=widgets.FloatLogSlider(value=3e-4, base=10, min=-10, max=0, step=0.1,␣
↪description='hydraulic conductivity [m/s]:', disabled=False),
         ne=widgets.FloatSlider(value=0.2, min=0.001, max=1, step=0.05, description=
↪'effective porosity [-]:', disabled=False),
         m= widgets.FloatSlider(value=7,min=0, max=30,step=1, description='thickness␣
↪[m]:' , disabled=False),
         v=widgets.FloatSlider(value=0.4, min=0.00001, max=5, step=0.1, description=
↪'uniform velocity [m/d]:', disabled=False),
         Q=widgets.FloatSlider(value=800, min=100, max=1000, step=0.1, description=
↪'pumping rate [m^3/d]:', disabled=False),
         t1=widgets.BoundedFloatText(value=10, min=1, max=100, step=0.1, description=
↪'t1 [d]:', disabled=False),
         t2=widgets.BoundedFloatText(value=30, min=1, max=100, step=0.1, description=
↪'t2 [d]:', disabled=False),
         t3=widgets.BoundedFloatText(value=50, min=1, max=100, step=0.1, description=
↪'t3 [d]:', disabled=False),
         )
```

```
interactive(children=(FloatLogSlider(value=0.0003, description='hydraulic␣
↪conductivity [m/s]:', max=0.0, min=-...
```

```
<function __main__.uniform_flow(K, ne, m, v, Q, t1, t2, t3)>
```

Example of Protection Zones

The figure below show a schematic of a well protection zone. The main goals of the protection zones are:

To protect aquifers against pollution

Simple and robust zones based on aquifer pollution vulnerability and source protection perimeters

A sensible balance between the protection of groundwater resources (aquifer as a whole) and specific sources (e.g., wells)

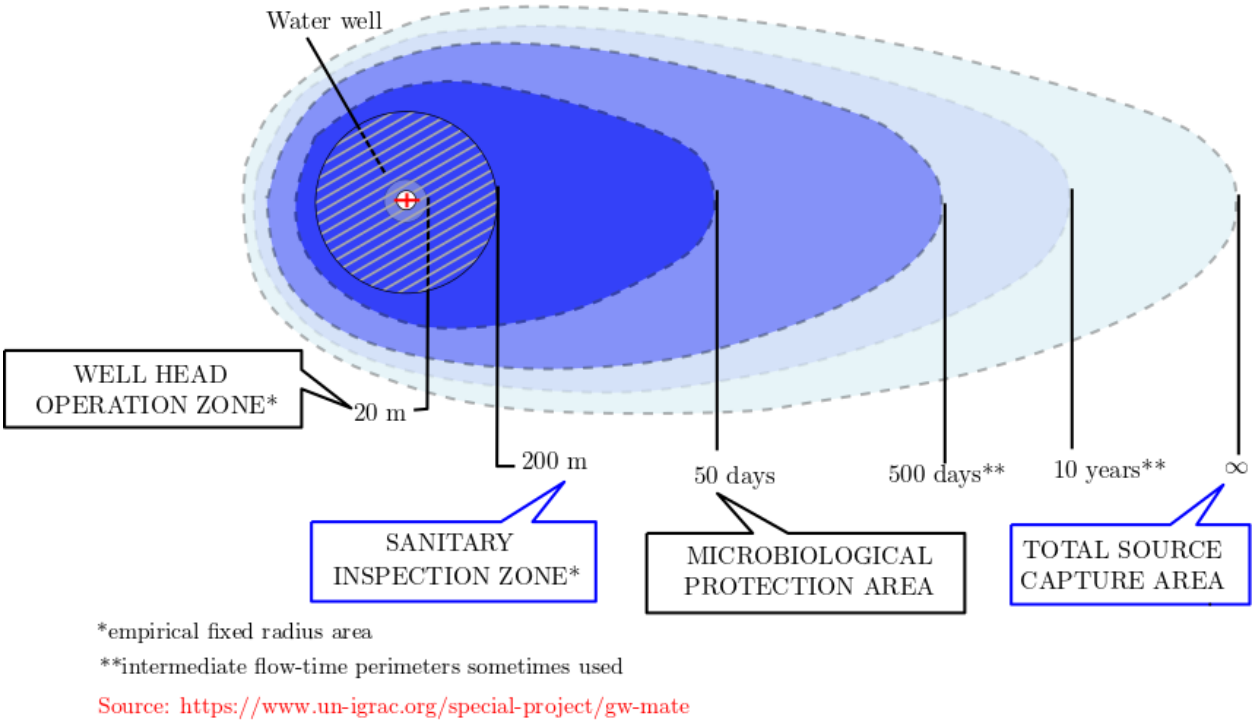


Fig. 2.20: A schematic of the groundwater protection zone.

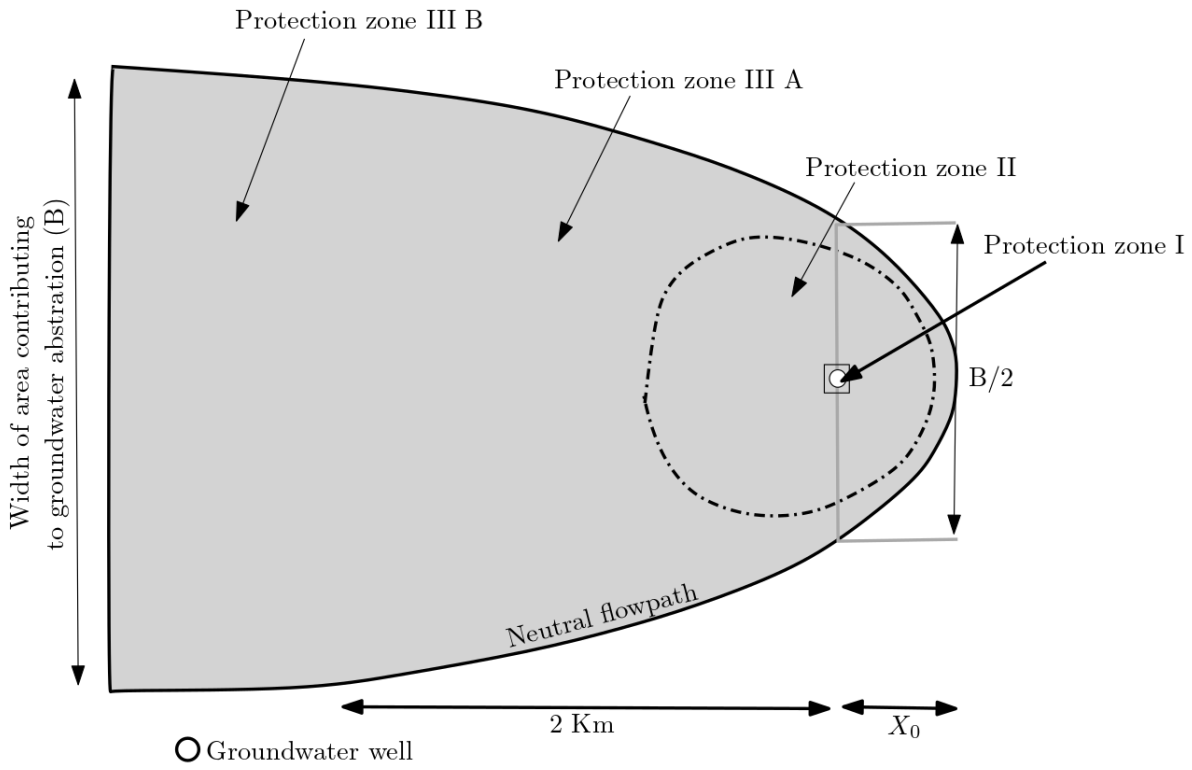
As can be observed in the figure, the zones are based on the travel time, and each travel time is divided into protection of groundwater resources against different activities.

In Germany law DVGW W-101 exitst for groundwater protection zones. As per the German rule, the groundwater protection zones is divided into three different zones:

Zone I : The *immediate* protection zone - at least 10 m from the water well, not less than 20 m in the upstream direction of a spring.

Zone II : The *inner* protection zone - 50-day travel time but not less than 100 m from the well or spring.

Zone III : The *outer* protection zone - entire contribution zone of the groundwater catchment area (maybe sub-divided see figure)



Source: https://www.geozentrum-hannover.de

Fig. 2.21: Groundwater protection zone in Germany

Well Capture Zones in Natural Aquifers

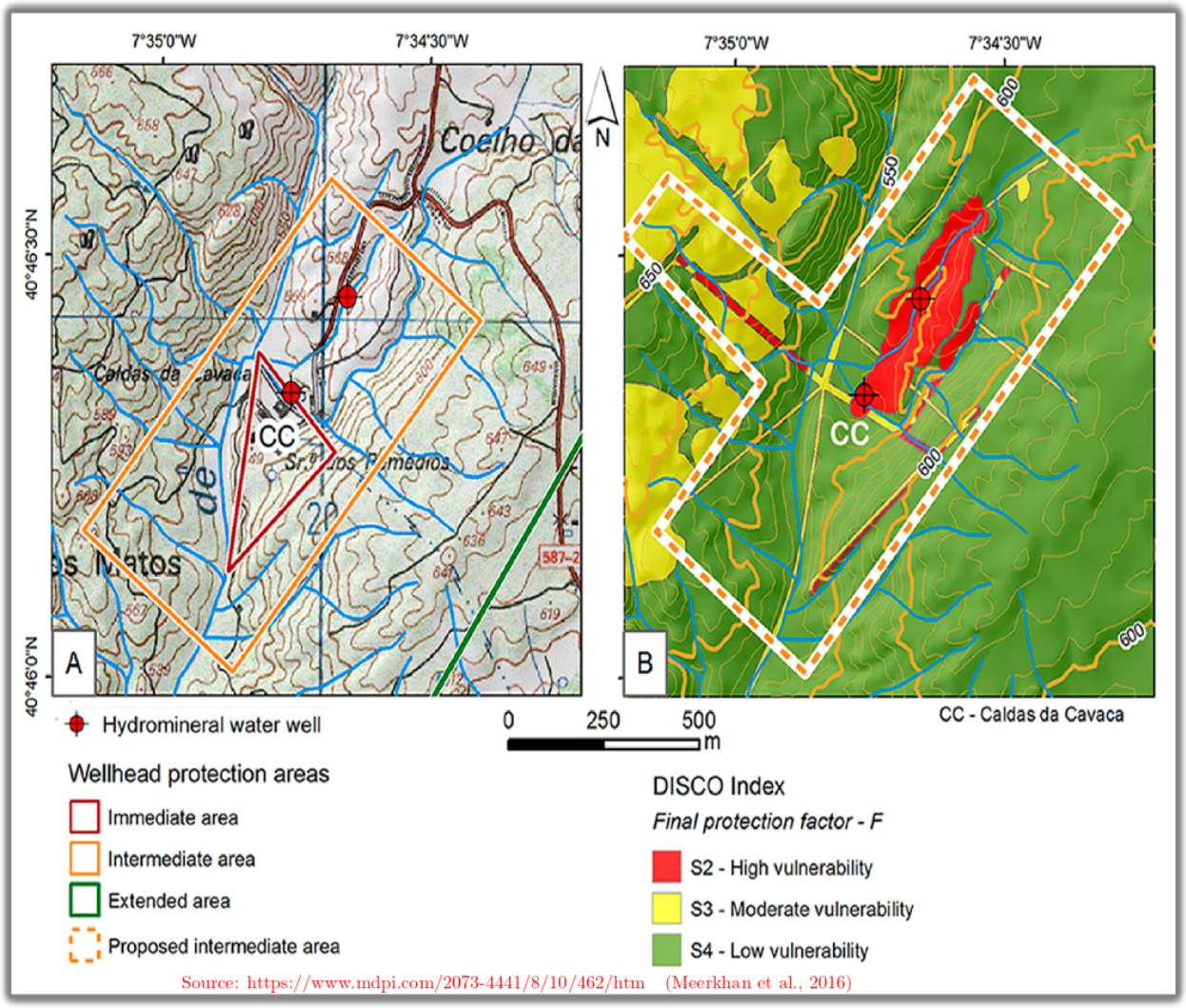


Fig. 2.22: Groundwater protection zone natural aquifers

2.7.8 Darcy's Law in Anisotropic

Principal Axes of Hydraulic Conductivity

In anisotropic aquifer refers to those aquifers whose properties vary with direction. This means that properties such as hydraulic conductivity will have maximum value along a direction and the minimum value along the other direction. It has been found that (also depicted in fig **:refnum:`axes_an`**) these two directions are perpendicular to each other. This is true in general, not only for the layered systems that was covered in the previous lecture

The three *principal axes* of hydraulic conductivity are oriented

- along the direction with maximum hydraulic conductivity
- along the direction with hydraulic conductivity
- perpendicular to both

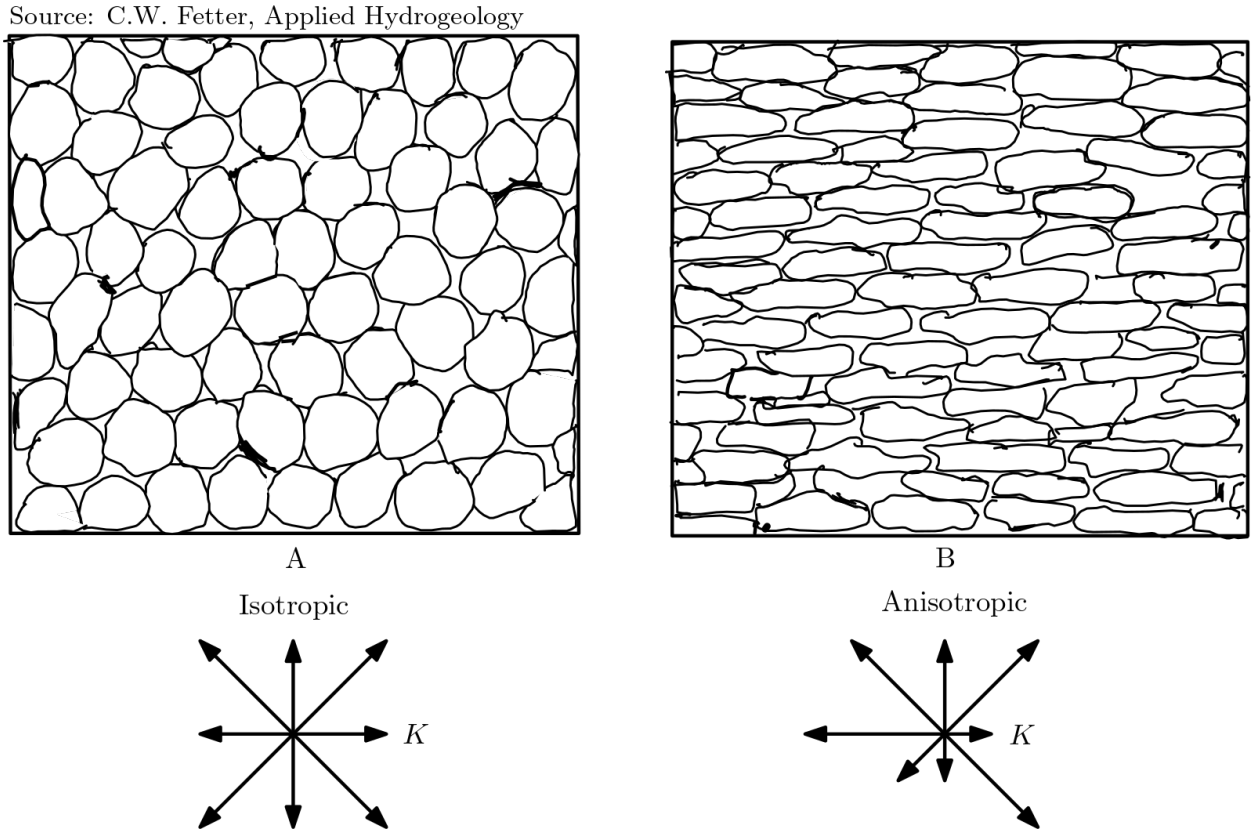


Fig. 2.23: Hydraulic conductivity in anisotropic aquifers

Formulation of Darcy's Law in Anisotropic Aquifers

If the *principal axes* of conductivity coincide with the axes of a Cartesian coordinate system, Darcy's law for anisotropic aquifer is given as

$$\begin{bmatrix} v_{fx} \\ v_{fy} \\ v_{fz} \end{bmatrix} = - \begin{bmatrix} K_x, 0, 0 \\ 0, K_y, 0 \\ 0, 0, K_z \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{bmatrix}$$

In this equation, hydraulic conductivity has to be written as a matrix. To be precise, K represents a **tensor**, i.e., a quantity which is subject to certain rules under coordinate transforms (from Cartesian to cylinder coordinates, for example). These rules guarantee that Darcy's law can also be applied in coordinate systems other than Cartesian.

If the aquifer is isotropic in the horizontally oriented xy-plane, we get,

$$\begin{bmatrix} v_{fx} \\ v_{fy} \\ v_{fz} \end{bmatrix} = - \begin{bmatrix} K_h, 0, 0 \\ 0, K_h, 0 \\ 0, 0, K_v \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{bmatrix}$$

The most general case is encountered when all *principal axes* are associated with different hydraulic conductivities and none of them coincides with a coordinate axis:

$$\begin{bmatrix} v_{fx} \\ v_{fy} \\ v_{fz} \end{bmatrix} = - \begin{bmatrix} K_{xx}, K_{xy}, K_{xz} \\ K_{xy}, K_{yy}, K_{yz} \\ K_{xz}, K_{yz}, K_{zz} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{bmatrix}$$

From this, one may easily conclude that it is advantageous to arrange coordinate axes in parallel with *principal axes* of K. Unfortunately, this is not always possible, e.g., when layers are folded.

Direction of flow

With directional dependence on conductivity also affects the direction of the flow. This is summarized in the figure below. Basically we have

Isotropic aquifer: K can be represented by a scalar and the direction of flow is opposite to the direction of the gradient.

Anisotropic aquifer: K has to be represented by a tensor (matrix) and the angle between the gradient vector and the flow direction is between 90° and 180°.

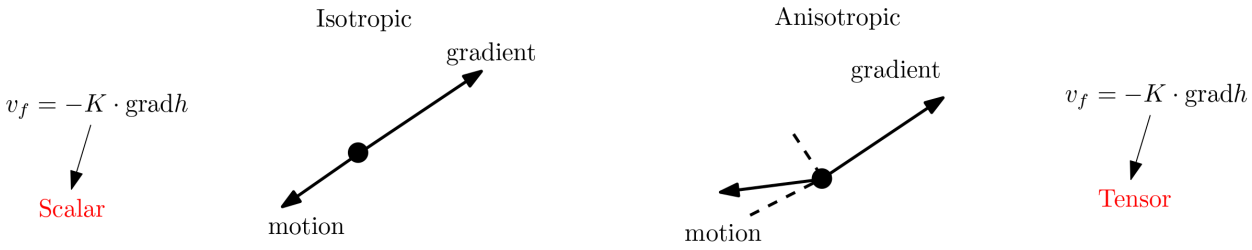


Fig. 2.24: Direction of flow in anisotropic aquifer

Examples for Flow Nets in Anisotropic Aquifers

The figure provides flow nets for different ratios of anisotropy but for the identical conditions at domain boundaries (i.e., inflow from the top, outflow to a pipe on the left). As can be observed isolines and streamlines intersect each other at right angles only if the aquifer is isotropic.

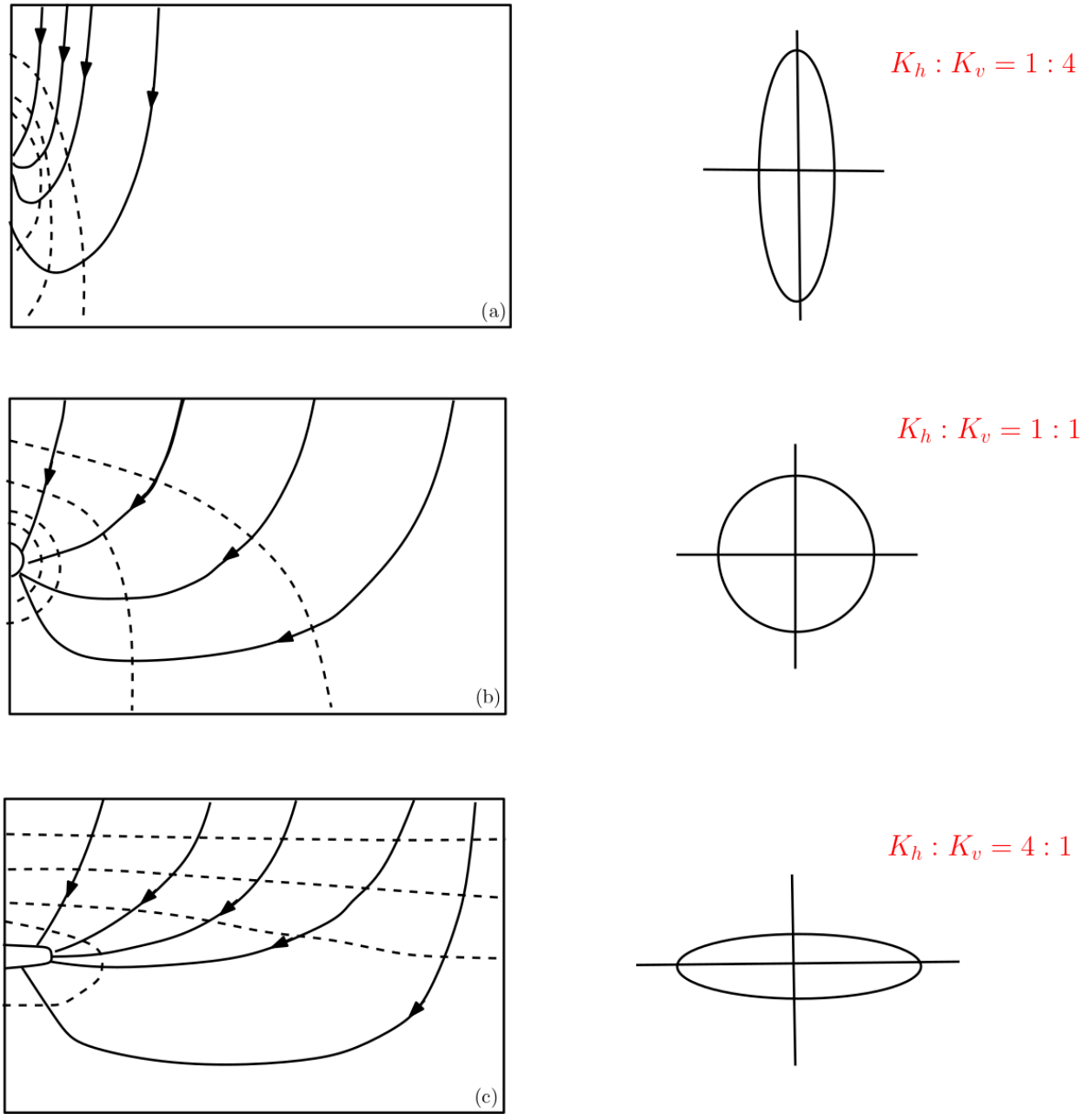


Fig. 2.25: Flow nets in anisotropic aquifer

The interactive simulation on effect of anisotropy on flux and gradient can be found in: *Simulating the Anisotropy and flow direction*

```
# The library used
```

(continues on next page)

(continued from previous page)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from ipywidgets import widgets, interactive



# the main programme
def aniso(a_d, ani_r):

# interim calculation
    a_r = a_d*np.pi/180

    i_xr = np.cos(a_r) # (-), rel. hyd grad. along x
    i_zr = np.sin(a_r) # (-), rel. hyd grad. along z
    K_h = 1 # (-), m/s K_h
    K_v = 1/ani_r # m/s, rel K_v
    f_x = -i_xr*K_h # m/s
    f_z = -i_zr*K_v # m/s
    f_m = np.sqrt(f_x*f_x+f_z*f_z) # m/s

    args = (K_h*i_xr*i_xr + K_v*i_zr*i_zr)/f_m
    an_i_f = ((np.pi-np.arccos(args))*180/np.pi)# deg,

# plots axes

    fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(10,6), gridspec_kw={'width_ratios':␣
↪[3, 1]})

# points for gradient and flux

    grad_px = [0, i_xr]#i_xr
    grad_pz = [0, i_zr]#i_zr

    flux_px = [0, f_x]
    flux_pz = [0, f_z]

# creating points for intersect lines (5 of them)

    p1=-0.5*i_zr; p2 = 0.5*i_zr; p3 =-0.5*i_zr+0.35*i_xr; p4 = 0.5*i_zr+0.35*i_xr; p5␣
↪= -0.5*i_zr+0.7*i_xr
    p6 = 0.5*i_zr+0.7*i_xr;p7 = -0.5*i_zr-0.35*i_xr; p8 = 0.5*i_zr-0.35*i_xr; p9 = -0.
↪5*i_zr-0.7*i_xr;p10 =0.5*i_zr-0.7*i_xr

    q1=0.5*i_xr; q2 =- 0.5*i_xr; q3 =0.5*i_xr+0.35*i_zr; q4 = -0.5*i_xr+0.35*i_zr; q5␣
↪= 0.5*i_xr+0.7*i_zr
    q6 = -0.5*i_xr+0.7*i_zr;q7 = 0.5*i_xr-0.35*i_zr;q8 = -0.5*i_xr-0.35*i_zr; q9 = 0.
↪5*i_xr-0.7*i_zr;q10 = -0.5*i_xr-0.7*i_zr

# plotted points
    l_1x =[p1, p2]; l_1y = [q1, q2]
    l_2x =[p3, p4]; l_2y = [q3, q4]
    l_3x =[p5, p6]; l_3y = [q5, q6]
    l_4x =[p7, p8]; l_4y = [q7, q8]
    l_5x =[p9, p10]; l_5y = [q9, q10]
```

(continues on next page)

(continued from previous page)

```
# creating points for anisotropy
    r1 =1.05 if ani_r >= 1 else 1.5-0.45*ani_r
    r2 = 1.95 if ani_r >= 1 else 1.5+0.45*ani_r
    r3 = 0.5*(r1+r2); r4 = r3

    s1 = -0.5; s2 = s1
    s3 = -0.05 if ani_r<=1 else -0.5+0.45/ani_r
    s4 = -0.95 if ani_r<=1 else -0.5-0.45/ani_r

# plotted points
    Iso_1x = [r1, r2]; Iso_1y = [s1, s2]
    Iso_2x = [r3, r4]; Iso_2y = [s3, s4]
    Iso_x =[r1, r2, r3, r4]; Iso_y = [s1,s2,s3,s4]

# plotting all points

# plotting gradient/flux lines

    ax1.plot(grad_px, grad_pz, "g", label=" gradient") # plotting gradient
    ax1.plot(flux_px, flux_pz, "r", label=" flux") # plotting flux

# plotting intersect lines
    ax1.plot(l_1x, l_1y, "b", label = "head isoline")
    ax1.plot(l_2x, l_2y, "b")
    ax1.plot(l_3x, l_3y, "b")
    ax1.plot(l_4x, l_4y, "b")
    ax1.plot(l_5x, l_5y, "b")
    ax1.legend()


    ax1.spines['left'].set_position('center') # bring the axis lines in center
    ax1.spines['bottom'].set_position('center')
    ax1.spines['right'].set_color('none') # remove the top box
    ax1.spines['top'].set_color('none')
    ax1.set_xticks([]);ax1.set_yticks([]); # remove the ticks
    ax1.set_title("Anisotropy flux and gradient", y=0, pad=-25, verticalalignment="top
↪")


# plotting Anisotropy
    ax2.plot(Iso_1x, Iso_1y, "k", label = r"$K_h: K_v$")
    ax2.plot(Iso_2x, Iso_2y, "k")
    ax2.legend(bbox_to_anchor=(-0.4, -0.05), loc='lower left')
    ax2.set_xlim(1, 2)
    ax2.set_ylim(-1, 1)
    ax2.axis('off')
    ax2.set_title("Anisotropy ratio", y=0, pad=-25, verticalalignment="top");

interactive(aniso,
         a_d=widgets.BoundedFloatText(value=45, min=0, max=360, step=0.5,␣
↪description=r'angle (°)', disabled=False),
         ani_r=widgets.BoundedIntText(value=1, min=1, max=100, step=1, description='K
↪<sub>h</sub>/K<sub>v</sub>', disabled=False),)
```

```
interactive(children=(BoundedFloatText(value=45.0, description='angle (°)', max=360.0,
↪ step=0.5), BoundedIntTe...
```

2.8 Tutorial 1 - Python Programming language

(The contents presented in this section were developed by Dr. P. K. Yadav.)

2.8.1 What is it?

Python is an open-source interpreted, high-level, general-purpose programming language. This means:

- Interpreted/high-level language: This makes we avoid the nuances of fundamental coding as done by computer programmers/engineers.
- General purpose programming language and *Open-source* ecosystem: This means it is extensible. Already over 200,000 Python packages are available (Check Here). Also, it means being **free** of cost.
- For Groundwater: We can use Python packages such as **Numpy** (for numerical computing), **Scipy** (for scientific computing), **Sympy** (for symbolic computing), **Matplotlib** (for plotting) etc. for our computing and modelling in the course.

2.8.2 A bit of history of Python Programming language

Python is now over 15 years old programming language. Its development can be traced from:

- **Guido van Rossum** began developing Python in 1980 at Centrum Wiskunde & Informatica (CWI), the Netherlands. Its implementation (*Python v.1*) was released in 1994.
- *Python 2.0*, released in 2000 became one of the most used general purpose programming language. Python 2.0 is now being replaced by *Python 3.0* (from 2020).
- *Python 3.0* will be used in our class. It is *not* 100% compatible with earlier versions of *Python.*
- *Python* name comes from the British comedy group Monty Python (Van Rossum enjoyed their show). The official *Python* documentation (Check here) also contains various references to Monty Python routines.

2.8.3 Why use Python Programming Language

Many reasons but to put a few points here:

- *Python* is a common tool among engineers, experts and researchers at universities and industry.
- *Python* is system independent, therefore it is highly portable. Beside, it is a versatile (multi-purpose) language.
- *Python* is incredibly flexible and can be adapted to specific local needs using enormous number of **PACKAGES**. Beside, it can easily interface with other languages e.g., C++, Java.
- *Python* is under incredibly active development, improving greatly, and supported wildly by both professional and academic developers.

```
import panel as pn
pn.extension("katex")

p1 = pn.pane.Markdown("""
### Python popularity
""")

p2 = pn.pane.PNG("images/bg1_f1.png", width=500)
```

(continues on next page)

(continued from previous page)

```
p3= pn.pane.Markdown("""</br></br></br>

+ _Python_ has become a mainstream computing language.
+ Details of the plot are [here](shorturl.at/htwQ7).
+ This all means - it is good to learn to code in _Python_
""")

p4 = pn.Column(p1,p2)

pn.Row(p4, p3)
```

```
Row
    [0] Column
        [0] Markdown(str)
        [1] PNG(str, width=500)
    [1] Markdown(str)
```

2.8.4 Very basics of Python Programming

Python is a very extensive language. To get started we learn the very fundamentals of the language.

Fundamentals of Python Programming Language

- ☞ Comments symbol: with #
- ☞ End statement: with semicolon (;) or line-end
- ☞ Multi-line statement: with continuation character (\)
- ☞ Multiple statement: separate each with ;
- ☞ Code Indentation: is required (4 spaces for each)
- ☞ String quotes: use either " " or ' '
- ☞ Naming: is case sensitive b ≠ B
- ☞ Variable Assignment: use =, e.g. `x=5`
- ☞ Multiple Assignment: allowed e.g, `a = b = c = 1` or `a,b,c = 1,7,10`
- ☞ Help: use `help()` and get the cheatsheet from https://www.pythoncheatsheet.org/

Data Types in Python

- Numbers: numbers of any types: `int (e.g., 10)`, `long (e.g., 5192L )`, `float (e.g., 13.4)`, `complex (e.g., 2.13j)`
- String: with quotation mark, e.g., `"john"`
- List: are modifiable placed in [] separated by comma (,). e.g., `[1, 3, "apple", 2, "jo"]`
- Tuple: are non- modifiable placed in ( ) separated by comma (,). e.g., `[11, 3, "ape", 2, "job"]`
- Dictionary: For table-like data. Placed in {} separated by comma (,), e.g.,
  `{'name': 'john','code':6734, 'dept': 'sales'}`

Basic operators in Python

Refer to *Python* documentation for complete description. Python documentation is very extensive and can be obtained from here

Arithmetic Operations

Sym.	Operation	e.g.,
+/-	Add/Subs.	a+b
*	Multiply	a*b
/	Divide	a/b
**	exponent	a**2 = a^2
%	Modulus	Reminder a/b

Bitwise/Comparison

sym.	Operation	e.g.,
&	AND	a & b
|	OR	a|b
==	equal to	a==b
!=	not equal to	a!=b
>/<	greater/less	a>b

A FUNCTION in Python

A *function* in a programming provide an ability to develop a reusable code-block with an option of several operations. This means, a function have input (or a set of input) and provide an (or a set of) output.

Python Function e.g.

```
1 def func1(a,b):
2
3     c = a+b
4     d = c*a
5
6     return c, d
```

What is it?

def in line-1 begins a `Function` block
func1 in line 1 is user-defined function name
a,b in line 1 are function `input`
c,d in lines 3-4 is function `operation`
return line-6 is function `output` block

Semicolon (:) in line 1 and *Indentation* after line 1 are required. **def**, **return** are *Python* keywords. There are quite few of them.

2.8.5 How much Python Programming should we know?

This is probably the most important question. The **clear** answer at least for this course is that **no** programming has to be learned. This course do not expect any pre-coding skills. This course is intended for Basic Groundwater teaching, and that is the focus. But, how about learning groundwater by coding?

Eventually, the depth of programming to learn is an individual choice. This course considers programming as a tool to learn better.

In this course the *codes* can are written in a way so that it can be easily read. In addition, this interactive book will allow quite many of the *code* to be edited and executed in the book itself. For more advanced learning the popular notebook interface **JUPYTER** is to be used.

JUPYTER interface, on which interface this book is developed, is very briefly explained in the next presentation.

2.9 Tutorial 1 - JUPYTER Notebook Interface for Python

(The contents presented in this section were developed by Dr. P. K. Yadav.)

As we mentioned earlier, this interactive book/course can be used without any programming experience. Code based calculations that is part of this book can be mostly executed in the book itself. The codes provided in the book, still with only limited skill programming skill, can be adopted for more illustrative use. This can be done in two ways. First approach, and a quick one, will be to use the web-based tool called Binder Project. The second approach will be to use the codes in the off-line systems. Common to both approach is very useful computing interface called **JUPYTER**.

Here we briefly learn about **JUPYTER** interface.

JUPYTER is a computing interface and has been in development since 2015. *JUPYTER* provide computing interface for several programming language, and thus its name is derived from:

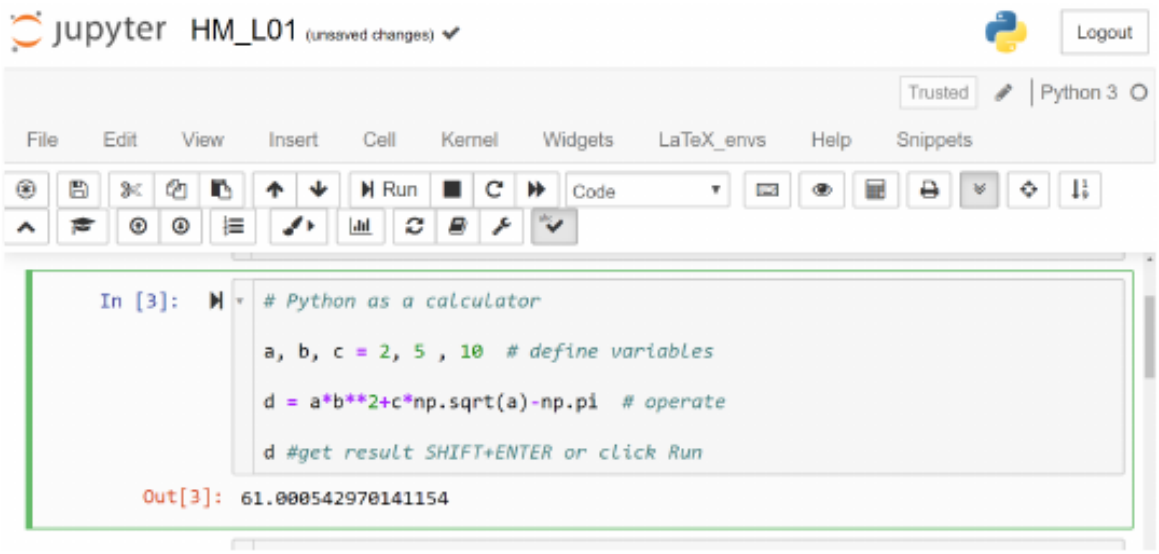
- **JU** : Julia programming language
- **PY** : Python programming language
- **R** : R programming language

More important aspects of **JUPYTER** computing interface:

- Browser-based tool: - i.e., should also run in smartphone/tabs in a
- OPen-source: i.e., community based development and personalization
- Active development: Very actively under development, especially from academic/research sector.

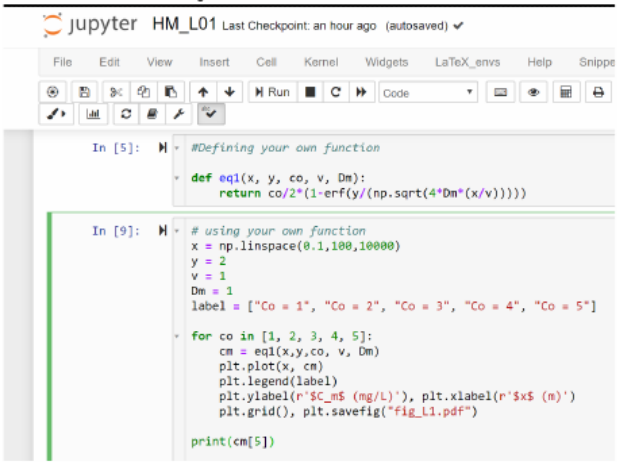
2.9.1 How to use JUPYTER

JUPYTER has a block-based interface (called **CELL**). Each *Cell* is either an **input (In[1])** cell or corresponding an **output (Out[1])** cell. The *input* and *output* cells can be interactively operated.

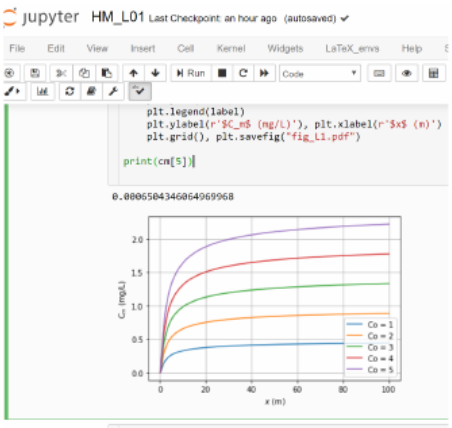


The *cell-based* interface is very intuitive specially for learning as it can show combine for example the code or mathematical concept with a visualization, i.e., both the cause and effect can be observed immediately and dynamically.

Definition parts



Execution and Result parts



For more advanced use, JUPYTER interface can be used for developing codes in different programming languages other than *Python.*

The JUPYTER cheat-sheet (from here] can be helpful to get quickly started.

Also, for better computing and learning, installing **JUPYTER** locally in personal system is encouraged. This can be done using the instructions provided here.

2.10 Installing Python and JUPYTER notebook in your system

The document from here provide details of installing Python and JUPYTER notebook in your system.

```
import numpy as np
import pandas as pd
import panel as pn
pn.extension("katex", "mathjax")
```

2.11 Tutorial 2 - Aquifer and Storage Properties

(The contents presented in this section were re-developed principally by Dr. P. K. Yadav. The original contents are from Prof. Rudolf Liedl)

2.11.1 Tutorial Problem 1

```
#
r1_1 = pn.pane.Markdown("""

The park called ,,Grosser Garten'' in Dresden is underlain by an unconfined aquifer␣
↪consisting of
alluvial deposits. How much additional water is stored under the park if groundwater␣
↪levels
rise by 3 m during a wet period?
""", style={'font-size': '12pt'})

spacer = pn.Spacer(width=50)

r1_2 = pn.pane.PNG("images/T02_TP1.png", width=500)

pn.Row(r1_1,spacer, r1_2)
```

```
Row
    [0] Markdown(str, style={'font-size': '12pt'})
    [1] Spacer(width=50)
    [2] PNG(str, width=500)
```

```
#
r1_3 = pn.pane.Markdown("""

### Tutorial Problem 1 - Solution ###


**For details check lecture slide: L02- 8**
<br><br>

The basic configuration of an unconfined aquifer:

""", style={'font-size': '12pt'})

spacer = pn.Spacer(width=50)
```

(continues on next page)

(continued from previous page)

```
r1_4 = pn.pane.PNG("images/T02_TP1_2.png", width=600)

pn.Row(r1_3,spacer, r1_4)


#<img src="images/T01_TP1_2.PNG" alt="Grosser Garten Map"  width="100%" height="100%"
↪>
```

```
Row
    [0] Markdown(str, style={'font-size': '12pt'})
    [1] Spacer(width=50)
    [2] PNG(str, width=600)
```

```
# The given information of the problem are

L = 2 # km as length of garten
W = 1 # km as Width of garten
DP = 3 # m change in pressure head
n = 0.3 # porosity that we assume to be 30%
A = L * W *1e6 # m^2, Area unit convered.
DV = A * DP # m^3, increase in total volume due to change in pressure head
AW = n * DV # m^3, additional Water volumne

print("Park Area is {0:1.1f}".format(A),"m\u00b2")
print("Increase in total volume: {0:1.1E}".format(DV),"m\u00b3")
print("Additional water volume: {0:1.1E}".format(AW),"m\u00b3")
```

```
Park Area is 2000000.0 m²
Increase in total volume: 6.0E+06 m³
Additional water volume: 1.8E+06 m³
```

2.11.2 Tutorial Problem 2

```
#
r2_1 = pn.pane.Markdown("""
The lithological information derived from three boreholes is given in the figure
↪below.
Try to correlate the layers to obtain a 2D cross section of the subsurface structure.
""", style={'font-size': '12pt'})

r2_2 = pn.pane.PNG("images/T02_TP2_1.png", width=400)

r2_3 = pn.pane.Markdown("""
### Tutorial Problem 2 - Solution ###

**For details check lecture slide: L03-15**.<br>

Few correlations between the layers are presented in the figure below:
""", style={'font-size': '12pt'})

R1 = pn.Row(r2_1, r2_2)
pn.Column(R1, r2_3)
```

```
Column
    [0] Row
        [0] Markdown(str, style={'font-size': '12pt'})
        [1] PNG(str, width=400)
    [1] Markdown(str, style={'font-size': '12pt'})
```

```
# SOlution 2

img_1 = pn.pane.PNG("images/T02_TP2_1.png", width=500)
img_2 = pn.pane.PNG("images/T02_TP2_2.png", width=300)
img_3 = pn.pane.PNG("images/T02_TP2_3.png", width=300)
img_4 = pn.pane.PNG("images/T02_TP2_4.png", width=300)

tabs = pn.Tabs(('Sample', img_1), ("Solution 1", img_2), ("Solution 2", img_3), (
↪"Solution 3", img_4))
tabs
```

```
Tabs
    [0] PNG(str, width=500)
    [1] PNG(str, width=300)
    [2] PNG(str, width=300)
    [3] PNG(str, width=300)
```

2.11.3 Tutotorial Problem 3

```
r3_1 = pn.pane.Markdown("""

This problem addresses a confined aquifer with a thickness of 61 m and a specific␣
↪storage of
1.2 X 10<sup>-5</sup> 1/m. Due to water injection the pressure head rises by 0.75 m␣
↪on average over an area 230 m in diameter.
How much water is injected?

""", width=950, style={'font-size': '12pt'})

r3_2 = pn.pane.Markdown("""
### Tutorial Problem 3 - Solution ###

**For details check lecture slide: L03-28** <br><br>

Basic configuration of a confined aquifer:
<br><br>
""", style={'font-size': '12pt'})

r3_3 = pn.pane.PNG("images/T02_TP3.png", width=600)

r3_4= pn.pane.LaTeX(r"""
specific storage: <br> $S_s = \frac{\Delta V_w}{V_T \cdot \Delta \psi } $ <br>
with <br>
$\Delta V_w$ = change in water volume [L$^3$] <br>
$V_T$ = total volume [L$^3$]<br>
$\Delta \psi$ = change in pressure head [L]
""", style={'font-size': '12pt'})
```

(continues on next page)

(continued from previous page)

```
r3_5 = pn.Row(r3_3, r3_4)
pn.Column(r3_1, r3_2, r3_5)
```

```
Column
    [0] Markdown(str, style={'font-size': '12pt'}, width=950)
    [1] Markdown(str, style={'font-size': '12pt'})
    [2] Row
        [0] PNG(str, width=600)
        [1] LaTeX(str, style={'font-size': '12pt'})
```

```
# Given data
d = 230 # m,  diameter of the aquifer
m = 61 # m, thickness of the aquifer
Ss = 1.2 * 10e-5 # 1/m, specific storage
DP_h = 0.75 # m, pressure head difference

# Calculations

A = np.pi *(d/2)**2 # m^2, area of the aquifer
Vt = A*m # m^3 Total volume of the aquifer
DV_w = Ss*Vt*DP_h # m^3, additional water volume

print("The Aquifer Area is {0:1.2E}".format(A),"m\u00b2")
print("The Total Volume is {0:1.2E}".format(Vt),"m\u00b3")
print("The Additional Water is {0:1.2f}".format(DV_w),"m\u00b3")
```

```
The Aquifer Area is 4.15E+04 m²
The Total Volume is 2.53E+06 m³
The Additional Water is 228.10 m³
```

2.11.4 Tutorial Problem 4:

```
# Tut Problem 4
rx_1 = pn.pane.Markdown("""
We consider an unconfined aquifer with a storage coefficient of 0.19. What will be␣
↪the change
in water volume if the following drawdowns are observed in four sub-areas in a dry␣
↪period:
""", width = 700, style={'font-size': '12pt'})
rx_1
```

```
Markdown(str, style={'font-size': '12pt'}, width=700)
```

```
#
Head = ["Sub-Area", "Size, (Km2)", "drawdown (m)", "Change in water volume (m3)"]
Sub_Area = ["A", "B", "C", "D"] #name
Size = [36, 18, 72, 85] # km^2, area
Drawdown = [0.85, 1.09, 1.65, 2.37] # m, equivalent to change in pressure head
Q4t = np.transpose([Sub_Area, Size, Drawdown])

data = {"Sub-Area": Sub_Area, "Size, (Km2)": Size, "drawdown (m)": Drawdown, }
df1= pd.DataFrame(data)
df1.set_index('Sub-Area')
```

```
          Size, (Km2)  drawdown (m)
Sub-Area
A                  36          0.85
B                  18          1.09
C                  72          1.65
D                  85          2.37
```

2.11.5 Tutorial Problem 4 – Solution

For details check lecture slides L03 - 28, 29 and 31

In unconfined aquifer Stortavity (S) is used instead of storage coefficient S_s. They both are related with thickness (m) as:

$$S = S_s \times m$$

and so in unconfined aquifer, we get:

$$S = \frac{\Delta V_w}{A \cdot \Delta \psi}$$

with ΔV_w = change in water volume [L^{33}] A = Domain area [L^2] $\Delta \psi$ = change in pressure head [L]

```
# Given information
s = 0.19
Size = [50, 18, 72, 85] # km^2, area
Drawdown = [0.85, 1.09, 1.65, 2.37] # m, equivalent to change in pressure head

# Solution part
Vol_cha = s*np.multiply(Size, Drawdown)*10**6 # m^3, vol change = s*area size *␣
↪drawdawn

# output printing
data2 = {"Sub-Area": Sub_Area, "Size, (Km2)": Size, "drawdown (m)": Drawdown, "Change␣
↪in volume (Km3)":Vol_cha/1e9  }
df2= pd.DataFrame(data2)
df3= df2.set_index('Sub-Area')

print(df3, "\n")
print("The total change in the volume of water is: {0:0.3f} Km\u00b3".format(sum(Vol_
↪cha/1e9)))
```

```
          Size, (Km2)  drawdown (m)  Change in volume (Km3)
Sub-Area
A                  50          0.85                0.008075
B                  18          1.09                0.003728
C                  72          1.65                0.022572
D                  85          2.37                0.038275

The total change in the volume of water is: 0.073 Km³
```

2.11.6 Tutorial Problem 5:

A confined aquifer is considered in this problem. Specific storage and total porosity equal 7.5×10^{-6} 1/m and 30%, respectively. What is the compressibility of the porous medium? (compressibility of water: 4.6×10^{-10} m^2/N, density of water: 998 kg/m^3)

2.11.7 Tutorial Problem 5 – Solution

For details check slide nr. L03-28

Specific Storage,

$$S_s = (n\alpha_w + \alpha_{pm})\rho_w g$$

with: n = Total porosity [-] g = acceleration due to gravity [L/T^2] α_w = compressibility of water [LT^2/M] α_{pm} = compressibility of porous medium [LT^2/M] ρ_w = density of water [M/L^3]

Solve for α_{pm}: $\frac{S_s}{\rho_w g} - n\alpha_w = \alpha_{pm}$

```
# Given data

n = 0.3 # unitless, total porosity
rho_w = 998 # kg/m3, density of water
g = 9.81 # m/s2, accl. due to gravity
alpha_w = 4.6*1e-10 # m2/N, compressibility of water
S_s = 7.5*1e-6 # 1/m, specific storage

# calculated land subsidence (LS)
alpha_pm5 = S_s/(rho_w*g) - n*alpha_w

print("The Compressibility of Porous mdeid is {0:0.2E}".format(alpha_pm5), "m\u00b2/N
↪")
```

```
The Compressibility of Porous mdeid is 6.28E-10 m²/N
```

2.11.8 Tutorial Problem 6:

Due to water extraction from a confined aquifer the pressure head is lowered by 183 m. The following aquifer parameters are available: storage coefficient = 5×10^{-4}, total porosity = 0.33, thickness (before water extraction) = 80 m, compressibility of the porous medium = 6.9×10^{-8} m^2/N and the density of water is 998 kg/m^3

What is the amount of land subsidence resulting from the water extraction?

2.11.9 Tutorial Problem 6 – Solution

For details see slide nr. L03-25 and 26

Change in total volume due to Δp_{pm} :

$$\Delta V_T = \alpha_{pm} V_T \rho_w g \Delta\psi$$

α_{pm} = compressibility of porous medium [LT^2/M] V_T = total volume [L^3] ρ_w = density of water [M/L^3] g = acceleration of gravity [L/T^2] $\Delta\psi$ = change in pressure head [L]

$\Delta V_T = A \times \Delta m$ and $V_T = A \times m$ and with A = area of the aquifer [L/T^2] and m = Thickness of the aquifer [L]. Substituting these relation in the above equation we get:

$$\Delta m = \alpha_{pm} m \rho_w g \Delta \psi$$

```
# Given data
alpha_pm = 6.9 * 1e-8 # m2/N, compressibility of porous medium
m = 80 # m, thickness
rho_w = 998 # kg/m3, density of water
g = 9.81 # m/s2, accl. due to gravity
DP_h = 30 # m, change in pressure head

# calculated land subsidence (LS)
LS = alpha_pm*m*rho_w*g*DP_h

print("The land subsidence is {0:0.2f}".format(LS), "m")
```

```
The land subsidence is 1.62 m
```

2.11.10 HOME WORK PROBLEMS

Homework Problem 1

The pressure head in an aquifer extending over 200 km^2 is decreased by 1.60 m. Determine the loss of groundwater in the aquifer for two scenarios: The aquifer is unconfined (storage coefficient 0.13). The aquifer is confined (storage coefficient 0.0005).

Homework Problem 2

Conduct a sieve analysis for a dried soil sample (see data in the table below)

1. Draw the granulometric curve (cumulative mass distribution) and briefly characterise the sediment with regard to its major constituent(s).
2. What is the coefficient of uniformity?

```
#
Head = ["mesh size [mm]", "residue in the sieve [g] ", " total", " / total"]
Size = [6.3, 2, 0.63, 0.2, 0.063, "< 0.063 /cup"]
residue = [11, 62, 288, 189, 42, 8]


data3 = {"mesh size [mm]": Size, "residue in the sieve [g] ": residue}
df4= pd.DataFrame(data3)
df4.set_index("mesh size [mm]")
```

mesh size [mm]	residue in the sieve [g]
6.3	11
2	62
0.63	288
0.2	189
0.063	42
< 0.063 /cup	8

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import panel as pn
pn.extension("katex")
```

2.12 Tutorial 3 - Darcy Law and Conductivity

(The contents presented in this section were re-developed principally by Dr. P. K. Yadav. The original contents are from Prof. Rudolf Liedl)

- **tutorial problems on Darcy's law and intrinsic permeability**
- **homework problems on Darcy's law and intrinsic permeability**

2.12.1 Tutorial Problems on

- Darcy's Law and
- Intrinsic Permeability

2.12.2 Tutorial Problem 7: Flow Direction and Hydraulic Gradient

Indicate the direction of flow shown in the figure in next slides, and determine the hydraulic gradient for a Darcy column with length L = 50 cm! (Figures not to scale.)

Tutorial Problem 7 – Solution

The relevant topic is covered in Lecture 04, slide 8

```
png_pane = pn.pane.PNG("images/T03_TP7_a1.png", width=600)
png_pane
```

```
PNG(str, width=600)
```

```
# Image (a)
L = 0.5 # m, length of the column
Ea_hl = 0.2 #, m, elevation head, left
Pa_hl = 0.1 #, m  pressure head, left
Ea_hr = 0.2 #, m, elevation head, right
Pa_hr = 0.3 #, m  pressure head, right
Ha_hl = Ea_hl + Pa_hl # m, hydraulic head, left
Ha_hr = Ea_hr + Pa_hr # m, hydraulic head, right
DH_a = Ha_hr - Ha_hl
H_ga = DH_a/L#, no unit, hydraulic gradient

print("Hydraulic head LEFT: {0:1.1f}".format(Ha_hl),"m"); print("Hydraulic head␣
↪RIGHT:: {0:1.1f}".format(Ha_hr),"m")
print("Hydraulic Head Difference: {0:1.1f}".format(DH_a),"m");print("Hydraulic␣
↪gradient: {0:1.1f}".format(H_ga))
png_pane.object = "images/T03_TP7_a2.png"
```

```
Hydraulic head LEFT: 0.3 m
Hydraulic head RIGHT:: 0.5 m
Hydraulic Head Difference: 0.2 m
Hydraulic gradient: 0.4
```

```
png_pane2 = pn.pane.PNG("images/T03_TP7_b1.png", width=500)
png_pane2
```

```
PNG(str, width=500)
```

```
# Image (b)
L = 0.5 # m, length of the column
Eb_hl = 0.2 #, m, elevation head, left
Pb_hl = 0.1 #, m  pressure head, left
Eb_hr = 0.05 #, m, elevation head, right
Pb_hr = 1.3 #, m  pressure head, right
Hb_hl = Eb_hl + Pb_hl # m, hydraulic head, left
Hb_hr = Eb_hr + Pb_hr # m, hydraulic head, right
DH_b = Hb_hr - Hb_hl
H_gb = DH_b/L#, no unit, hydraulic gradient
print("Hydraulic head LEFT: {0:1.1f}".format(Hb_hl),"m");print("Hydraulic head␣
↪RIGHT:: {0:1.1f}".format(Hb_hr),"m")
print("Hydraulic Head Difference: {0:1.1f}".format(DH_b),"m");print("Hydraulic␣
↪gradient: {0:1.1f}".format(H_gb))
png_pane2.object = "images/T03_TP7_b2.png"
```

```
Hydraulic head LEFT: 0.3 m
Hydraulic head RIGHT:: 1.4 m
Hydraulic Head Difference: 1.1 m
Hydraulic gradient: 2.1
```

```
png_pane3 = pn.pane.PNG("images/T03_TP7_c1.png", width=400)
png_pane3
```

```
PNG(str, width=400)
```

```
# Image (c)
L = 0.5 # m, length of the column
Ec_hl = 0.3 #, m, elevation head, left
Pc_hl = 0.1 #, m  pressure head, left
Ec_hr = 0.2 #, m, elevation head, right
Pc_hr = 0.2 #, m  pressure head, right
Hc_hl = Ec_hl + Pc_hl # m, hydraulic head, left
Hc_hr = Ec_hr + Pc_hr # m, hydraulic head, right
DH_c = Hc_hr - Hc_hl
H_gc = DH_c/L#, no unit, hydraulic gradient
print("Hydraulic head LEFT: {0:1.1f}".format(Hc_hl),"m");print("Hydraulic head␣
↪RIGHT:: {0:1.1f}".format(Hc_hr),"m")
print("Hydraulic Head Difference: {0:1.1f}".format(DH_c),"m");print("Hydraulic␣
↪gradient: {0:1.1f}".format(H_gc))
png_pane3.object = "images/T03_TP7_c2.png"
```

```
Hydraulic head LEFT: 0.4 m
Hydraulic head RIGHT:: 0.4 m
```

(continues on next page)

(continued from previous page)

```
Hydraulic Head Difference: 0.0 m
Hydraulic gradient: 0.0
```

```
png_pane4 = pn.pane.PNG("images/T03_TP7_d1.png", width=400)
png_pane4
```

```
PNG(str, width=400)
```

```
# Image (d)
L = 0.5 # m, length of the column
Ed_hl = 0.3 #, m, elevation head, left
Pd_hl = 0.1 #, m  pressure head, left
Ed_hr = 0.2 #, m, elevation head, right
Pd_hr = 0.1 #, m  pressure head, right
Hd_hl = Ed_hl + Pd_hl # m, hydraulic head, left
Hd_hr = Ed_hr + Pd_hr # m, hydraulic head, right
DH_d = Hd_hr - Hd_hl
H_gd = DH_d/L#, no unit, hydraulic gradient
#output
print("Hydraulic head LEFT: {0:1.1f}".format(Hd_hl),"m");print("Hydraulic head␣
↪Right:: {0:1.1f}".format(Hd_hr),"m")
print("Hydraulic Head Difference: {0:1.1f}".format(DH_d),"m");print("Hydraulic␣
↪gradient: {0:1.1f}".format(H_gd))
png_pane4.object = "images/T03_TP7_d2.png"
```

```
Hydraulic head LEFT: 0.4 m
Hydraulic head Right:: 0.3 m
Hydraulic Head Difference: -0.1 m
Hydraulic gradient: -0.2
```

2.12.3 Tutorial Problem 8

The hydraulic conductivity of a fine sand sample was found to be equal to 1.36×10^{-5} m/s in a Darcy experiment using water at a temperature of 20°C. What is the intrinsic permeability of this sample? Give results in cm^2 and D. (density of water at 20°C: 998.2 kg/m^3; dynamic viscosity of water at 20°C: 1.0087×10^{-3} Pa·s; 1 D = 0.987×10^{-12} m^2)

Tutorial Problem 8 - Solution

Relevant topics are covered in Lecture 04 slides 18-20.

Relationship between hydraulic conductivity K and intrinsic permeability k from lecture notes: $$K_{water} = k \cdot \frac{\rho_{water} \cdot g}{\eta_{water}}$$

Solve for , k

$$k = \frac{\eta_{water} \cdot K_{water}}{\rho_{water} \cdot g}$$

```
#Given
Darcy = 0.987 * 10**-12 # m^2, 1D = 0.987*10^-12 m^2
nu_w = 1.00087*10**-3 # Pa-S dynamic viscosity of water
```

(continues on next page)

(continued from previous page)

```
K_w = 1.36*10**-5 # m/s Conductivity of water
g = 9.81 # m/s^2 accln due to gravity
rho_w = 998.2 # kg/m^3, density of water

# Solution
k = (nu_w*K_w)/(rho_w*g)#, m^2, permeability of water
k_D = k/Darcy


print("The permeability is {0:1.1E}".format(k),"m\u00b2")
print("The permeability in Darcy unite is: {0:1.2f}".format(k_D),"D")
```

```
The permeability is 1.4E-12 m²
The permeability in Darcy unite is: 1.41 D
```

2.12.4 Tutorial Problem 9: Constant-Head Permeameter

```
## Tutorial Problem 9: Constant-Head Permeameter

r1 = pn.pane.LaTeX(r"""
(a). Derive the expression for $K$ given below.
$$
K = \frac{QL}{A(h_{in}-h_{out})}
$$

(b). The hydraulic conductivity of a medium sand sample (length 15 cm,
cross-sectional area 25 cm$^2$) is to be determined. The hydraulic head
difference is 5 cm and a water volume of 100 cm$^3$ pas-sed the sample during
an experimental period of 12 min.
<br><br>

(c). How long would 100 cm$^3$ diesel (density: 0.85 g/cm$^3$, dynamic viscosity: 3.5
↪$\cdot 10^{-3}$
Pa$\cdot$s at 20$^\circ$C) need to pass the sample under a head difference of 5 cm␣
↪(density and
dynamic viscosity of water at 20$^\circ$C: 998.2 kg/m$^3$ and 1.0087$\cdot 10^{-3}$ Pa
↪$\cdot$s, resp.)?
""", width=450, style={'font-size': '12pt'})
spacer = pn.Spacer(width=100)

r2 =pn.pane.PNG("images/T03_TP9.png", width=400)

pn.Row(r1,spacer, r2)
```

```
Row
    [0] LaTeX(str, style={'font-size': '12pt'}, width=450)
    [1] Spacer(width=100)
    [2] PNG(str, width=400)
```

```
### Tutorial Problem 9 - Solution ###

r1 = pn.pane.LaTeX(r""" Solution Problem 9: <br>
```

(continues on next page)

(continued from previous page)

```
The relevant topic can be found in lecture 04, slides 15, 18-20
<br><br>

Let the reference datum be at the bottom. Then from Darcy's Law:
$$Q= -A\cdot K\frac{h_{out}-h_{in}}{L}$$
With,


Q = discharge [L$^3$/T]<br>
L =column length [L]<br>
A = cross-sectional area of column [L$^2$]<br>
K = hydraulic conductivity [L/T]<br>
h$_{in}$ =hydraulic head at column inlet [L]<br>
h$_{out}$ = hydraulic head at column outlet [L]<br>
<br>
Solve for $K$:
$$
K= \frac{Q\cdot L}{A\cdot(h_{in}-h_{out})}
$$

If the reference level $(z = 0)$ is located at the downgradient overflow, then set $h_
↪{out} = 0$.
""", width= 500, style={'font-size': '13pt'} )
spacer = pn.Spacer(width=100)

r2 =pn.pane.PNG("images/T03_TP9a.png", width=300)

pn.Row(r1,spacer, r2, width=1000)
```

```
Row(width=1000)
    [0] LaTeX(str, style={'font-size': '13pt'}, width=500)
    [1] Spacer(width=100)
    [2] PNG(str, width=300)
```

```
#Given (solution on 9b)
L = 15 #cm, length of column
A = 25 # cm^2, surface area of column
h_diff = 5 # cm, h_in-h_out
Q = 100/12 # cm^3/min discharge per min

# Solution using derived equation in first part of the problem
# K = QL/A(h_in- h_out)

K = (Q*L)/(A*h_diff)# cm/min, required conductivity
K_1 = K*10**-2/60 #, m/s, conductivity in m/s

#output
print("The conductivity in column is {0:1.2E}".format(K),"cm/min")
print("The conductivity in column is {0:1.2E}".format(K_1),"m/s \n")

if K_1 <= 1.67*10**-4:
    print("Fine to medium sand")
else:
    print("to check further") # to be completed later.
```

```
The conductivity in column is 1.00E+00 cm/min
The conductivity in column is 1.67E-04 m/s

Fine to medium sand
```

Continue solution on 9c

Discharge and Darcy's law: $Q_{water} = \frac{V}{t_{water}} = -A \cdot K_{water} \cdot \frac{\Delta h}{L}$

Solve for t_{water}: $t_{water} = \frac{V}{Q_{water}} = -\frac{V}{A \cdot K_{water} \cdot \Delta h / L} = -\frac{V \cdot L}{A \cdot K_{water} \cdot \Delta h}$

Same step for t_{diesel}: $t_{diesel} = -\frac{V \cdot L}{A \cdot K_{diesel} \cdot \Delta h}$

time ratio: $\frac{t_{diesel}}{t_{water}} = \frac{-\frac{V \cdot L}{A \cdot K_{diesel} \cdot \Delta h}}{-\frac{V \cdot L}{A \cdot K_{water} \cdot \Delta h}} = \frac{K_{water}}{K_{diesel}}$

Use relationship between conductivity K and permeability k from lecture notes (slides 18) $\$\frac{K_{water}}{K_{diesel}} = \frac{k \cdot \frac{\rho_{water} \cdot g}{\eta_{water}}}{k \cdot \frac{\rho_{diesel} \cdot g}{\eta_{diesel}}} = \frac{\rho_{water} \cdot \eta_{diesel}}{\rho_{diesel} \cdot \eta_{water}}\$$

Solve for t_{diesel}

```
# Given data

rho_w = 920.2 # kg/m^3, density of water at 20°C
eta_w = 1.0087*10**-3#, Pa-S dynamic viscosity of water
rho_d = 0.85 # g/cm^3, density of diesel at 20°C
eta_d = 3.5*10**-3#, Pa-S dynamic viscosity of diesel
V_d = 100 # cm^3 volume of diesel
t_w = 12 # min, time taken by water

# Calculations

t_d = (rho_w*eta_d)/(rho_d*1000*eta_w)*t_w

# multiplied by 1000 to convert unit g/cm^3 to kg/m^3

print("The time required for diesel will be: {0:0.2f}".format(t_d), "min")
```

```
The time required for diesel will be: 45.08 min
```

2.12.5 Tutorial Problem 10: Falling-Head Permeameter

```
# Tutorial Problem 10

r10_1 = pn.pane.LaTeX(r"""
$$
K = \frac{d_t^2 L}{d_c^2 L}\cdot \ln\frac{h_{in}(0)-h_{out}}{h_{in}(t)-h_{out}}
$$
""", width=400, style={'font-size': '12pt'})

r10_2 = pn.pane.Markdown("""
1. Derive the expression for K given above.
<br><br>
2. The hydraulic conductivity of a fine sand sample (length 15 cm, diameter 10 cm) is to be determined.
The hydraulic head difference at the beginning and at the end of the experiment after 528 min is 5 cm and 0.5 cm, resp.
```

(continues on next page)

(continued from previous page)

```
The inner tube diameter is 2 cm.

""", width= 500, style={'font-size': '12pt'})

r10_2a = pn.pane.Markdown("""
### Tutorial Problem 10: Solution ###
<br>
**Relevant information can be found in Lecture 04, Slides 14 and 16**
""", width= 500, style={'font-size': '12pt'})

col1 = pn.Column(r10_1, r10_2, r10_2a)

r10_3 =pn.pane.PNG("images/T03_TP10.png", width=350)
spacer3 = pn.Spacer(width=50)
pn.Row(col1, spacer3, r10_3)
```

```
Row
    [0] Column
        [0] LaTeX(str, style={'font-size': '12pt'}, width=400)
        [1] Markdown(str, style={'font-size': '12pt'}, width=500)
        [2] Markdown(str, style={'font-size': '12pt'}, width=500)
    [1] Spacer(width=50)
    [2] PNG(str, width=350)
```

```
# Tutorial Problem 10: Solution

r10_a1 = pn.pane.LaTeX(r"""
Darcy's Law:
$$
Q(t) = -A_c \cdot K\cdot\frac{h_{out} - h_{in}(t)}{L}
$$

Volumetric budget for standpipe:
$$
Q(t) = -\frac{dV_t}{dt}(t) = -A_t \cdot \frac{dh_{in}}{dt}(t)
$$
with <br>
$A_t$ = cross-sectional area of standpipe [L$^2$]<br>
$V_t$ = water volume in standpipe [L$^3$]<br>
<br>
combine Darcy's law and the volumetric budget:
$$
-A_t \cdot \frac{dh_{in}}{dt}(t) =  -A_c \cdot K\cdot\frac{h_{out} - h_{in}(t)}{L}
$$

solve for $dh_{in}/dt$:
$$
\frac{dh_{in}}{dt} = \frac{K}{L}\frac{A_c}{A_t}(h_{out}-h_{in}) = \frac{K}{L}\bigg(\
↪frac{d_c}{d_t}\bigg)^2(h_{out}-h_{in})
$$

""", style={'font-size': '13pt'})
r10_a2 =pn.pane.PNG("images/T03_TP10.png", width=300)


pn.Row(r10_a1, r10_a2)
```

```
Row
    [0] LaTeX(str, style={'font-size': '13pt'})
    [1] PNG(str, width=300)
```

```
#
r10_a3 = pn.pane.LaTeX(r"""
Equation for falling head: $ \frac{dh_{in}}{dt}  = \frac{K}{L}\big(\frac{d_c}{d_t}\
↪big)^2(h_{out}-h_{in}) $
<br> <br>
This equation is an ordinary differential equation of first order.
Providing hydraulic head $h_{in}(0)$ at the beginning of the experiment $(t = 0)$, it
may be solved by separation of variables:
$$
\frac{dh_{in}}{h_{out}-h_{in}} = \frac{K}{L}\bigg(\frac{d_c}{d_t}\bigg)^2 dt
$$
integrations on both sides by considering the initial condition:
$$
\int_{h_{in}(0)}^{h_{in}(t)}\frac{dh_{in}}{h_{out}-h_{in}} = \int_0^t \frac{K}{L}\
↪bigg(\frac{d_c}{d_t}\bigg)^2 dt = \frac{K}{L}\bigg(\frac{d_c}{d_t}\bigg)^2 \int_0^t␣
↪dt
$$
determine integral functions:
$$
\Bigg[-\ln(h_{out} - h_{in}) \Bigg]_{h_{in}(0)}^{h_{in}(t)} = \frac{K}{L}\bigg(\frac
↪{d_c}{d_t}\bigg)^2 [t]_0^t
$$
insert limits of integration:
$$
-\ln\frac{h_{out}-h_{in}(t)}{h_{out}-h_{in}(0)} = \frac{K}{L}\bigg(\frac{d_c}{d_t}\
↪bigg)^2 t
$$
solve for K:
$$
K = \bigg(\frac{d_c}{d_t}\bigg)^2 \frac{L}{t}\ln\frac{h_{in}(0)-h_{out}}{h_{in}(t)-h_
↪{out}}
$$
""", width=600, style={'font-size': '12pt'})
r10_a3
```

```
LaTeX(str, style={'font-size': '12pt'}, width=600)
```

```
# Solution 9(2)

L = 15 # cm, length
L_m = L/100 # m, length
d_c = 10 # cm, diameter column
d_t = 2 # cm, diameter tube
h_d0 = 5 # cm, head difference at start
h_dt = 0.5 # cm, head difference at time t
t = 528 # min, total time
t_s = 528*60 # sec, total time in seconds

#solution using the developed equation

K = (d_t/d_c)**2 * ((L_m)/t_s)*np.log(h_d0/h_dt)
```

(continues on next page)

(continued from previous page)

```
#Output
print("The conductivity in column is {0:1.2E}".format(K),"m/s \n")

if K < 1.67*10**-5:
    print("Silt or silty sand")
else:
    print("to check further") # to be completed later.
```

```
The conductivity in column is 4.36E-07 m/s

Silt or silty sand
```

2.12.6 HOME WORK PROBLEMS

Darcy's Law and Intrinsic Permeability

There is no obligation to solve homework problems!

Try to submit within next 2 weeks.

2.12.7 Homework Problem 3

```
#
r_h3a = pn.pane.Markdown("""

**A**. Derive an expression for hydraulic conductivity *_K_* for the constant-head␣
↪permeameter shown in the figure.<br><br>
**B**. The hydraulic conductivity of a sample (length 10 cm, diameter 4 cm) is to be␣
↪determined.
The water depths a<sub>1</sub> and a<sub>2</sub> equal 6 cm and 3 cm, resp. A water␣
↪volume of 250 ml passed the sample during an experimental period of 36 s. <br> <br>
**C**. Which material could be contained in the sample?
""", width = 400, style={'font-size': '12pt'})
spacer2=pn.Spacer(width=50)

r_h3b = pn.pane.PNG("images/T03_TH3.png", width=500)

pn.Row(r_h3a,spacer2, r_h3b)
```

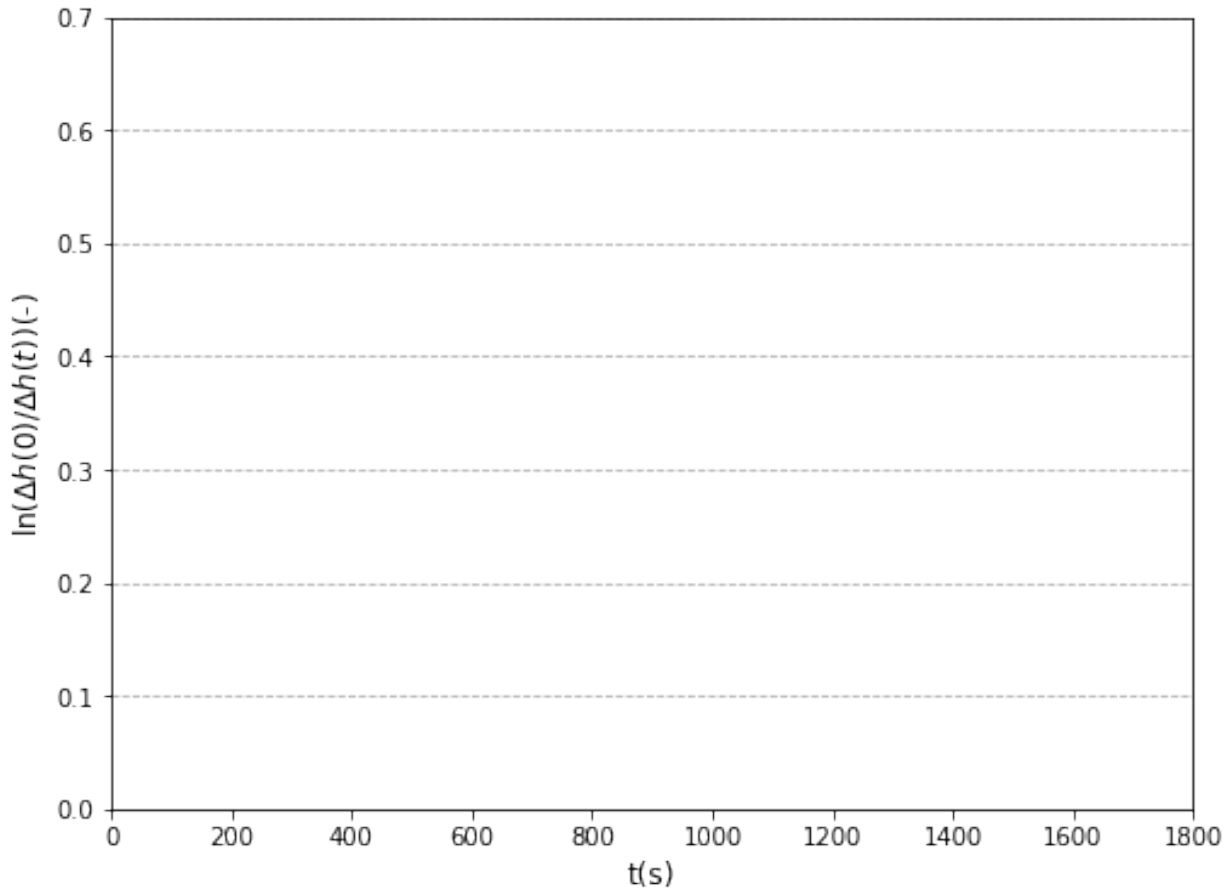
```
Row
    [0] Markdown(str, style={'font-size': '12pt'}, width=400)
    [1] Spacer(width=50)
    [2] PNG(str, width=500)
```

2.12.8 Homework Problem 4

```
#
r_h4 = pn.pane.Markdown("""
A Darcy experiment is performed by a falling-head permeameter using water at 20°C.
Length and diameter of the sample are 20 cm and 6 cm, resp. The inner tube diameter␣
↪is 4 cm.
The following data are available for the time-dependent hydraulic head difference :
""", style={'font-size': '12pt'})

r_h4b = pn.pane.PNG("images/T03_TH4a.png", width=400)

r_h4c = pn.pane.Markdown("""
**A.** Convert times to seconds and plot the logarithm of the ratios of head␣
↪differences ln(h(0)/h(t)) vs. time t.
(Use the coordinate system on next page). <br><br>
**B.** Determine the slope of the corresponding regression line.<br><br>
**C.** Determine hydraulic conductivity K.<br><br>
**D.** Determine intrinsic permeability k.<br>

""", style={'font-size': '12pt'})
r_h4d = pn.Column(r_h4, r_h4b, r_h4c)
r_h4e = pn.pane.PNG("images/T03_TP10.png", width=400)
spacer2=pn.Spacer(width=50)
pn.Row(r_h4d, spacer2, r_h4e)
```

```
Row
    [0] Column
        [0] Markdown(str, style={'font-size': '12pt'})
        [1] PNG(str, width=400)
        [2] Markdown(str, style={'font-size': '12pt'})
    [1] Spacer(width=50)
    [2] PNG(str, width=400)
```

```
#
fig, ax = plt.subplots(figsize=(8, 6))
plt.grid(axis='y', linestyle='--')
plt.xlim((0, 1800)); plt.ylim((0,0.7))
plt.xlabel("t(s)", fontsize=12 )
plt.ylabel(r"ln($\Delta h(0)/\Delta h(t)$)(-)", fontsize=12);
```




```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import panel as pn
pn.extension('katex')
```

2.13 Tutorial 4 - Effective K & Recitation

- **solutions for homework problems 1 – 4**
- **tutorial problems on effective conductivity and flow nets**
- **homework problems on effective conductivity and flow nets**

2.13.1 Solutions for Homework Problems 1 – 2

2.13.2 Homework Problems 1

```
#
r1_1 = pn.pane.Markdown("""
The pressure head in an aquifer extending over 200 km<sup>2</sup> is decreased by 1.
↪60 m.
Determine the loss of groundwater in the aquifer for two scenarios:
A. The aquifer is unconfined (storage coefficient 0.13).
B. The aquifer is confined (storage coefficient 0.0005).

""",width = 800, style={'font-size': '12pt'})

r1_2= pn.pane.PNG("images/T03_H1.png", width=350)
#r1_2 = pn.pane.PNG("images/T03_H1.PNG")

### Tutorial Problem 7 - Solution ###

#<img src="images/T03_H1.PNG" alt="Grosser Garten Map"  width="40%" height="100%" >

r1_3 = pn.pane.Markdown("""
### Solution - Homework Problem 1 ###
<br>
Relevant information can be found in Lecture L03, Slides- 28-30

""",width = 800, style={'font-size': '12pt'})

r1_3b = pn.pane.LaTeX(r"""
<br>
The relevant equations is:<br>
$$
S = \Delta V_w/(A\cdot \Delta H)
$$

""",width = 800, style={'font-size': '12pt'})
pn.Column(r1_1, r1_2, r1_3, r1_3b)
```

```
Column
    [0] Markdown(str, style={'font-size': '12pt'}, width=800)
    [1] PNG(str, width=350)
    [2] Markdown(str, style={'font-size': '12pt'}, width=800)
    [3] LaTeX(str, style={'font-size': '12pt'}, width=800)
```

```
# Given
A = 200 # km^2, aquifer area
D_h = 1.6*0.05 # m, head decrease
S_u = 0.13 # (-), Storativity unconfined aquifer
S_c = 0.0005 # (-) Storage coefficient, confined aquifer

# interim calculation
A_m2 = A*10**6 # m^2, aquifer area unit converted

# Solution
DV_wu = A_m2*S_u*D_h # m^3, change in water volume unconfined aquifer
DV_wc = A_m2*S_c*D_h  # m^3 change in water volume unconfined aquifer
```

(continues on next page)

(continued from previous page)

```
# output

print("Change in water volume in unconfined aquifer is: {0:1.1e}".format(DV_wu),"m\
↪u00b3 \n")
print("Change in water volume in confined aquifer is: {0:1.1e}".format(DV_wc),"m\u00b3
↪")
```

```
Change in water volume in unconfined aquifer is: 2.1e+06 m3

Change in water volume in confined aquifer is: 8.0e+03 m3
```

2.13.3 Homework Problem 2

Conduct a sieve analysis for a dried soil sample (see data in the table below)

1. Draw the granulometric curve (cumulative mass distribution) and briefly characterise the sediment with regard to its major constituent(s).
2. What is the coefficient of uniformity?

```
#
Head = ["mesh size [mm]", "residue in the sieve [g] ", " total", " / total"]
Size = [6.3, 2, 0.63, 0.2, 0.063, "< 0.063 /cup"]
residue = [11, 162, 88, 189, 42, 8]


data1= {"mesh size [mm]": Size, "residue in the sieve [g] ": residue}
df1= pd.DataFrame(data1)
df1.set_index("mesh size [mm]")
```

mesh size [mm]	residue in the sieve [g]
6.3	11
2	162
0.63	88
0.2	189
0.063	42
< 0.063 /cup	8

Solution of problem 2

```
# solution of problem 2

t_sample = np.sum(residue) # g, add the residue column to get total mass
retain_per = np.round(residue/t_sample *100,2) # %, # retain percentage residue/total␣
↪mass
retain_per_cumsum =np.round(np.cumsum(retain_per),2) # get the cummulative sum of the␣
↪reatined
passing_per = np.round(100 - retain_per_cumsum,3) # substract 100-cummsum to get␣
↪passing % - the last column

data2 = {"mesh size [mm]": Size, "residue in the sieve [g]": residue, "total %":␣
↪retain_per, "/total %": passing_per }
```

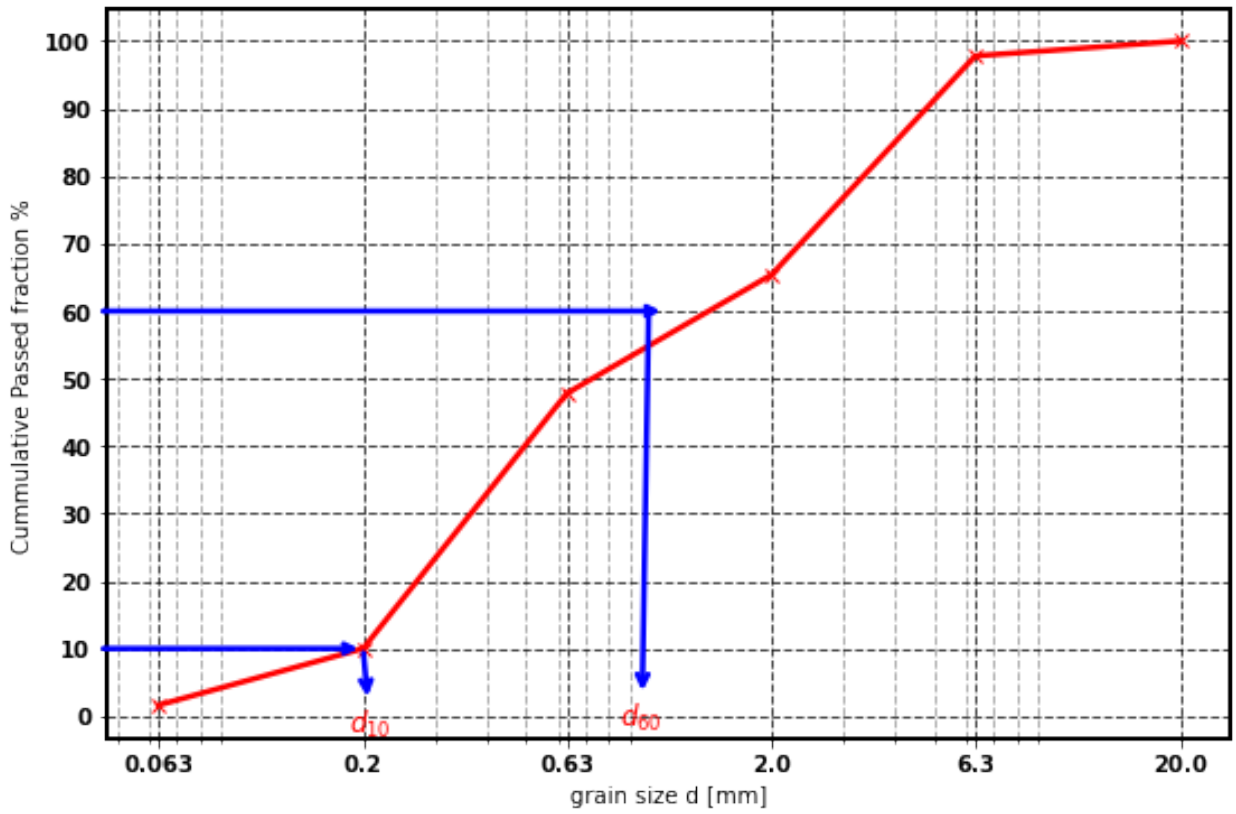
(continues on next page)

(continued from previous page)

```
df2= pd.DataFrame(data2)
df2 = df2.set_index("mesh size [mm]")
df2
```

mesh size [mm]	residue in the sieve [g]	total %	/total %
6.3	11	2.2	97.8
2	162	32.4	65.4
0.63	88	17.6	47.8
0.2	189	37.8	10.0
0.063	42	8.4	1.6
< 0.063 /cup	8	1.6	0.0

```
# Plotting granulometric curve

plt.rcParams['axes.linewidth']=2
plt.rcParams['grid.linestyle']='--'
plt.rcParams['grid.linewidth']=1

x = np.append([20], Size[:5]) # adding for all left over.
y = np.append([100],passing_per[:5])

fig = plt.figure(figsize=(9,6));

plt.plot(x, y, 'x-', color='red', lw=2.5);
tics=x.tolist()

plt.xscale('log');lw=2.5

plt.grid(which='major', color='k', alpha=0.7)
plt.grid(which='minor', color='k', alpha=0.3)
plt.xticks(x, tics);
plt.yticks(np.arange(0,110,10));

#plt.title('grain size distribution (combined wet sieving and sedimentation analysis)
↪');
plt.xlabel('grain size d [mm]');
plt.ylabel('Cummulative Passed fraction %');

plt.annotate('', xy=(0.20, 10),  xycoords='data', xytext=(0.045, 10),␣
↪arrowprops=dict(arrowstyle='->', color="b", lw=2.5),ha='right', va='top',)
plt.annotate('', xy=(1.1, 60),  xycoords='data', xytext=(0.045, 60),␣
↪arrowprops=dict(arrowstyle='->', color="b", lw=2.5),ha='right', va='top',)
plt.annotate(r'$d_{60}$', xy=(1, 60),  xycoords="data", xytext=(0.85, -3),color='red',
↪size=12, arrowprops=dict(arrowstyle='<-', color="b", lw=2.5),ha='left', va='bottom',
↪)
plt.annotate(r'$d_{10}$', xy=(0.20, 10),  xycoords='data', xytext=(0.235, 1.5),color=
↪'red',size=12, arrowprops=dict(arrowstyle='<-', color="b", lw=2.5),ha='right', va=
↪'top',)
plt.rcParams["font.weight"] = "bold"

plt.savefig("fig6.png", dpi=300)

mpl_pane = pn.pane.Matplotlib(fig)
```




```
# From the figure
d_10 = 0.22 # mm,approx, diameter 10% passing, see the arrow bottom in x-axis
d_60 = 1.0 # mm, approx diameter 10% passing, see the arrow bottom in x-axis

c_u = d_60/d_10 # [], coefficient of uniformity

#Output
print("\n The coefficient of uniformity is: {0:1.1f}".format(c_u), "\n")
r2_1 = pn.pane.Markdown("""
**Major constituents: coarse sand/medium sand** """, width=600, style={'font-size':
↪'12pt', 'color': 'blue'} )
pn.Row(r2_1)
```

```
 The coefficient of uniformity is: 4.5
```

```
Row
    [0] Markdown(str, style={'font-size': '12pt', ...}, width=600)
```

2.13.4 Tutorial Problems on effective conductivity

Tutorial problem 11

A sandy layer with a thickness of 2.5 m is embedded between two gravel layers. Both gravel layers have a thickness of 1.5 m and a hydraulic conductivity of 3.7·10-3 m/s. Steady-state groundwater flow is in parallel to the layering. A hydraulic gradient of 0.001 and an overall discharge of 1 m^3/d per unit width have been determined.

a. Determine the effective hydraulic conductivity. b. What is the hydraulic conductivity of the sand layer? c. Which effective hydraulic conductivity would be obtained if flow was assumed perpendicular to the layering? d. Calculate effective hydraulic conductivity if the angle between the flow direction and the layering equals 45°.

```
# Solution of Problem 11
r5_3 = pn.pane.PNG("images/T03_TP11_a.png", width=400)
r5_4 = pn.pane.LaTeX(r"""
Known relationships are (see Lecture 05, Slides 8-13, 22):
$$
Q = WmK\frac{\Delta H}{L}
$$
$$
K = \frac{Q/W}{m\cdot \Delta H \cdot L}
$$
Weighted arithmetic mean to determine hydraulic conductivity for sand:

$$
K = \frac{1}{m}\sum_{i=1}^n m_i\cdot K_i
$$

where $i$ is different layers
""", width = 500, style={'font-size': '12pt'})
spacer2 = pn.Spacer(width=100)

pn.Row(r5_3, spacer2, r5_4)
```

```
Row
    [0] PNG(str, width=400)
    [1] Spacer(width=100)
    [2] LaTeX(str, style={'font-size': '12pt'}, width=500)
```

```
#Given Solution of 11 a, b

Q = 1 # m^3/d, discharge
W = 1 # m, per unit width
K_g = 3.7*1E-3# m/s, conductivity of gravel layer
m_g = 1.5 # m, thickness of gravel layer
m_s = 2.5 # m, thickness of sand layer
Dh_L = 0.001 # (-), hydraulic gradient

#interim calculation
m = 2*m_g + m_s # m. total thickness of aquifer

#Solution of 11a
Keff_h = (Q/W)/(m*Dh_L) # m/d, conductivity
Keff_hs = Keff_h/(24*3600)# m/s, conductivity unit changed

#Solution of 11b
# K_eff = (2*m_g*K_g + m_s*K_s)/m # (Keff*m-2*m_g*K_g)/m_s
```

(continues on next page)

(continued from previous page)

```
K_s = ((m*Keff_hs - 2*m_g*K_g))/m_s  # m/s cond. of sand layer

print("\n")
print("Effective horizontal hydraulic conductivity (Keff_h) = {0:1.2f}".format(Keff_
↪h), "m/d\n" )
print("Effective horizontal hydraulic conductivity (Keff_hs) = {0:1.3E}".format(Keff_
↪hs), "m/s\n" )
print("Hydraulic conductivity of sand layer (K_s) = {0:1.1E}".format(K_s), "m/s\n" )
```

```
Effective horizontal hydraulic conductivity (Keff_h) = 181.82 m/d

Effective horizontal hydraulic conductivity (Keff_hs) = 2.104E-03 m/s

Hydraulic conductivity of sand layer (K_s) = 1.9E-04 m/s
```

```
#Given Solution of 11 c, d

r5_5 = pn.pane.PNG("images/T03_TP11_b.png", width=200)
r5_6 = pn.pane.PNG("images/T03_TP11_c.png", width=200)

r5_7 = pn.Column(r5_5, r5_6)

r5_8 = pn.pane.LaTeX(r"""
Vertical effective conductivity is given by weighted harmoninc mean

$$
K = \frac{m}{2\cdot \frac{m_g}{K_g} + \frac{m_s}{K_s} }
$$

<br>
For inclined aquider the effective conductivity is:

$$
K = \frac{1}{\frac{\cos^2\theta}{K_h} + \frac{\sin^2\theta}{K_v}}
$$

""", style={'font-size': '13pt'})

#pn.Row(r5_7,spacer2, r5_8)
```

```
# Solution of 11c

Keff_v = m/(2*(m_g/K_g)+ (m_s/K_s)) # m/s, conductivity K_v

#Given
theta = 45 # theta
theta_r = theta*(np.pi)/180 # degree to radian conversion
K_h = Keff_hs # m/s, solution from 11a
K_v = Keff_v # m/s, solution from 11c

# solution from 11d
Keff_i = 1/((np.cos(theta_r)**2/K_h)+(np.sin(theta_r)**2/K_v))

print("\n")
```

(continues on next page)

(continued from previous page)

```
print("Effective horizontal hydraulic conductivity (Keff_hs) = {0:1.2E}".format(Keff_
↪hs), "m/s\n" )
print("Effective vertical hydraulic conductivity (Keff_v) = {0:1.2E}".format(Keff_v),
↪"m/s\n" )
print("Effective inclined hydraulic conductivity (Keff_i) = {0:1.2E}".format(Keff_i),
↪"m/s\n" )
```

```
Effective horizontal hydraulic conductivity (Keff_hs) = 2.10E-03 m/s

Effective vertical hydraulic conductivity (Keff_v) = 3.93E-04 m/s

Effective inclined hydraulic conductivity (Keff_i) = 6.62E-04 m/s
```

```
r8_1= pn.pane.Markdown("""
#Homework Problems on  Effective Conductivity <br><br><br>
""", width = 800, style={'font-size': '12pt'})


r8_2= pn.pane.Markdown("""
#There is no obligation to solve homework problems!
""", width = 800, style={'font-size': '12pt', 'color':'red'})

pn.Column(r8_1,r8_2)
```

```
Column
    [0] Markdown(str, style={'font-size': '12pt'}, width=800)
    [1] Markdown(str, style={'font-size': '12pt', ...}, width=800)
```

```
#
r9_1= pn.pane.Markdown("""
## Homework Problem 5: Effective Hydraulic Conductivity ##
A gravel layer with a thickness of 2.5 m is embedded between two sand layers. Both␣
↪sand layers have a thickness of
1.5 m and a hydraulic conductivity of 3.7·10<sup>-4</sup> m/s. Steady-state␣
↪groundwater flow is perpendicular to the layering.
An overall head difference of 5.5 cm and a discharge of 500 l/d per unit area have␣
↪been determined <br><br>

**a.** Determine the effective hydraulic conductivity.<br><br>
**b.** What is the hydraulic conductivity of the gravel layer?<br><br>
**c.** Which effective hydraulic conductivity would be obtained if flow was assumed␣
↪to be in parallel with the layering?<br><br>
**d.** Calculate effective hydraulic conductivity if the angle between the flow␣
↪direction and the layering equals 30°. <br>

""", width = 900, style={'font-size': '12pt'})
r9_1
```

```
Markdown(str, style={'font-size': '12pt'}, width=900)
```

2.14 Tutorial 5 - Tutorial problems aquifer heterogeneity/anisotropy

```
import numpy as np
import matplotlib.pyplot as plt
import panel as pn
pn.extension('katex', 'mathjax')
```

2.14.1 Tutorial Problem 12: Hydrologic Triangle

```
r6_1 = pn.pane.Markdown("""
The figure below shows the position of four groundwater observation wells with␣
↪measured hydraulic heads in m a.s.l.
<br> <br>
**a.** Sketch head isolines for intervals of 1 m by applying the hydrologic triangle␣
↪method.<br><br>
**b.** Indicate the flow direction.

""",width = 400, style={'font-size': '13pt'})

r6_2 = pn.pane.PNG("images/T05_TP12_a.png", width=400)

spacer2 = pn.Spacer(width=100)


pn.Row(r6_1,spacer2, r6_2)
```

```
Row
    [0] Markdown(str, style={'font-size': '13pt'}, width=400)
    [1] Spacer(width=100)
    [2] PNG(str, width=400)
```

2.14.2 Solution of Tutotrial Problem 12

The following 4 steps are to be performed:

- **Step I** : Connects all the points
- **Step II** : Divide the connected lines at equal head-level (here = 1 m)
- **Step III** : Join all the equal head lines
- **Step IV** : Mark the flow direction from higher head towards lower head

```
# solution Tutorial problem 12
img_0 = pn.pane.PNG("images/T05_TP12_a.png", width=400)
img_1 = pn.pane.PNG("images/T05_TP12_b.png", width=500)
img_2 = pn.pane.PNG("images/T05_TP12_c.png", width=500)
img_3 = pn.pane.PNG("images/T05_TP12_d.png", width=500)
img_4 = pn.pane.PNG("images/T05_TP12_e.png", width=700)

tabs = pn.Tabs(('Given Head values', img_0), ('Step I: Connects points', img_1), (
↪"Step II: Divide line", img_2), ("Step III: Join lines", img_3), ("Step IV: Mark␣
↪flow direction", img_4))
tabs
```

```
Tabs
    [0] PNG(str, width=400)
    [1] PNG(str, width=500)
    [2] PNG(str, width=500)
    [3] PNG(str, width=500)
    [4] PNG(str, width=700)
```

2.14.3 Tutorial Problem 13: Flow Nets

```
#
r7_1 = pn.pane.Markdown("""


Sketch head isolines and streamlines for the two configurations a) and b) of a well␣
↪doublette shown below. In both cases flow nets should be sketched without and with␣
↪the uniform flow component.

""",width=800,  style={'font-size': '13pt'})

r7_2 = pn.pane.Markdown("""
 a) withdrawal at both wells:<br><br><br>
""",width=400,  style={'font-size': '13pt'})

r7_3 = pn.pane.PNG("images/T05_TP13_a.png", width=200)

r7_4 = pn.Column(r7_2,r7_3)

r7_5 = pn.pane.Markdown("""
 b) Injection and withdrawl wells:<br><br><br>
""",width=400,  style={'font-size': '13pt'})

r7_6 = pn.pane.PNG("images/T05_TP13_b.png", width=200)

spacer3 = pn.Spacer(width=200)

r7_7 = pn.Column( r7_5, r7_6)
r7_8 = pn.Row(r7_4, spacer3 , r7_7)
pn.Column(r7_1,  r7_8)
```

```
Column
    [0] Markdown(str, style={'font-size': '13pt'}, width=800)
    [1] Row
        [0] Column
            [0] Markdown(str, style={'font-size': '13pt'}, width=400)
            [1] PNG(str, width=200)
        [1] Spacer(width=200)
        [2] Column
            [0] Markdown(str, style={'font-size': '13pt'}, width=400)
            [1] PNG(str, width=200)
```

2.14.4 Solution of Tutorial Problem

This is to be sketched and demonstrated.

2.14.5 Tutorial Problem 14 (special topic)

From the laboratory test the degree of saturation(θ) of the unsaturated core (temperature = 9°C) sample was found to be 30% and relative permeability (k_r) is assumed to be 0.1. From the grain analysis the sample was determined to be predominantly medium sand (intrinsic permeability, $k = 1.61 \times 10^{-7}$ cm^2). Provided that density (ρ) and dynamic viscosity of water (μ) at 9°C is 999.73 kg/m^3 and 0.0013465 N·s/m^2 respectively, find the conductivity of the sample. What will be the conductivity of the same sample when the moisture content is 1% ($k_r \approx 0.001$) and 80% ($k_r \approx 0.4$). Explain the effect of moisture content on the sample.

2.14.6 Solution of Tutorial Problem 14

Lecture contents on the topic in L02- slides 02, 22 & 26

Hydraulic conductivity of the unsaturated sample ($\theta < 100\%$) can be obtained from the following expression:

$$K(\theta) = \left(\frac{k\rho g}{\mu}\right) k_r(\theta)$$

```
# Given
kr_30  = 0.1 # (-), relative permeability for moist. cont. 30%
i_p = 1.61 * 10**-7 # cm^2, intrinsic permeability
rho = 999.73 # kg/m^3, Sample density
mu = 0.0013467 #  N-s/m^2, dynamic visc.
g_c = 9.81 # N/kg, force unit used for gravitational constant

# Solutions 1
i_pm = i_p/10000 # m^2 unit conversion for int. permeab.
K_30 = (i_p*rho*g_c/mu)*kr_30

# Solution 2  when moisture content is 1% and 80%
kr_1 = 0.001 # (-), relative permeability for moist. cont. 1%
kr_80 = 0.4 # (-), relative permeability for moist. cont. 80%
K_1 = (i_p*rho*g_c/mu)*kr_1
K_80 = (i_p*rho*g_c/mu)*kr_80

# output

print("The conductivity of water when moisture content is 30%  is: {0:1.1e}".format(K_
↪30),"m/s \n")
print("The conductivity of water when moisture content is 1%  is: {0:1.1e}".format(K_
↪1),"m/s \n")
print("The conductivity of water when moisture content is 80%  is: {0:1.1e}".format(K_
↪80),"m/s \n")
print("The conductivity of media increases very rapidly with increase of moisture␣
↪content")
```

```
The conductivity of water when moisture content is 30%  is: 1.2e-01 m/s

The conductivity of water when moisture content is 1%  is: 1.2e-03 m/s

```

(continues on next page)

(continued from previous page)

```
The conductivity of water when moisture content is 80%  is: 4.7e-01 m/s

The conductivity of media increases very rapidly with increase of moisture content
```

2.14.7 Tutorial Problem 15

```
r2_2 = pn.pane.LaTeX(r"""

From the analysis of laboratory results the unsaturated hydraulic conductivity fits␣
↪the following exponential
model as a function of pressure head ($\psi$): $K(\psi) = K_s \exp(\alpha\cdot \psi)$,
↪
with $K_s$ [LT$^{-1}$] the saturated hydraulic conductivity and $\alpha$ [L$^{-1}$] a␣
↪fit parameter.
For the pressure head measurements and the data provided in the figure below, find
↪$K(\psi)$.
Also, find the Darcy velocity for this case.
""", width = 900, style={'font-size': '13pt'})

r2_3 =pn.pane.PNG("images/T05_TP15.png", width=400)


pn.Column(r2_2, r2_3)
```

```
Column
    [0] LaTeX(str, style={'font-size': '13pt'}, width=900)
    [1] PNG(str, width=400)
```

2.14.8 Solution Tutorial Problem 15

```
# Given

K_s = 2 # cm/d # saturated conductivity
al_a = 0.04 # 1/cm, fit constant
Ph_a = -100 # cm, pressure head at A
Ph_b = -90 # cm, pressure head at B
Z_a = 300 # cm, elevation head at A from datum
Z_b = 200 # cm, elevation head at B from datum

# Solution 1
Ph_m = (Ph_a+Ph_b)/2 # mean pressure head
K_psi = K_s*np.exp(al_a*Ph_m) # cm/d, from the given model

#Solution 2
H_A = Ph_a+Z_a # cm, hydraulic head at A
H_B = Ph_b+Z_b # cm, hydraulic head at B
dh_dz = (H_B - H_A)/(Z_b - Z_a) # (-), hydraulic head gradient
q_z = -K_psi*dh_dz # cm/d, Darcy velocity

print("The unsaturated conductiviy of the sample is: {0:1.3f}".format(K_psi), "cm/d\n
↪")
print("The Darcy velocity is: {0:1.3f}".format(q_z), "cm/d\n")
print("The negative sign indicates the direction opposite to increase in z.")
```

```
The unsaturated conductiviy of the sample is: 0.045 cm/d

The Darcy velocity is: -0.040 cm/d

The negative sign indicates the direction opposite to increase in z.
```

2.14.9 Homework Problems

Homework Problem 6: Hydrologic Triangle

```
r10_1= pn.pane.Markdown("""
The figure below shows the position of five groundwater observation wells with␣
↪measured hydraulic heads in m a.s.l.
 <br><br>

**a.** Sketch head isolines for intervals of 1 m by applying the hydrologic triangle␣
↪method.
<br><br>
**b.** Indicate the flow direction.<br><br>
""", width = 500, style={'font-size': '13pt'})
r10_2 = pn.pane.PNG("images/T05_TH6.png", width=400)

pn.Row(r10_1, r10_2)
```

```
Row
    [0] Markdown(str, style={'font-size': '13pt'}, width=500)
    [1] PNG(str, width=400)
```

Homework Problem 7: Flow Nets

```
#
r11_1= pn.pane.Markdown("""
Sketch head isolines and streamlines for the well doublette shown below.
In this case, injection and withdrawal of groundwater is superimposed to a uniform␣
↪flow component.
 <br><br><br><br><br><br>
 """, width = 900, style={'font-size': '13pt'})

r11_2 = pn.pane.PNG("images/T05_TH7.png", width=600)

r11_3= pn.pane.Markdown("""
 <br><br><br><br><br><br>
 """, width = 900, style={'font-size': '13pt'})

pn.Column(r11_1, r11_2, r11_3)
```

```
Column
    [0] Markdown(str, style={'font-size': '13pt'}, width=900)
    [1] PNG(str, width=600)
    [2] Markdown(str, style={'font-size': '13pt'}, width=900)
```

2.15 Simulating Mass Budget

(The contents presented in this section were re-developed principally by Dr. P. K. Yadav. The original tool, Spreadsheet based, was developed by Prof. Rudolf Liedl)

2.15.1 How to use the tool?

1. Go to the Binder by clicking the rocket button (top-right of the page)
2. Execute the code cell
3. Change the values of different quantities in the box.

This tool can also be downloaded and run locally. For that download the *deacy.ipynb* file and execute the process in any editor (e.g., JUPYTER notebook, JUPYTER lab) that is able to read and execute this file-type.

The code may also be executed in the book page.


```
# Used library
import numpy as np # for calculation
import matplotlib.pyplot as plt # for plots
import pandas as pd  # for table
import ipywidgets as widgets  # for widgets

# The main function

def mass_bal(n_simulation, MA, MB, MC, R_A, R_B):

    A = np.zeros(n_simulation) # creat an array with zros
    B = np.zeros(n_simulation)
    C = np.zeros(n_simulation)
    time  = np.arange(n_simulation)

    for i in range(0,n_simulation-1):
        A[0] = MA  # starting input value

        B[0] = MB
        C[0] = MC
        A[i+1] = A[i]-R_A*A[i]
        B[i+1] = B[i]+R_A*A[i]-R_B*B[i]
        C[i+1] = C[i]+R_B*B[i]
        summ = A[i]+B[i]+C[i]

    d = {"Mass_A": A, "Mass_B": B, "Mass_C": C, "Total Mass": summ}
    df = pd.DataFrame(d) # Generating result table
    label = ["Mass A (g)", "Mass B (g)", "Mass C (g)"]
    fig = plt.figure(figsize=(6,4))
    plt.plot(time, A, time, B, time, C, linewidth=3);  # plotting the results
    plt.xlabel("Time [Time Unit]"); plt.ylabel("Mass [g]") # placing axis labels
    plt.legend(label, loc=0);plt.grid(); plt.xlim([0,n_simulation]); plt.
↪ylim(bottom=0) # legends, grids, x,y limits
    plt.show() # display plot

    return print(df.round(2))
```

(continues on next page)

(continued from previous page)

```
# Widgets and interactive

N = widgets.BoundedIntText(value=20,min=0,max=100,step=1,description= '&Delta; t (day)
↪',disabled=False)

A = widgets.BoundedFloatText(value=100,min=0,max=1000.0,step=1,description='M<sub>A</
↪sub> (kg)',disabled=False)

B = widgets.BoundedFloatText(value=5,min=0,max=1000.0,step=1,description='M<sub>B</
↪sub> (kg)',disabled=False)

C = widgets.BoundedFloatText(value=10,min=0,max=1000,step=0.1,description='M<sub>C</
↪sub> (kg)',disabled=False)

RA = widgets.BoundedFloatText(value=0.2,min=0,max=100,step=0.1,description='R<sub>A</
↪sub> (day<sup>-1 </sup>)',disabled=False)

RB = widgets.BoundedFloatText(value=0.2,min=0,max=100,step=0.1,description='R<sub>B</
↪sub> (day<sup>-1 </sup>)',disabled=False)


interactive_plot = widgets.interactive(mass_bal, n_simulation = N, MA=A, MB=B, MC=C,␣
↪R_A=RA, R_B=RB,)
output = interactive_plot.children[-1]
#output.layout.height = '350px'
interactive_plot
```

```
interactive(children=(BoundedIntText(value=20, description='&Delta; t (day)'),␣
↪BoundedFloatText(value=100.0, d...
```

2.16 Simulating Seive Analysis

2.16.1 How to use the tool?

1. Go to the Binder by clicking the rocket button (top-right of the page)
2. Execute the code cell
3. Change the values of different quantities in the box and click the **run interact**.
4. From the resulting figure, using your mouse and selecting points in the figure obtain d10 and d60.
5. Execute the second code-cell and provide d10, d60 and temperature date
6. Click the exectute button.
7. For re-simulations - changes the input values in the boxes and click the "**run interact**" button.

This tool can also be downloaded and run locally. For that download the *deacy.ipynb* file and execute the process in any editor (e.g., JUPYTER notebook, JUPYTER lab) that is able to read and execute this file-type.

The code may also be executed in the book page.

```
# used Python library
import numpy as np # for calculation
import matplotlib.pyplot as plt  # for plotting
import pandas as pd  # for data table
import ipywidgets as widgets # for widgets
%matplotlib widget
import warnings; warnings.simplefilter('ignore')

print("Please provide the seive data in the boxes:  ")

def SA(mu, m1, m2, m3, m4, m1):
    dia = [6,2,0.6,0.2, 0.06, 0.01] # mm, diameter <0.06 (cup)= 0.01, >2 = 6
    mass = [mu, m1, m2, m3, m4, m1] # g, the residue in seive
    Total_mass = np.sum(mass)  # add the mass column to get total mass
    retain_per = np.round(mass/Total_mass*100,3)   # retain percentage
    retain_per_cumsum = np.round(np.cumsum(retain_per),3) # get the cummulative sum␣
↪of the reatined
    passing_per = np.round(100 - retain_per_cumsum, 3) # substract 100-cummsum to get␣
↪passing %
    data = {"mesh diameter [mm]": dia, "residue in the sieve [g]": mass, "total":␣
↪retain_per, "/total": passing_per }

    df1= pd.DataFrame(data)
    df1 = df1.set_index("mesh diameter [mm]")
    print(df1)

    plt.rcParams['axes.linewidth']=2
    #plt.rcParams["axes.edgecolor"]='white'
    plt.rcParams['grid.linestyle']='--'
    plt.rcParams['grid.linewidth']=1
    x = np.append([20],dia) # adding data to extend over 6 mm dia
    y = np.append([100],passing_per) # adding 100% to plot

    fig, ax = plt.subplots(figsize=(6,4))
    fig.canvas.header_visible = False
    plt.semilogx(x, y, 'x-', color='red')
    tics=x.tolist()

    ax.grid(which='major', color='k', alpha=0.7)
    ax.grid(which='minor', color='k', alpha=0.3)
    ax.set_xticks(x);
    ax.set_yticks(np.arange(0,110,10));
    plt.title('grain size distribution');
    plt.xlabel('grain size d [mm]');
    plt.ylabel('grain fraction < d ins % of total mass');
    ax.set_xlim(0, 30)
    from matplotlib.ticker import StrMethodFormatter
    ax.xaxis.set_major_formatter(StrMethodFormatter('{x:0.2f}'))


style = {'description_width': '200px'}

Inter=widgets.interact_manual(SA,
                        mu= widgets.FloatText(description="6 mm", style=style),
                        m1= widgets.FloatText(description="2 mm",style=style),
                        m2= widgets.FloatText(description="0.6 mm", style=style),
                        m3= widgets.FloatText(description="0.2 mm", style=style),
```

(continues on next page)

(continued from previous page)

```
                        m4= widgets.FloatText(description="0.06 mm", style=style),
                        ml= widgets.FloatText(description="0.01 mm", style=style))
```

```
Please provide the seive data in the boxes:
```

```
interactive(children=(FloatText(value=0.0, description='6 mm',␣
↪style=DescriptionStyle(description_width='200px...
```

The plot shown is interactive use the pointer and others tools in the graph to obtain d10 and d60 for the next step

```
def SA2(d10, d60, t):
    U = d60/d10
    K_h =  0.0116*(0.7+0.03*t)*d10**2
    print("\n The coefficient of non-uniformity: {0:0.2f}".format(U), "\n")
    print("The Hydraulic Conductivity based on Hazen Formula: {0:0.2e} m/s".format(K_
↪h))

style = {'description_width': '200px'}

Inter=widgets.interact_manual(SA2,
                        d10= widgets.FloatText(description="d10 (mm)", style=style),
                        d60= widgets.FloatText(description="d60 (mm)",style=style),
                        t= widgets.FloatText(description="Temperature (°C)",␣
↪style=style))
```

```
interactive(children=(FloatText(value=0.0, description='d10 (mm)',␣
↪style=DescriptionStyle(description_width='2...
```

2.17 Simulating Effective hydraulic conductivity

2.17.1 How to use the tool?

1. Go to the Binder by clicking the rocket button (top-right of the page)
2. Execute the code cell
3. Change the values of different quantities (layer thickness and corresponding conductivity) in the box and click the **run interact**.
4. For re-simulations - changes the input values in the boxes and click the "**run interact**" button.

This tool can also be downloaded and run locally. For that download the ***effective_K.ipynb*** file from the book GitHub site, and execute the process in any editor (e.g., JUPYTER notebook, JUPYTER lab) that is able to read and execute this file-type.

The code may also be executed in the book page. You may also run this tool (smartphone optimized) from: https://keff-app.herokuapp.com/

The codes are licensed under CC by 4.0 (use anyways, but acknowledge the original work)

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

(continues on next page)

(continued from previous page)

```
import pandas as pd
import ipywidgets as widgets
from ipywidgets import interact, interactive, fixed, interact_manual
plt.rcParams["font.weight"] = "bold"
plt.rcParams["font.size"] = 8
import warnings
warnings.filterwarnings('ignore')


def eff_K(M1, M2, M3, K1, K2, K3):

    K = [K1, K2, K3]
    K_f = ["%0.2e" %elem for elem in K]
    INPUT = {"Thickness [L]": [M1, M2, M3], "Hydraulic Conductivity [L/T]": K_f}
    index = ["Layer 1", "Layer 2", "Layer 3"]
    df = pd.DataFrame(INPUT, index=index)
    tt = M1+M2 + M3  # m, totial thickness

    # finding relative thickness,
    RL1, RL2, RL3 = M1/tt, M2/tt, M3/tt
    HRL1, HRL2, HRL3 = 1/K1, 1/K2, 1/K3
    WHK1, WHK2, WHK3 = RL1*K1, RL2*K2,RL3*K3
    WHR1,WHR2, WHR3 = RL1/K1, RL2/K2, RL3/K3

    # creating intermediate table
    RL =  [RL1, RL2, RL3]
    HRL = [HRL1, HRL2, HRL3]
    WHK = [WHK1, WHK2, WHK3]
    WHR = [WHR1,WHR2, WHR3]
    RL_f = [ '%.2f' %elem for elem in RL ]
    HRL_f = [ '%.2e' %elem for elem in HRL ]
    WHK_f = [ '%.2e' %elem for elem in WHK ]
    WHR_f = [ '%.2e' %elem for elem in WHR ]

    index2 = ["Layer 1", "Layer 2", "Layer 3", "Sum"]
    CAL1 = {"Relative Thickness [-]":RL_f, "Hydraulic Resistance [T/L]":HRL_f,
            "Weighted Hyd. Cond. [L/T]": WHK_f, "Weighted Hyd. Resistance [T/L]": WHR_
↪f}
    df2 = pd.DataFrame(CAL1)

    print("\n\n\033[1m Intermediate Calculations: \033[0m \n")
    print(df2, "\n")

    # calculations Parallel flow
    HR_eff = sum(WHR)
    HR_eff_a = max(WHR)

    HC_eff = 1/HR_eff
    HC_eff_a = 1/HR_eff_a

    RT1 = 0
    RT2 = RT1+RL1
    RT3 = RT2+RL2
    RT4 = 1

    RH1 = 1
    RH2 = 1-HC_eff*WHR1
```

(continues on next page)

(continued from previous page)

```
    RH3 = HC_eff*WHR3
    RH4 = 0

      # creating data table
    RH = [RH1, RH2, RH3, RH4]
    RH_f = ["%0.2f" %elem for elem in RH]
    RT = [RT1, RT2, RT3, RT4]
    RT_f = ["%0.2f" %elem for elem in RT] # 0.2f is for number format

    df3 = {"Relative Thickness [-]": RT_f, "Relative Head [-]": RH_f}
    df3 = pd.DataFrame(df3)


    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.set_xlim(0, 1.01); ax.set_ylim(0,1.01)
    ax.xaxis.set_ticks_position('top')
    ax.xaxis.set_label_position('top')
    ax.set_xlabel("Relative head [-]", fontsize=12)
    ax.set_ylabel("Relative thickness [-]", fontsize=12)
    plt.gca().invert_yaxis()
    ax.spines['right'].set_visible(False)
    ax.spines['bottom'].set_visible(False)

    ax.axhline(y=0, color='r', linewidth=2)
    ax.axhline(y=RT2, color='r', linewidth=2)
    ax.axhline(y=RT3, color='r', linewidth=2)
    ax.axhline(y=RT4, color='r', linewidth=2)
    ax.plot(RH, RT)

    plt.xticks(np.arange(0, 1.1, 0.1))
    plt.yticks(np.arange(0, 1.1, 0.1))

    print("\n\n\033[1m Parallel flow: \033[0m \n")

    print("The Effective Hydraulic Conductivity is: {0:0.2e}".format(HC_eff), "m/s\n")
    print("The Approximate Effective Hydraulic Conductivity is: {0:0.2e}".format(HC_
↪eff_a), "m/s\n")
    print("The Effective Hydraulic Resistance is: {0:0.2e}".format(HR_eff), "s/m\n")
    print("The Approximate Effective Hydraulic Resistance is {0:0.2e}".format(HR_eff_
↪a), "s/m\n")

    print(df3, "\n")
    plt.show(fig)

    # Perpendendicular flow

    WHK_eff = sum(WHK)
    WHK_eff_a = max(WHK)

    WHR_eff = 1/WHK_eff
    WHR_eff_a = 1/WHK_eff_a

    RD1 = WHK1/WHK_eff
    RD2 = WHK2/WHK_eff
    RD3 = WHK3/WHK_eff
```

(continues on next page)

(continued from previous page)

```
    RD = [RD1, RD2, RD3]
    RD_f = ["%0.2f" %elem for elem in RD]

    df4 = pd.DataFrame({"Relative Discharge [-]": RD_f}, index= index)

    fig2 = plt.figure()
    plt.gca().invert_yaxis()
    ay = fig2.add_subplot(1,1,1)
    ay.barh(index, RD)
    plt.xticks(np.arange(0, 1.1, 0.1))
    ay.set_xlabel("Relative discharge [-]", fontsize=12)
    ay.set_xlabel("Layer number", fontsize=12)



    print("\n\033[1m Perpendicular flow: \033[0m \n")

    print("The Effective Hydraulic Conductivity is: {0:0.2e}".format(WHK_eff), "s/m \n
↪")
    print("The Approximate Effective Hydraulic Conductivity is {0:0.2e}".format(WHK_
↪eff_a), "s/m\n")
    print("The Effective Hydraulic Resistance is: {0:0.2e}".format(WHR_eff), "m/s\n")
    print("The Approximate Effective Hydraulic Resistance is: {0:0.2e}".format(WHR_
↪eff_a), "m/s\n\n")

    print(df4, "\n")
    plt.show(fig2)

style = {'description_width': 'initial'}
Inter=widgets.interact_manual(eff_K,
                      M1= widgets.FloatText(description="Layer Thickness 1",␣
↪style=style),
                      K1= widgets.FloatText(description="Hydraulic Conductivity 1",
↪style=style),
                      M2= widgets.FloatText(description="Layer Thickness 2",␣
↪style=style),
                      K2= widgets.FloatText(description="Hydraulic Conductivity 2",␣
↪style=style),
                      M3= widgets.FloatText(description="Layer Thickness 3",␣
↪style=style),
                      K3= widgets.FloatText(description="Hydraulic Conductivity 3",␣
↪style=style))
```

```
interactive(children=(FloatText(value=0.0, description='Layer Thickness 1',␣
↪style=DescriptionStyle(description...
```

2.18 Uniform Flow and Well*

The worksheet addresses the superposition of uniform and radial steady-state groundwater flow in a homogeneous, confined aquifer of uniform thickness without recharge. The radial flow component may represent an extraction or injection well.

The worksheet calculates hydraulic head isolines (red), streamlines (blue / black), and isochrones (green) by using an analytical solution. The set of streamlines includes the dividing streamline (black). In addition, the capture width of the well (dashed lines) and the position of the stagnation point are determined. Three travel time values representing isochrones can be selected by the user.

input parameters	units	remarks
hydraulic conductivity	m/s	enter positive number
effective porosity	-	enter number between 0 and 1
thickness	mm/a	enter positive number
uniform velocity	m/d	enter number different from zero*
pumping rate	m^3/d	enter number different from zero**
travel time	d	enter positive number

* Positive or negative numbers correspond to uniform flow in parallel with or antiparallel to the x-axis, resp. ** Positive or negative numbers correspond to water extraction or injection, resp.

Contributed by Ms. Anne Pförtner and Sophie Pförtner. The original concept from Prof. R. Liedl spreasheet code.

```
import numpy as np
from ipywidgets import *
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

#definition of the function
def uniform_flow(K, ne, m, v, Q, t1, t2, t3):

    #intermediate results
    K_1 = K*86400                          #hydraulic conductivity [m/d]
    capture_width = Q/m/v                  #capture width [m]
    L_ref = Q/(2*np.pi*np.exp(1)*m*v)      #[m]
    h_ref = v/K_1*L_ref                    #[m]
    t_ref = 0.5*ne*Q/np.pi/m/v**2          #[d]
    stagnation_x = np.exp(1)*L_ref         #stagnation point (x) [m]
    stagnation_y = 0                       #stagnation point (y) [m]

    #isolines; Syntax: isolines_[X or Y]_plot_[h_ref]_[optional: 1 or 2]
    isolines_x_plot_n5_1=[]
    isolines_y_plot_n5_1=[]
    isolines_x_plot_n5_2 = []
    isolines_y_plot_n5_2 = []
    isolines_x_plot_n2_5_1 = []
    isolines_y_plot_n2_5_1 = []
    isolines_x_plot_n2_5_2 = []
    isolines_y_plot_n2_5_2 = []
    isolines_x_plot_0_1 = []
    isolines_y_plot_0_1 = []
    isolines_x_plot_0_2 = []
    isolines_y_plot_0_2 = []
```

(continues on next page)

(continued from previous page)

```
    isolines_x_plot_2_5 = []
    isolines_y_plot_2_5 = []
    isolines_x_plot_5 = []
    isolines_y_plot_5 = []
    isolines_x_plot_7_5 = []
    isolines_y_plot_7_5 = []
    isolines_x_plot_10 = []
    isolines_y_plot_10 = []
    isolines_x_plot_12_5 = []
    isolines_y_plot_12_5 = []
    isolines_x_plot_15 = []
    isolines_y_plot_15 = []


    for x in range(0, 100):
        isolines_x_n5_1=L_ref*((x*0.169103048517306+(100-x)*-0.150360933444141)/100)
        isolines_x_n5_2=L_ref*((x*12.35+(100-x)*11.6815653622516)/100)
        isolines_x_n2_5_1=L_ref*((x*0.474722923528955+(100-x)*-0.350418198256065)/100)
        isolines_x_n2_5_2=L_ref*((x*9.45+(100-x)*8.22933315122817)/100)
        isolines_x_0_1=L_ref*((x*np.exp(1)+(100-x)*-0.75695357132717)/100)
        isolines_x_0_2=L_ref*((x*6.65+(100-x)*np.exp(1))/100)
        isolines_x_2_5=L_ref*((x*4+(100-x)*-1.46395392968976)/100)
        isolines_x_5=L_ref*((x*1.5+(100-x)*-2.50444142220744)/100)
        isolines_x_7_5=L_ref*((x*-0.875+(100-x)*-3.84154019983304)/100)
        isolines_x_10=L_ref*((x*-3.15+(100-x)*-5.4105773228373)/100)
        isolines_x_12_5=L_ref*((x*-5.4+(100-x)*-7.15205676143326)/100)
        isolines_x_15=L_ref*((x*-7.65+(100-x)*-9.02099613666581)/100)

        if x == 0:
            isolines_y_n5_1 = 0
            isolines_y_n5_2 = 0
            isolines_y_n2_5_1 = 0
            isolines_y_n2_5_2 = 0
            isolines_y_0_1 = 0
            isolines_y_0_2 = 0
            isolines_y_2_5 = 0
            isolines_y_5 = 0
            isolines_y_7_5 = 0
            isolines_y_10 = 0
            isolines_y_12_5 = 0
            isolines_y_15 = 0

        else:
            isolines_y_n5_1 = np.sqrt((L_ref*np.exp(-5/np.exp(1)+isolines_x_n5_1/L_
↪ref/np.exp(1)))**2-isolines_x_n5_1**2)
            isolines_y_n5_2 = np.sqrt((L_ref*np.exp(-5/np.exp(1)+isolines_x_n5_2/L_
↪ref/np.exp(1)))**2-isolines_x_n5_2**2)
            isolines_y_n2_5_1 = np.sqrt((L_ref*np.exp(-2.5/np.exp(1)+isolines_x_n2_5_
↪1/L_ref/np.exp(1)))**2-isolines_x_n2_5_1**2)
            isolines_y_n2_5_2 = np.sqrt((L_ref*np.exp(-2.5/np.exp(1)+isolines_x_n2_5_
↪2/L_ref/np.exp(1)))**2-isolines_x_n2_5_2**2)
            isolines_y_0_1 = np.sqrt((L_ref*np.exp(0/np.exp(1)+isolines_x_0_1/L_ref/
↪np.exp(1)))**2-isolines_x_0_1**2)
            isolines_y_0_2 = np.sqrt((L_ref*np.exp(0/np.exp(1)+isolines_x_0_2/L_ref/
↪np.exp(1)))**2-isolines_x_0_2**2)
            isolines_y_2_5 = np.sqrt((L_ref*np.exp(2.5/np.exp(1)+isolines_x_2_5/L_ref/
↪np.exp(1)))**2-isolines_x_2_5**2)
```

(continues on next page)

(continued from previous page)

```
            isolines_y_5 = np.sqrt((L_ref*np.exp(5/np.exp(1)+isolines_x_5/L_ref/np.
↪exp(1)))**2-isolines_x_5**2)
            isolines_y_7_5 = np.sqrt((L_ref*np.exp(7.5/np.exp(1)+isolines_x_7_5/L_ref/
↪np.exp(1)))**2-isolines_x_7_5**2)
            isolines_y_10 = np.sqrt((L_ref*np.exp(10/np.exp(1)+isolines_x_10/L_ref/np.
↪exp(1)))**2-isolines_x_10**2)
            isolines_y_12_5 = np.sqrt((L_ref*np.exp(12.5/np.exp(1)+isolines_x_12_5/L_
↪ref/np.exp(1)))**2-isolines_x_12_5**2)
            isolines_y_15 = np.sqrt((L_ref*np.exp(15/np.exp(1)+isolines_x_15/L_ref/np.
↪exp(1)))**2-isolines_x_15**2)

        isolines_x_plot_n5_1.append(isolines_x_n5_1)
        isolines_y_plot_n5_1.append(isolines_y_n5_1)
        isolines_x_plot_n5_2.append(isolines_x_n5_2)
        isolines_y_plot_n5_2.append(isolines_y_n5_2)
        isolines_x_plot_n2_5_1.append(isolines_x_n2_5_1)
        isolines_y_plot_n2_5_1.append(isolines_y_n2_5_1)
        isolines_x_plot_n2_5_2.append(isolines_x_n2_5_2)
        isolines_y_plot_n2_5_2.append(isolines_y_n2_5_2)
        isolines_x_plot_0_1.append(isolines_x_0_1)
        isolines_y_plot_0_1.append(isolines_y_0_1)
        isolines_x_plot_0_2.append(isolines_x_0_2)
        isolines_y_plot_0_2.append(isolines_y_0_2)
        isolines_x_plot_2_5.append(isolines_x_2_5)
        isolines_y_plot_2_5.append(isolines_y_2_5)
        isolines_x_plot_5.append(isolines_x_5)
        isolines_y_plot_5.append(isolines_y_5)
        isolines_x_plot_7_5.append(isolines_x_7_5)
        isolines_y_plot_7_5.append(isolines_y_7_5)
        isolines_x_plot_10.append(isolines_x_10)
        isolines_y_plot_10.append(isolines_y_10)
        isolines_x_plot_12_5.append(isolines_x_12_5)
        isolines_y_plot_12_5.append(isolines_y_12_5)
        isolines_x_plot_15.append(isolines_x_15)
        isolines_y_plot_15.append(isolines_y_15)

    #streamlines; synthax: streamlines_[X or Y]_plot_[psi]
    streamlines_x_plot_0 = [0, (L_ref*-10)]
    streamlines_y_plot_0 = [0, 0]
    streamlines_x_plot_0_2 = []
    streamlines_y_plot_0_2 = []
    streamlines_x_plot_0_4 = []
    streamlines_y_plot_0_4 = []
    streamlines_x_plot_0_6 = []
    streamlines_y_plot_0_6 = []
    streamlines_x_plot_0_8 = []
    streamlines_y_plot_0_8 = []
    streamlines_x_plot_1 = []
    streamlines_y_plot_1 = []
    streamlines_x_plot_1_2 = []
    streamlines_y_plot_1_2 = []
    streamlines_x_plot_1_4 = []
    streamlines_y_plot_1_4 = []
    streamlines_x_plot_1_6 = []
    streamlines_y_plot_1_6 = []

    for x in range(0,100):
```

(continues on next page)

(continued from previous page)

```
        streamlines_y_0_2 = L_ref*((x*0+(100-x)*1.34462005667342)/100)
        streamlines_y_0_4 = L_ref*((x*0+(100-x)*2.6992421745751)/100)
        streamlines_y_0_6 = L_ref*((x*0+(100-x)*4.07255559164565)/100)
        streamlines_y_0_8 = L_ref*((x*0+(100-x)*5.47097889806004)/100)
        streamlines_y_1 = L_ref*((x*0+(100-x)*6.89826117541355)/100)
        streamlines_y_1_2 = L_ref*((x*2.13413758353342+(100-x)*8.35561532789609)/100)
        streamlines_y_1_4 = L_ref*((x*4.24381382643103+(100-x)*9.8422946888304)/100)
        streamlines_y_1_6 = L_ref*((x*6.31283612436048+(100-x)*11.3562442221618)/100)

        streamlines_x_0_2 = streamlines_y_0_2/np.tan(streamlines_y_0_2/L_ref/np.
↪exp(1)-np.pi*0.2)
        streamlines_x_0_4 = streamlines_y_0_4/np.tan(streamlines_y_0_4/L_ref/np.
↪exp(1)-np.pi*0.4)
        streamlines_x_0_6 = streamlines_y_0_6/np.tan(streamlines_y_0_6/L_ref/np.
↪exp(1)-np.pi*0.6)
        streamlines_x_0_8 = streamlines_y_0_8/np.tan(streamlines_y_0_8/L_ref/np.
↪exp(1)-np.pi*0.8)
        streamlines_x_1 = streamlines_y_1/np.tan(streamlines_y_1/L_ref/np.exp(1)-np.
↪pi*1)
        streamlines_x_1_2 = streamlines_y_1_2/np.tan(streamlines_y_1_2/L_ref/np.
↪exp(1)-np.pi*1.2)
        streamlines_x_1_4 = streamlines_y_1_4/np.tan(streamlines_y_1_4/L_ref/np.
↪exp(1)-np.pi*1.4)
        streamlines_x_1_6 = streamlines_y_1_6/np.tan(streamlines_y_1_6/L_ref/np.
↪exp(1)-np.pi*1.6)

        streamlines_x_plot_0_2.append(streamlines_x_0_2)
        streamlines_y_plot_0_2.append(streamlines_y_0_2)
        streamlines_x_plot_0_4.append(streamlines_x_0_4)
        streamlines_y_plot_0_4.append(streamlines_y_0_4)
        streamlines_x_plot_0_6.append(streamlines_x_0_6)
        streamlines_y_plot_0_6.append(streamlines_y_0_6)
        streamlines_x_plot_0_8.append(streamlines_x_0_8)
        streamlines_y_plot_0_8.append(streamlines_y_0_8)
        streamlines_x_plot_1.append(streamlines_x_1)
        streamlines_y_plot_1.append(streamlines_y_1)
        streamlines_x_plot_1_2.append(streamlines_x_1_2)
        streamlines_y_plot_1_2.append(streamlines_y_1_2)
        streamlines_x_plot_1_4.append(streamlines_x_1_4)
        streamlines_y_plot_1_4.append(streamlines_y_1_4)
        streamlines_x_plot_1_6.append(streamlines_x_1_6)
        streamlines_y_plot_1_6.append(streamlines_y_1_6)

    #isochrones
    isochrones_x_plot_t1 = []
    isochrones_y_plot_t1 = []
    isochrones_x_plot_t2 = []
    isochrones_y_plot_t2 = []
    isochrones_x_plot_t3 = []
    isochrones_y_plot_t3 = []

    #iterate 5 times with start value t_xmin1/ t_xmax1

    t1_xmin1=-np.exp(1)*np.sqrt(np.exp(2*(t1/t_ref))-1)
    t1_xmin6= t1_xmin1+(np.exp(1)-t1_xmin1)*(1+np.exp(1)/t1_xmin1*(np.log(1-t1_xmin1/
↪np.exp(1))+(t1/t_ref)))
    t1_xmax1= np.exp(1)*np.sqrt(1-np.exp(-2*(t1/t_ref)))
```

(continues on next page)

(continued from previous page)

```
    t1_xmax6= t1_xmax1+(np.exp(1)-t1_xmax1)*(1+np.exp(1)/t1_xmax1*(np.log(1-t1_xmax1/
↪np.exp(1))+(t1/t_ref)))
    t2_xmin1=-np.exp(1)*np.sqrt(np.exp(2*(t2/t_ref))-1)
    t2_xmin6= t2_xmin1+(np.exp(1)-t2_xmin1)*(1+np.exp(1)/t2_xmin1*(np.log(1-t2_xmin1/
↪np.exp(1))+(t2/t_ref)))
    t2_xmax1=np.exp(1)*np.sqrt(1-np.exp(-2*(t2/t_ref)))
    t2_xmax6= t2_xmax1+(np.exp(1)-t2_xmax1)*(1+np.exp(1)/t2_xmax1*(np.log(1-t2_xmax1/
↪np.exp(1))+(t2/t_ref)))
    t3_xmin1=-np.exp(1)*np.sqrt(np.exp(2*(t3/t_ref))-1)
    t3_xmin6= t3_xmin1+(np.exp(1)-t3_xmin1)*(1+np.exp(1)/t3_xmin1*(np.log(1-t3_xmin1/
↪np.exp(1))+(t3/t_ref)))
    t3_xmax1=np.exp(1)*np.sqrt(1-np.exp(-2*(t3/t_ref)))
    t3_xmax6= t3_xmax1+(np.exp(1)-t3_xmax1)*(1+np.exp(1)/t3_xmax1*(np.log(1-t3_xmax1/
↪np.exp(1))+(t3/t_ref)))
    for i in range(4):
        t1_xmin6= t1_xmin6+(np.exp(1)-t1_xmin6)*(1+np.exp(1)/t1_xmin6*(np.log(1-t1_
↪xmin6/np.exp(1))+(t1/t_ref)))
        t1_xmax6= t1_xmax6+(np.exp(1)-t1_xmax6)*(1+np.exp(1)/t1_xmax6*(np.log(1-t1_
↪xmax6/np.exp(1))+(t1/t_ref)))
        t2_xmin6= t2_xmin6+(np.exp(1)-t2_xmin6)*(1+np.exp(1)/t2_xmin6*(np.log(1-t2_
↪xmin6/np.exp(1))+(t2/t_ref)))
        t2_xmax6= t2_xmax6+(np.exp(1)-t2_xmax6)*(1+np.exp(1)/t2_xmax6*(np.log(1-t2_
↪xmax6/np.exp(1))+(t2/t_ref)))
        t3_xmin6= t3_xmin6+(np.exp(1)-t3_xmin6)*(1+np.exp(1)/t3_xmin6*(np.log(1-t3_
↪xmin6/np.exp(1))+(t3/t_ref)))
        t3_xmax6= t3_xmax6+(np.exp(1)-t3_xmax6)*(1+np.exp(1)/t3_xmax6*(np.log(1-t3_
↪xmax6/np.exp(1))+(t3/t_ref)))

    for x in range (0,100):

        isochrones_x_t1 = 0.5*L_ref*(t1_xmin6+t1_xmax6+(t1_xmax6-t1_xmin6)*np.cos(np.
↪pi*(100-x)/100))
        isochrones_x_t2 = 0.5*L_ref*(t2_xmin6+t2_xmax6+(t2_xmax6-t2_xmin6)*np.cos(np.
↪pi*(100-x)/100))
        isochrones_x_t3 = 0.5*L_ref*(t3_xmin6+t3_xmax6+(t3_xmax6-t3_xmin6)*np.cos(np.
↪pi*(100-x)/100))

        if x == 0:
            isochrones_y_t1 = 0
            isochrones_y_t2 = 0
            isochrones_y_t3 = 0
        else:

            isochrones_y1_t1 = L_ref*np.exp(1)*np.arccos((0.5*isochrones_x_t1/np.
↪exp(1)/L_ref+np.exp(-(t1/t_ref)-isochrones_x_t1/np.exp(1)/L_ref))/(1-0.5*isochrones_
↪x_t1/np.exp(1)/L_ref))
            isochrones_y_t1 = L_ref*np.exp(1)*np.arccos((isochrones_x_t1/L_ref*(np.
↪sin(isochrones_y1_t1/np.exp(1)/L_ref)/(isochrones_y1_t1/L_ref)-0.5*np.
↪cos(isochrones_y1_t1/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t1/t_ref)-isochrones_x_t1/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t1/np.exp(1)/L_ref))

            isochrones_y1_t2 = L_ref*np.exp(1)*np.arccos((0.5*isochrones_x_t2/np.
↪exp(1)/L_ref+np.exp(-(t2/t_ref)-isochrones_x_t2/np.exp(1)/L_ref))/(1-0.5*isochrones_
↪x_t2/np.exp(1)/L_ref))
            isochrones_y_t2 = L_ref*np.exp(1)*np.arccos((isochrones_x_t2/L_ref*(np.
↪sin(isochrones_y1_t2/np.exp(1)/L_ref)/(isochrones_y1_t2/L_ref)-0.5*np.
↪cos(isochrones_y1_t2/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t2/t_ref)-isochrones_x_t2/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t2/np.exp(1)/L_ref))
```

(continues on next page)

(continued from previous page)

```
            isochrones_y1_t3 = L_ref*np.exp(1)*np.arccos((0.5*isochrones_x_t3/np.
↪exp(1)/L_ref+np.exp(-(t3/t_ref)-isochrones_x_t3/np.exp(1)/L_ref))/(1-0.5*isochrones_
↪x_t3/np.exp(1)/L_ref))
            isochrones_y_t3 = L_ref*np.exp(1)*np.arccos((isochrones_x_t3/L_ref*(np.
↪sin(isochrones_y1_t3/np.exp(1)/L_ref)/(isochrones_y_t3/L_ref)-0.5*np.cos(isochrones_
↪y1_t3/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t3/t_ref)-isochrones_x_t3/np.exp(1)/L_
↪ref))/(1-0.5*isochrones_x_t3/np.exp(1)/L_ref))

            for i in range(4):
                isochrones_y_t1 = L_ref*np.exp(1)*np.arccos((isochrones_x_t1/L_
↪ref*(np.sin(isochrones_y_t1/np.exp(1)/L_ref)/(isochrones_y_t1/L_ref)-0.5*np.
↪cos(isochrones_y_t1/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t1/t_ref)-isochrones_x_t1/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t1/np.exp(1)/L_ref))

                isochrones_y_t2 = L_ref*np.exp(1)*np.arccos((isochrones_x_t2/L_
↪ref*(np.sin(isochrones_y_t2/np.exp(1)/L_ref)/(isochrones_y_t2/L_ref)-0.5*np.
↪cos(isochrones_y_t2/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t2/t_ref)-isochrones_x_t2/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t2/np.exp(1)/L_ref))

                isochrones_y_t3 = L_ref*np.exp(1)*np.arccos((isochrones_x_t3/L_
↪ref*(np.sin(isochrones_y_t3/np.exp(1)/L_ref)/(isochrones_y_t3/L_ref)-0.5*np.
↪cos(isochrones_y_t3/np.exp(1)/L_ref)/np.exp(1))+np.exp(-(t3/t_ref)-isochrones_x_t3/
↪np.exp(1)/L_ref))/(1-0.5*isochrones_x_t3/np.exp(1)/L_ref))


        isochrones_x_plot_t1.append(isochrones_x_t1)
        isochrones_y_plot_t1.append(isochrones_y_t1)
        isochrones_x_plot_t2.append(isochrones_x_t2)
        isochrones_y_plot_t2.append(isochrones_y_t2)
        isochrones_x_plot_t3.append(isochrones_x_t3)
        isochrones_y_plot_t3.append(isochrones_y_t3)

    #still necessary: mirror on x-axis
    isolines_y_plot_n5_1_mirror = -1*(np.asarray(isolines_y_plot_n5_1))
    isolines_y_plot_n5_2_mirror = -1*(np.asarray(isolines_y_plot_n5_2))
    isolines_y_plot_n2_5_1_mirror = -1*(np.asarray(isolines_y_plot_n2_5_1))
    isolines_y_plot_n2_5_2_mirror = -1*(np.asarray(isolines_y_plot_n2_5_2))
    isolines_y_plot_0_1_mirror = -1*(np.asarray(isolines_y_plot_0_1))
    isolines_y_plot_0_2_mirror = -1*(np.asarray(isolines_y_plot_0_2))
    isolines_y_plot_2_5_mirror = -1*(np.asarray(isolines_y_plot_2_5))
    isolines_y_plot_5_mirror = -1*(np.asarray(isolines_y_plot_5))
    isolines_y_plot_7_5_mirror = -1*(np.asarray(isolines_y_plot_7_5))
    isolines_y_plot_10_mirror = -1*(np.asarray(isolines_y_plot_10))
    isolines_y_plot_12_5_mirror = -1*(np.asarray(isolines_y_plot_12_5))
    isolines_y_plot_15_mirror = -1*(np.asarray(isolines_y_plot_15))

    streamlines_y_plot_0_2_mirror = -1*(np.asarray(streamlines_y_plot_0_2))
    streamlines_y_plot_0_4_mirror = -1*(np.asarray(streamlines_y_plot_0_4))
    streamlines_y_plot_0_6_mirror = -1*(np.asarray(streamlines_y_plot_0_6))
    streamlines_y_plot_0_8_mirror = -1*(np.asarray(streamlines_y_plot_0_8))
    streamlines_y_plot_1_mirror = -1*(np.asarray(streamlines_y_plot_1))
    streamlines_y_plot_1_2_mirror = -1*(np.asarray(streamlines_y_plot_1_2))
    streamlines_y_plot_1_4_mirror = -1*(np.asarray(streamlines_y_plot_1_4))
    streamlines_y_plot_1_6_mirror = -1*(np.asarray(streamlines_y_plot_1_6))
```

(continues on next page)

(continued from previous page)

```
    isochrones_y_plot_t1_mirror = -1*(np.asarray(isochrones_y_plot_t1))
    isochrones_y_plot_t2_mirror = -1*(np.asarray(isochrones_y_plot_t2))
    isochrones_y_plot_t3_mirror = -1*(np.asarray(isochrones_y_plot_t3))


    fig, (ax1, ax2) = plt.subplots(2, figsize=(10, 10))

    #plotten incl. mirror on x-axis

    ax1.plot(isolines_x_plot_n5_1, isolines_y_plot_n5_1, 'r')
    ax1.plot(isolines_x_plot_n5_2, isolines_y_plot_n5_2, 'r')
    ax1.plot(isolines_x_plot_n2_5_1, isolines_y_plot_n2_5_1, 'r')
    ax1.plot(isolines_x_plot_n2_5_2, isolines_y_plot_n2_5_2, 'r')
    ax1.plot(isolines_x_plot_0_1, isolines_y_plot_0_1, 'r')
    ax1.plot(isolines_x_plot_0_2, isolines_y_plot_0_2, 'r')
    ax1.plot(isolines_x_plot_2_5, isolines_y_plot_2_5, 'r')
    ax1.plot(isolines_x_plot_5, isolines_y_plot_5, 'r')
    ax1.plot(isolines_x_plot_7_5, isolines_y_plot_7_5, 'r')
    ax1.plot(isolines_x_plot_10, isolines_y_plot_10, 'r')
    ax1.plot(isolines_x_plot_12_5, isolines_y_plot_12_5, 'r')
    ax1.plot(isolines_x_plot_15, isolines_y_plot_15, 'r')

    ax1.plot(isolines_x_plot_n5_1, isolines_y_plot_n5_1_mirror, 'r')
    ax1.plot(isolines_x_plot_n5_2, isolines_y_plot_n5_2_mirror, 'r')
    ax1.plot(isolines_x_plot_n2_5_1, isolines_y_plot_n2_5_1_mirror, 'r')
    ax1.plot(isolines_x_plot_n2_5_2, isolines_y_plot_n2_5_2_mirror, 'r')
    ax1.plot(isolines_x_plot_0_1, isolines_y_plot_0_1_mirror, 'r')
    ax1.plot(isolines_x_plot_0_2, isolines_y_plot_0_2_mirror, 'r')
    ax1.plot(isolines_x_plot_2_5, isolines_y_plot_2_5_mirror, 'r')
    ax1.plot(isolines_x_plot_5, isolines_y_plot_5_mirror, 'r')
    ax1.plot(isolines_x_plot_7_5, isolines_y_plot_7_5_mirror, 'r')
    ax1.plot(isolines_x_plot_10, isolines_y_plot_10_mirror, 'r')
    ax1.plot(isolines_x_plot_12_5, isolines_y_plot_12_5_mirror, 'r')
    ax1.plot(isolines_x_plot_15, isolines_y_plot_15_mirror, 'r')

    ax1.plot(streamlines_x_plot_0,streamlines_y_plot_0, 'b')
    ax1.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2, 'b')
    ax1.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4, 'b')
    ax1.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6, 'b')
    ax1.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8, 'b')
    ax1.plot(streamlines_x_plot_1,streamlines_y_plot_1, color = 'black')
    ax1.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2, 'b')
    ax1.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4, 'b')
    ax1.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6, 'b')


    ax1.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2_mirror, 'b')
    ax1.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4_mirror, 'b')
    ax1.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6_mirror, 'b')
    ax1.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8_mirror, 'b')
    ax1.plot(streamlines_x_plot_1,streamlines_y_plot_1_mirror, color = 'black')
    ax1.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2_mirror, 'b')
    ax1.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4_mirror, 'b')
    ax1.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6_mirror, 'b')


    ax1.set(xlabel='x [m]', ylabel ='y [m]', xlim = [-175, 225], ylim = [-175,175])
```

(continues on next page)

(continued from previous page)

```
    fig.savefig("isolines.png", dpi=300)

    ax2.plot(isochrones_x_plot_t1, isochrones_y_plot_t1, 'g')
    ax2.plot(isochrones_x_plot_t2, isochrones_y_plot_t2, 'g')
    ax2.plot(isochrones_x_plot_t3, isochrones_y_plot_t3, 'g')

    ax2.plot(isochrones_x_plot_t1, isochrones_y_plot_t1_mirror, 'g')
    ax2.plot(isochrones_x_plot_t2, isochrones_y_plot_t2_mirror, 'g')
    ax2.plot(isochrones_x_plot_t3, isochrones_y_plot_t3_mirror, 'g')

    ax2.plot(streamlines_x_plot_0,streamlines_y_plot_0, 'b')
    ax2.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2, 'b')
    ax2.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4, 'b')
    ax2.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6, 'b')
    ax2.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8, 'b')
    ax2.plot(streamlines_x_plot_1,streamlines_y_plot_1, color = 'black')
    ax2.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2, 'b')
    ax2.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4, 'b')
    ax2.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6, 'b')


    ax2.plot(streamlines_x_plot_0_2,streamlines_y_plot_0_2_mirror, 'b')
    ax2.plot(streamlines_x_plot_0_4,streamlines_y_plot_0_4_mirror, 'b')
    ax2.plot(streamlines_x_plot_0_6,streamlines_y_plot_0_6_mirror, 'b')
    ax2.plot(streamlines_x_plot_0_8,streamlines_y_plot_0_8_mirror, 'b')
    ax2.plot(streamlines_x_plot_1,streamlines_y_plot_1_mirror, color = 'black')
    ax2.plot(streamlines_x_plot_1_2,streamlines_y_plot_1_2_mirror, 'b')
    ax2.plot(streamlines_x_plot_1_4,streamlines_y_plot_1_4_mirror, 'b')
    ax2.plot(streamlines_x_plot_1_6,streamlines_y_plot_1_6_mirror, 'b')



    ax2.set(xlabel='x [m]', ylabel ='y [m]', xlim = [-175, 225], ylim = [-175,175])
    fig.savefig("isochrones.png", dpi=300)

interact(uniform_flow,
         K=widgets.FloatLogSlider(value=3e-4, base=10, min=-10, max=0, step=0.1,␣
↪description='hydraulic conductivity [m/s]:', disabled=False),
         ne=widgets.FloatSlider(value=0.2, min=0.001, max=1, step=0.05, description=
↪'effective porosity [-]:', disabled=False),
         m= widgets.FloatSlider(value=7,min=0, max=30,step=1, description='thickness␣
↪[m]:' , disabled=False),
         v=widgets.FloatSlider(value=0.4, min=0.00001, max=5, step=0.1, description=
↪'uniform velocity [m/d]:', disabled=False),
         Q=widgets.FloatSlider(value=800, min=100, max=1000, step=0.1, description=
↪'pumping rate [m^3/d]:', disabled=False),
         t1=widgets.BoundedFloatText(value=10, min=1, max=100, step=0.1, description=
↪'t1 [d]:', disabled=False),
         t2=widgets.BoundedFloatText(value=30, min=1, max=100, step=0.1, description=
↪'t2 [d]:', disabled=False),
         t3=widgets.BoundedFloatText(value=50, min=1, max=100, step=0.1, description=
↪'t3 [d]:', disabled=False),
         )
```

```
interactive(children=(FloatLogSlider(value=0.0003, description='hydraulic␣
↪conductivity [m/s]:', max=0.0, min=-...
```

```
<function __main__.uniform_flow(K, ne, m, v, Q, t1, t2, t3)>
```

2.19 Simulating the Anisotropy and flow direction

2.19.1 How to use the tool?

1. Go to the Binder by clicking the rocket button (top-right of the page)
2. Execute the code cell
3. Change the values of different quantities in the box.
4. For re-simulations - changes the input values in the boxes.

This tool can also be downloaded and run locally. For that download the *aniso2D.ipynb* file and execute the process in any editor (e.g., JUPYTER notebook, JUPYTER lab) that is able to read and execute this file-type.

The code may also be executed in the book page.

The codes are licensed under CC by 4.0 (use anyways, but acknowledge the original work)

```
# The library used

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from ipywidgets import widgets, interactive



# the main programme
def aniso(a_d, ani_r):

# interim calculation
    a_r = a_d*np.pi/180

    i_xr = np.cos(a_r) # (-), rel. hyd grad. along x
    i_zr = np.sin(a_r) # (-), rel. hyd grad. along z
    K_h = 1 # (-), m/s K_h
    K_v = 1/ani_r # m/s, rel K_v
    f_x = -i_xr*K_h # m/s
    f_z = -i_zr*K_v # m/s
    f_m = np.sqrt(f_x*f_x+f_z*f_z) # m/s

    args = (K_h*i_xr*i_xr + K_v*i_zr*i_zr)/f_m
    an_i_f = ((np.pi-np.arccos(args))*180/np.pi)# deg,

# plots axes

    fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(10,6), gridspec_kw={'width_ratios':␣
↪[3, 1]})

# points for gradient and flux

    grad_px = [0, i_xr]#i_xr
    grad_pz = [0, i_zr]#i_zr
```

(continues on next page)

(continued from previous page)

```
    flux_px = [0, f_x]
    flux_pz = [0, f_z]

# creating points for intersect lines (5 of them)

    p1=-0.5*i_zr; p2 = 0.5*i_zr; p3 =-0.5*i_zr+0.35*i_xr; p4 = 0.5*i_zr+0.35*i_xr; p5␣
↪= -0.5*i_zr+0.7*i_xr
    p6 = 0.5*i_zr+0.7*i_xr;p7 = -0.5*i_zr-0.35*i_xr; p8 = 0.5*i_zr-0.35*i_xr; p9 = -0.
↪5*i_zr-0.7*i_xr;p10 =0.5*i_zr-0.7*i_xr

    q1=0.5*i_xr; q2 =- 0.5*i_xr; q3 =0.5*i_xr+0.35*i_zr; q4 = -0.5*i_xr+0.35*i_zr; q5␣
↪= 0.5*i_xr+0.7*i_zr
    q6 = -0.5*i_xr+0.7*i_zr;q7 = 0.5*i_xr-0.35*i_zr;q8 = -0.5*i_xr-0.35*i_zr; q9 = 0.
↪5*i_xr-0.7*i_zr;q10 = -0.5*i_xr-0.7*i_zr

# plotted points
    l_1x =[p1, p2]; l_1y = [q1, q2]
    l_2x =[p3, p4]; l_2y = [q3, q4]
    l_3x =[p5, p6]; l_3y = [q5, q6]
    l_4x =[p7, p8]; l_4y = [q7, q8]
    l_5x =[p9, p10]; l_5y = [q9, q10]


# creating points for anisotropy
    r1 =1.05 if ani_r >= 1 else 1.5-0.45*ani_r
    r2 = 1.95 if ani_r >= 1 else 1.5+0.45*ani_r
    r3 = 0.5*(r1+r2); r4 = r3

    s1 = -0.5; s2 = s1
    s3 = -0.05 if ani_r<=1 else -0.5+0.45/ani_r
    s4 = -0.95 if ani_r<=1 else -0.5-0.45/ani_r

# plotted points
    Iso_1x = [r1, r2]; Iso_1y = [s1, s2]
    Iso_2x = [r3, r4]; Iso_2y = [s3, s4]
    Iso_x =[r1, r2, r3, r4]; Iso_y = [s1,s2,s3,s4]

# plotting all points

# plotting gradient/flux lines

    ax1.plot(grad_px, grad_pz, "g", label=" gradient") # plotting gradient
    ax1.plot(flux_px, flux_pz, "r", label=" flux") # plotting flux

# plotting intersect lines
    ax1.plot(l_1x, l_1y, "b", label = "head isoline")
    ax1.plot(l_2x, l_2y, "b")
    ax1.plot(l_3x, l_3y, "b")
    ax1.plot(l_4x, l_4y, "b")
    ax1.plot(l_5x, l_5y, "b")
    ax1.legend()


    ax1.spines['left'].set_position('center') # bring the axis lines in center
    ax1.spines['bottom'].set_position('center')
    ax1.spines['right'].set_color('none') # remove the top box
```

(continues on next page)

(continued from previous page)

```
    ax1.spines['top'].set_color('none')
    ax1.set_xticks([]);ax1.set_yticks([]); # remove the ticks
    ax1.set_title("Anisotropy flux and gradient", y=0, pad=-25, verticalalignment="top
↪")


# plotting Anisotropy
    ax2.plot(Iso_1x, Iso_1y, "k", label = r"$K_h: K_v$")
    ax2.plot(Iso_2x, Iso_2y, "k")
    ax2.legend(bbox_to_anchor=(-0.4, -0.05), loc='lower left')
    ax2.set_xlim(1, 2)
    ax2.set_ylim(-1, 1)
    ax2.axis('off')
    ax2.set_title("Anisotropy ratio", y=0, pad=-25, verticalalignment="top");

interactive(aniso,
         a_d=widgets.BoundedFloatText(value=45, min=0, max=360, step=0.5,␣
↪description=r'angle (°)', disabled=False),
         ani_r=widgets.BoundedIntText(value=1, min=1, max=100, step=1, description='K
↪<sub>h</sub>/K<sub>v</sub>', disabled=False))
```

```
interactive(children=(BoundedFloatText(value=45.0, description='angle (°)', max=360.0,
↪ step=0.5), BoundedIntTe...
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from xlrd import *
import ipysheet as ips
import panel as pn
%matplotlib inline
from scipy import stats
pn.extension('katex')
```

2.20 Groundwater Exam Solution - 2019-2020

(The contents presented in this section were re-developed principally by Dr. P. K. Yadav with supervision from Prof. Rudolf Liedl)

Q1. Aquifer Types (ca. 5 pts.)

a. Differentiate between Aquifer, Aquitard and Aquiclude (3 points)

b. Schematically present a confined aquifer (vertical cross-section) providing essential features with their legends (2 points)

Solution 1. a.

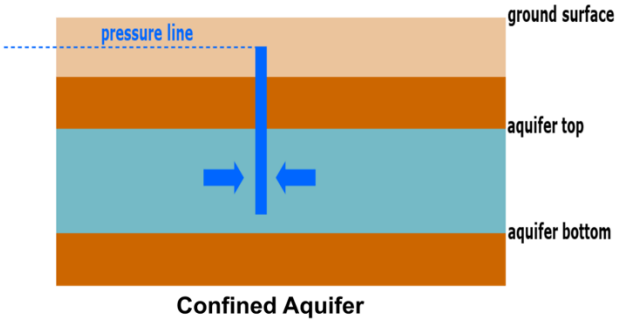
See slide: L03/08

An **aquifer** or a groundwater reservoir can store and transmit significant (= exploitable) amounts of groundwater.

An **aquitard** can store and transmit groundwater but to a much lesser extent than an (adjacent) aquifer.

An **aquiclude** can store groundwater but cannot transmit groundwater.

Solution 1b - (L03/11)



Confined Aquifer

Confined Aquifer

1. The essential feature of confined aquifer is provided in the figure above.

Q2. Groundwater storage (3 pts.)

Dry season in Dresden (2018-19) led to intense extraction of groundawater in rural areas. At one location in a confined aquifer (storage coeff. 4·10-4, total porosity = 30%), the pressure head was lowered by 150 m. The thickness of the aquifer was measured to be 90 m before the beginning of extraction and the compressibility of the porous medium in that region is estimated 6·10-8 m2/N. Density of water can be assumed to be 1000 kg/m3.

(Hint: $\Delta V_T = \alpha_{pm} \cdot \rho_w \cdot g \cdot V_T \cdot \Delta\psi$).

a. Approximately how much water was extracted? (1 point)

b. How much land subsidence due to water extraction is expected? (2 point)

Solution 2

Given relation:

For part a. (see Tut 02/P4)

$S_s = \frac{\Delta V_w}{V_T \cdot \Delta\psi}$

In confined aquifer S is used, which is obtained from:

$S = S_s \cdot m$

$\frac{S}{m} = \frac{\Delta V_w}{A_T \cdot m \cdot \Delta\psi}$

So,

$\frac{\Delta V_w}{A} = S \cdot \Delta\psi$

For part b.

$\Delta V_T = \alpha_{pm} \cdot \rho_w \cdot g \cdot V_T \cdot \Delta\psi$

$V_T = A \times h$, with A surface area and h aquifer thickness.

$\Delta V_T = A \times \Delta h$, with Δh change in thickness.

$A \times \Delta h = \alpha_{pm} \cdot \rho_w \cdot g \cdot A \cdot h \cdot \Delta\psi$

$\Delta h = \alpha_{pm} \cdot \rho_w \cdot g \cdot h \cdot \Delta\psi$, with Δh being the land subsidence

```
#  solution 2 a

# Given
```

(continues on next page)

(continued from previous page)

```
A = 1 # m^2, assuming 1 m^2 aquifer area
h = 90 # m, aquifer height before extraction
d_psi = 150 # m, change in pressure head
S_2 = 4*10**-4 #  specific storage
rho_w = 1000 # Kg/m^3, density of water
a_pm = 6* 10**-8 # m^2/N = m-s^2/kg, compressibility of porous medium
g = 9.81 # m/s^2, gravity factor

#Solution

d_V_w  = S_2*A*d_psi

print("The water abstraction volume per m\u00b2 aquifer is {0:0.3f}".format(d_V_w),
↪"m\u00b3")
```

```
The water abstraction volume per m² aquifer is 0.060 m³
```

```
#  solution 2 b


# Given

A = 1 # m^2, assuming 1 m^2 aquifer area
h = 90 # m, aquifer height before extraction
d_psi = 150 # m, change in pressure head
Ss = 4*10**-4 # specific storage
rho_w = 1000 # Kg/m^3, density of water
a_pm = 6* 10**-8 # m^2/N = m-s^2/kg, compressibility of porous medium
g = 9.81 # m/s^2, gravity factor

#interim calculation
V_T = A*h # m^3, Aquifer volume before extraction

#Solution

d_h  = a_pm*rho_w*g*h*d_psi

print("The water abstraction volume is {0:0.2f}".format(d_h), "m")
```

```
The water abstraction volume is 7.95 m
```

Q3. Aquifer Properties (ca. 10 pts.)

The hydraulic conductivity of a sample (length 15 cm, diameter 5 cm) is to be determined using a constant-head permeameter. For that 250 ml water is passed through the sample in 30 s while maintaining the head difference of 2.5 cm. Properties of water provided are: density of water at 20°C: 1000 kg/m3; dynamic viscosity of water at 20°C: 1.0087·10-3 Pa·s

a. Sketch the problem as accurately as possible providing essential features with legends (3 points)

b. What will be the conductivity of the sample? (4 points)

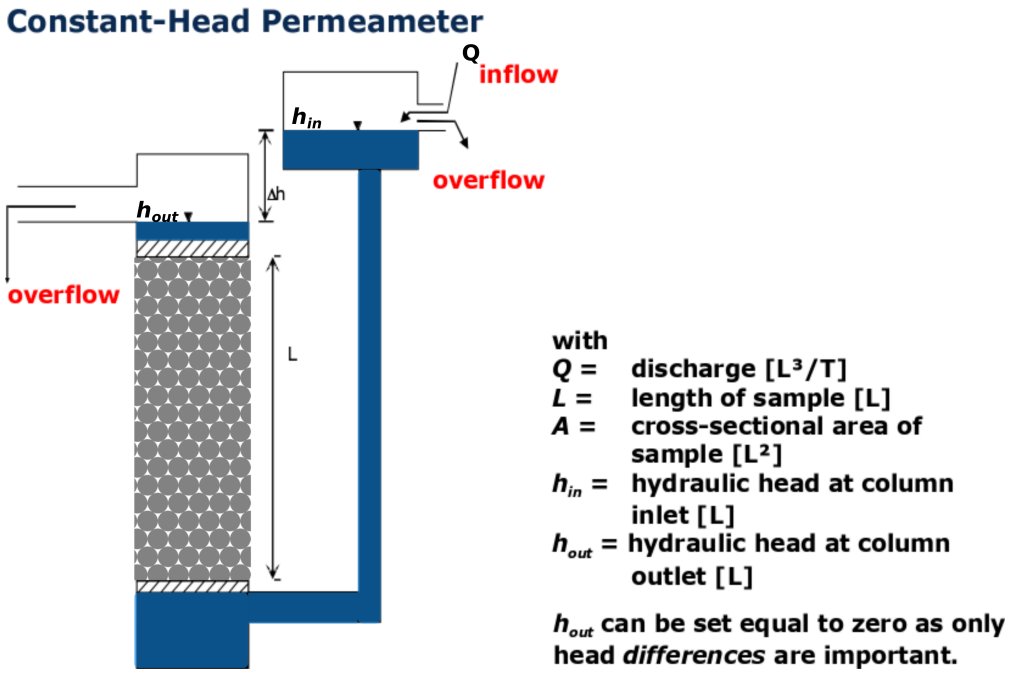
What is the intrinsic permeability of the sample? (2 points)

c. What soil type is likely the sample? (1 point)

(Hint: For the calculation of permeability, dynamic viscosity/density ratio is required.)

Solution 3 -

Solution 3a (L05/15)




```
#Solution 3b** (L05/15)

# Given

L_c = 15 # cm, column length
Dia_c = 5 # cm, diameter column
V_in= 250 # mL, water entering the column
t_c = 30 # s, time required to pass
d_3h = 2.5 # cm, head difference

# interim calculation
A_c = np.pi*Dia_c**2/4 # cm^2, Area of column
Q_c = V_in/t_c # cm^3/s, assume 1mL = 1 cm^3, Discharge out of column

#solution
K_c = (Q_c*L_c)/(A_c*d_3h) # cm/s, conductivity


print("The area of the column is {0:0.2f}".format(A_c), "cm\u00b2")
print("The discharge from the aquifer is {0:0.2f}".format(Q_c), "cm\u00b3/s")
print("The conductivity of the sample is {0:0.2f}".format(K_c), "cm/s")
print("The conductivity of the sample is {0:0.4f}".format(K_c/100), "m/s")
```

```
The area of the column is 19.63 cm²
The discharge from the aquifer is 8.33 cm³/s
The conductivity of the sample is 2.55 cm/s
The conductivity of the sample is 0.0255 m/s
```

```
#Solution 3c** (L05/18)

# Given

K_m = K_c/100 # m/s, conductivity
rho_3w = 1000 # Kg/m^3, density of water
nu_w = 1.0087*10**-3 # Pa-s = Kg/m-s, dynamic viscosity
g = 9.81 # m/s^2, gravity factor

#solution
k_c = K_m*nu_w/(rho_3w*g)

print("The permeability of the sample is {0:0.10f}".format(k_c), "m\u00b2")
print("The permeability of the sample is {0:0.2E}".format(k_c), "m\u00b2")
```

```
The permeability of the sample is 0.0000000026 m²
The permeability of the sample is 2.62E-09 m²
```

Solution 3d (L05/11)

The sample in the column is likely gravel or coarse sand.

Q4. Sieve Analysis (ca. 6 pts.)

Sieve experiments were performed with the bore samples and the following observations were obtained:

mesh diameter [mm]	residue in the sieve [g]	total	/total
6	0		
2	40		
0.6	250		
0.2	150		
0.06	60		
<0.06 (cup)	10		

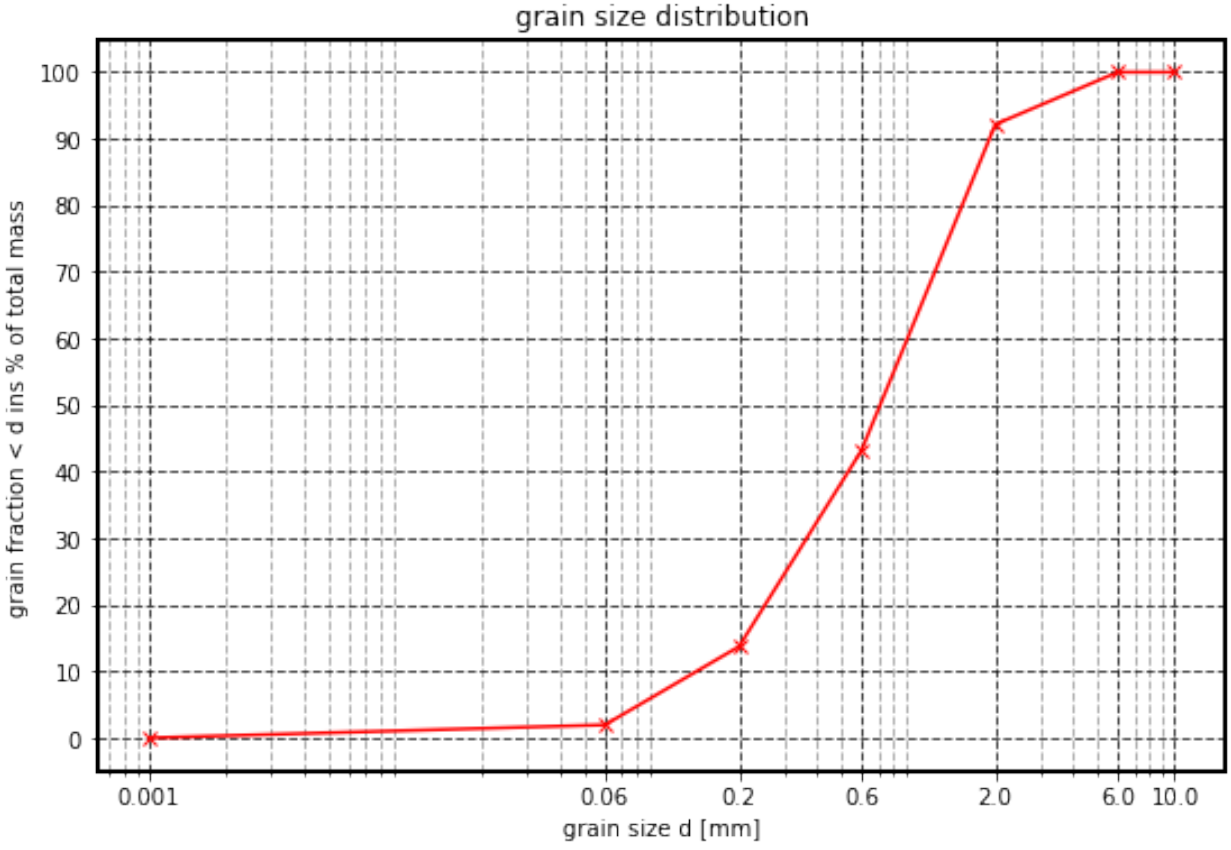
a. Draw the granulometric curve in the diagram below. (ca. 5 pts.)

b. Briefly characterise the sediment. (ca. 1 pt.)

```
#Solution 4 -(L03/18)

dia = [6,2,0.6,0.2, 0.06, 0.001] # mm, diameter <0.06 (cup)= 0.001
mass = [0, 40, 250, 150, 60, 10] # g, the residue in seive

# Calculation steps - filling table
Total_mass = np.sum(mass)  # add the mass column to get total mass
retain_per = mass/Total_mass*100   # retain percentage
retain_per_cumsum = np.cumsum(retain_per) # get the cummulative sum of the reatined
passing_per = 100 - retain_per_cumsum # substract 100-cummsum to get passing %

data = {"mesh diameter [mm]": dia, "residue in the sieve [g]": mass, "total": retain_
↪per, "/total": passing_per }

df1= pd.DataFrame(data)
df1
```

	mesh diameter [mm]	residue in the sieve [g]	total	/total
0	6.000	0	0.000000	100.000000
1	2.000	40	7.843137	92.156863
2	0.600	250	49.019608	43.137255
3	0.200	150	29.411765	13.725490
4	0.060	60	11.764706	1.960784
5	0.001	10	1.960784	0.000000

```
# plotting
plt.rcParams['axes.linewidth']=2
#plt.rcParams["axes.edgecolor"]='white'
plt.rcParams['grid.linestyle']='--'
plt.rcParams['grid.linewidth']=1
x = np.append([10],dia) # adding data to extend over 6 mm dia
y = np.append([100],passing_per) # adding 100% to plot

fig = plt.figure(figsize=(9,6))
plt.semilogx(x, y, 'x-', color='red')
tics=x.tolist()

plt.grid(which='major', color='k', alpha=0.7)
plt.grid(which='minor', color='k', alpha=0.3)
plt.xticks(x, tics);
plt.yticks(np.arange(0,110,10));
plt.title('grain size distribution');
plt.xlabel('grain size d [mm]');
plt.ylabel('grain fraction < d ins % of total mass');
```

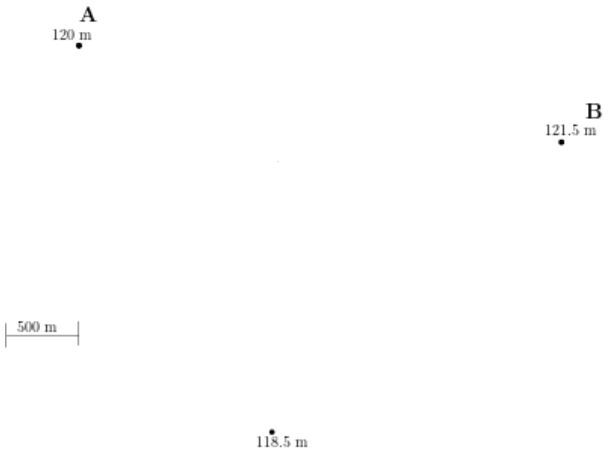


solution 4b

The sample can be considered uniformly distributed as over 70% of sample falls in the sand size (0.2 mm-2 mm). Therefore, the sample can be considered sandy.

Q5. Aquifer characterization (ca. 8 pts.)

Water levels in m a.s.l. were measured at three observation wells (see figure).



a. Sketch hydraulic head isolines for increments of 0.5 m. (ca. 3 points.)

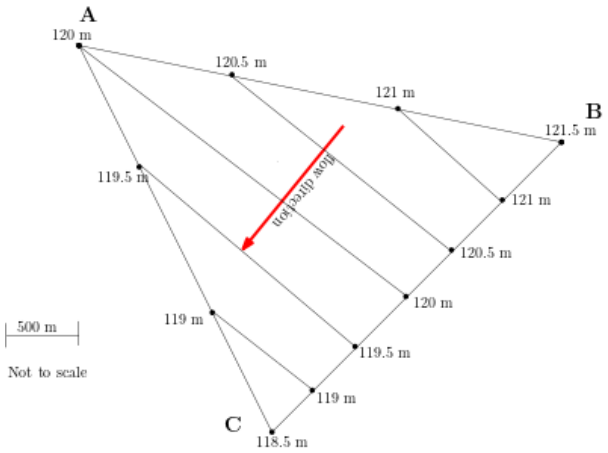
b. Gravel layer (thickness (t1) = 1.5 m, and conductivity (K1) = 3.7 10-3 is embedded between two sandy layers (t2 = 2 m, K2 = 3·10-4 m/s; and t3 = 3 m, K3 = 4·10-4 m/s). If the hydraulic gradient is 1% and overall discharge is 1 m3/d per unit width of the aquifer, find the effective hydraulic conductivity considering a parallelly layered aquifer.

(Hint: $K_{eff} = \frac{m}{\sum_{i=1}^{n} \frac{m_i}{K_i}}$ or $K_{eff} = \sum_{i=1}^{n} \frac{m_i \cdot K_i}{m}$) (ca. 2 points)

c. Distinguish between homogeneity and heterogeneity, and isotropy and anisotropy (ca. 3 points)

Solution 5a (L07/08-09)

The isolines and flow direction is provided in the figure below.




```
# Solution 5b (L06/08-13)

# Given:

G_t1 = 2 # m, sandy layer top
G_t2 = 1.5 # m, gravel layer middle
```

(continues on next page)

(continued from previous page)

```
G_t3 = 3 # m, sandy layer bottom
K_1 =  3.0*10**-4 # m/s cond. in G_t1
K_2 = 3.7*10**-3 # m/s cond. in G_t2
K_3 =  4.0*10**-4 # m/s cond. in G_t3
i = 1/100 # (), hydraulic gradient 1%
Q_5 = 1 # m^3/d per-W, discharge per unit width

#intermediate calculation
m = G_t1+G_t2+G_t3 # m, total aq. thickness

K_ef_h = (1/m) * (G_t1*K_1 + G_t2*K_2 + G_t3*K_3) # m/s, eff. horizontal cond.
K_ef_v = m/(G_t1/K_1 + G_t2/K_2 + G_t3/K_3) # # m/s, eff. vertical cond.

print("The thickness of the aquifer is {0:0.3f}".format(m), "m")
print("The effective horizontal conductivity of the aquifer is {0:0.2E}".format(K_ef_
↪h), "m/s")
print("The effective vertical conductivity of the aquifer is {0:0.2E}".format(K_ef_v),
↪ "m/s")
```

```
The thickness of the aquifer is 6.500 m
The effective horizontal conductivity of the aquifer is 1.13E-03 m/s
The effective vertical conductivity of the aquifer is 4.46E-04 m/s
```

Solution 5c (L06/23)

Homogeneity: An aquifer is homogeneous when its parameters are constant throughout the porous medium, i.e. the properties of the medium are independent of space

Heterogeneity: Heterogeneous aquifer have its properties varies in space or the properties are space dependent.

Isotropy: This relates to properties of aquifer being independent of direction, i.e., $K_v = K_h$

Anisotropy: In this case the aquifer properties are direction dependent, i.e., $K_v \neq K_h$.

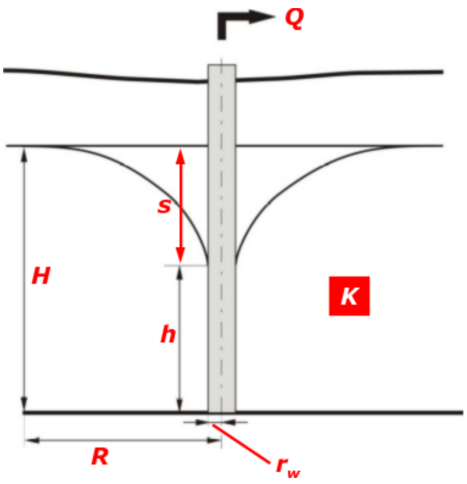
Q6. Well (ca. 5 pts.)

a. Sketch the pumping scenario of an unconfined aquifer (vertical cross section) and label all possible quantities (ca 3 pts.)

b. The conductivity of a confined aquifer (8 m thick) is estimated to be $4 \cdot 10^{-4}$ m/s. If the steady-state discharge 50 m3/s, using the Theis equation ($s = Q/4\pi TW(u)$, with $W(u) = 15$), find the drawdown in the aquifer. (2 points)

Solution 6a (L08/16)

Figure below presents the scenario of a well in an *unconfined* aquifer.



Relevant quantities:

- **pumping rate *Q* [L³/T]**
- **hydraulic conductivity *K* [L/T]**
- **water level at rest *H* [L]**
- **water level in the pumping well *h* [L]**
- **radius of influence *R* [L]**
- **well radius r_w (incl. gravel pack!) [L]**
- **drawdown $s = H - h$ [L]**

```
#Solution 6b

#Given

Q_6 = 50 # m^3/s, discharge
K_6 = 4*10**-4 # m/s, conductivity
m_6 = 8 # m, thickness
W_u = 15 # (), well function

# interim cal.
T_6 = K_6 * m_6 # m^2/s, Transmissivity T = K*m

# solution
s_6 = (Q_6/(4*np.pi*T_6)) * W_u # m, drawdown

print("The Transmissivity of the aquifer is {0:0.5f}".format(T_6), "m\u00b2/s")
print("The drawdown in the well is {0:0.2f}".format(s_6), "m")
```

```
The Transmissivity of the aquifer is 0.00320 m²/s
The drawdown in the well is 18650.97 m
```

Q7. Conservative Transport (ca. 7 pts.)

a. How is reactive transport different to conservative transport in the aquifers. (2 points)

b. With suitable sketch distinguish between advective flux and dispersive flux. (2 points)

c. A column (L = 1.2 m and Ø = 5 cm) was packed with sandy soil (ne= 35% K= 0,0002 m/s). The hydraulic head at the inlet and the outlet was set to 230 m and 235 m, resp. The NaCl solution with conc. 10 mg/L was steadily introduced to the column after saturating it with distilled water. The experiment condition was such that diffusive flow could be neglected. You may make justified assumption for any missing information.

c.i. What will be the advective mass flux at the outlet of the column? (1.5 points)

c.ii. Considering initial concentration difference between inlet and outlet to be 10 mg/L, what will be the dispersive mass flux at the outlet? (1.5 points)

(Hint: Dispersive and Advective fluxes are either of $n_e \cdot v \cdot C$ and $n_e \cdot \alpha \cdot v \cdot \Delta C/L$)

Solution 7a (L09/05)

A chemical in groundwater is subject to conservative transport processes if there is:

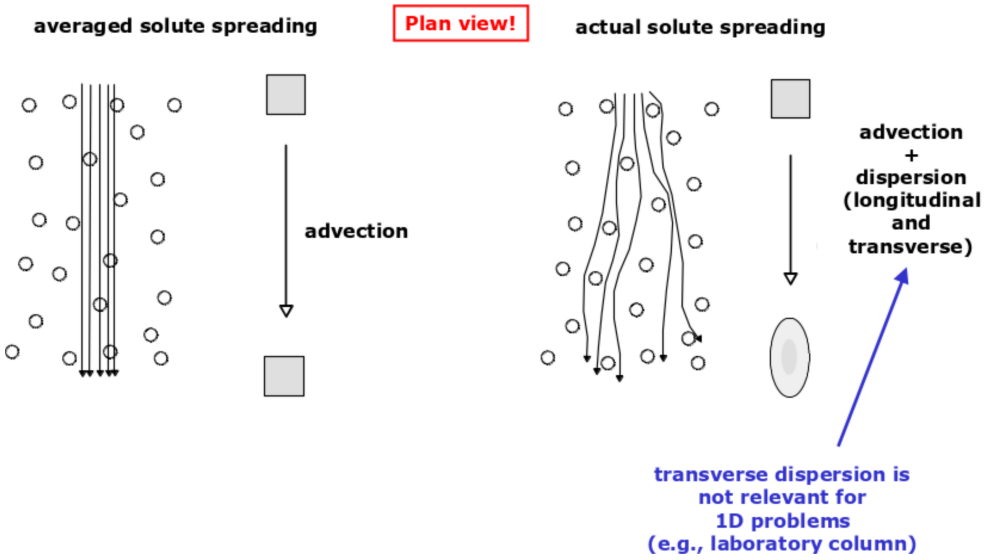
- no interaction with the solid material,

- no interaction with other chemicals,
- no interaction with microbes.

When either of the above are part of the groundwater, the transport process is reactive.

Solution 7b (L09/09)

The sketch below distinguish between advective and dispersive fluxes. The figure in the left is of advective process and that in the right results to dispersive flux.




```
#Solution 7c

L_7 = 1.2 # m, col. length
Dia_7 = 5 # cm, col. diameter
ne_7 = 0.35 # (), effective porosity
K_7 = 0.0002 # m/s, conductivity
H_7in = 235 # m, head inlet
H_7out = 230 # m, head outlet
C_7 = 10 # mg/L, NaCl concentration
al_7 = 1 # m, assumed
C_7d = 10 # mg/L

#intermediate calc.
i_7 = (H_7in-H_7out)/L_7 # (), head gradient
v_7dar = K_7*i_7 # m/s, darcy velocity
v_7av = v_7dar/ne_7 # m/s, average linear velocity


# Solution
F_7ad = ne_7*v_7av*C_7 # mg-m/L-s, advective flux
F_7dis = ne_7*al_7*v_7av*C_7d/L_7 # mg-m/L-s, dispersive flux

print("The hydraulic gradient is {0:0.4f}".format(i_7), "")
print("The Darcy velocity is {0:0.4f}".format(v_7dar), "m/s")
print("The average linear velocity is {0:0.4f}".format(v_7av), "m/s")
print("The advective flux is {0:0.10f}".format(F_7ad), "mg-m/L-s")
print("The dispersive flux is {0:0.10f}".format(F_7dis), "mg-m/L-s")
```

```
The hydraulic gradient is 4.1667
The Darcy velocity is 0.0008 m/s
The average linear velocity is 0.0024 m/s
The advective flux is 0.0083333333 mg-m/L-s
The dispersive flux is 0.0069444444 mg-m/L-s
```

Q8. Sorption Isotherms (ca. 10 pts)

Five batch tests (different initial concentrations – see table below) were performed to determine the sorption properties of a sediment. For each batch 20 g of sediment in 30 mL of water were used. The measured equilibrium solute concentrations are also provided in the table.

a) Complete the above table (ca. 3 pts.)

b) Plot the results in the diagram below and draw a Henry isotherm (ca. 3 pts.)

c) How is retardation related to isotherm (ca. 2 points) (value and unit!) (ca. 2 pts.).

```
head = ["Batch nr. ", "Initial Conc. (mg/L) ", "Equi. Conc. (mg/L)", "Sorbed mass (g)
↪", "Sorbed mass/solid (mg/g)" ]
bn = np.array([1,2,3,4,5])
C_0 = np.array([5, 10, 15, 20, 25])# mg/L, initial conc.
C_eq = np.array([2.5, 4.9, 8, 9.8, 13.2])# mg/L, equilibrium conc.
s2 = ips.sheet(rows=6, columns=5, row_headers=False, column_headers=head)
ips.column(0, bn, row_start=0)
ips.column(1, C_0, row_start=0)
ips.column(2, C_eq, row_start=0);
s2
```

```
Sheet(cells=(Cell(column_end=0, column_start=0, row_end=4, row_start=0, squeeze_
↪row=False, type='numeric', val...
```

```
# SOlution of Problem 10 a (T07/HP9)

# Given
v_ml = 30 # ml of water used in expt.
v_l = v_ml/1000 # L, unit conversion
m_s = 20 # g, solid mass used in expt.

bn = np.array([1,2,3,4,5])
C_0 = np.array([5, 10, 15, 20, 25])# mg/L, initial conc.
C_eq = np.array([2.5, 4.9, 8, 9.8, 13.2])# mg/L, equilibrium conc.
s_m = (C_0-C_eq)*v_l
m_m = s_m/m_s# mg/g, mass ratio

#output
d8 = {"Batch Nr": bn, "Initial Conc. (mg/L)": C_0, "Equi. Conc. (mg/L)": C_eq,
↪"Sorbed mass (g)": s_m, "Sorbed mass/solid (mg/g)" :m_m}
df9 = pd.DataFrame(d8); df9
```

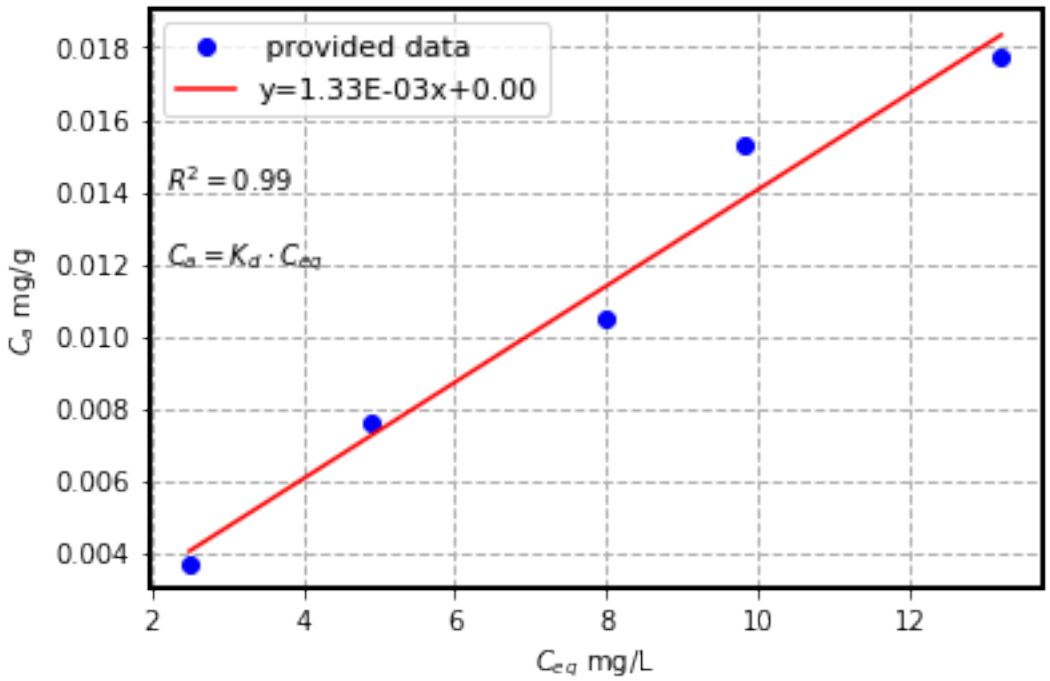
	Batch Nr	Initial Conc. (mg/L)	Equi. Conc. (mg/L)	Sorbed mass (g)	\
0	1	5	2.5	0.075	
1	2	10	4.9	0.153	
2	3	15	8.0	0.210	
3	4	20	9.8	0.306	
4	5	25	13.2	0.354	

(continues on next page)

(continued from previous page)

```
   Sorbed mass/solid (mg/g)
0                   0.00375
1                   0.00765
2                   0.01050
3                   0.01530
4                   0.01770
```

```
# Solution of proble 10 (b) (T07/HP9)
# fit
slope, intercept, r_value, p_value, std_err = stats.linregress(C_eq, m_m) # linear␣
↪regression

#plot and fit
fig = plt.figure(); plt.plot(C_eq, m_m, 'bo', label=' provided data');
pred = intercept + slope*C_eq # fit line
plt.plot(C_eq, pred, 'r', label='y={:.2E}x+{:.2f}'.format(slope,intercept)) ;
plt.xlabel(r"$C_{eq} $ mg/L"); plt.ylabel(r"$C_{a} $ mg/g");
plt.grid(); plt.legend(fontsize=11);  plt.text(2.2, 0.014,'$R^2 = %0.2f$' % r_value)
plt.text(2.2, 0.012,'$C_a = K_{d}\cdot C_{eq}$');
```



solution 8c (L10/13)

The following relation relates Retardation (R) with linear isotherm (K_d)

$$R = 1 + \frac{1 - n_e}{n_e} \rho_s K_d$$

with effective porosity n_e, solid density ρ_s.

Q9. Groundwater Modelling (ca. 8 points.)

a. Distinguish between conceptual model and mathematical model; and between analytical solution and empirical solution (ca. 4 points).

b. Draw a conceptual model for a rectangular aquifer 100 m long and 20 m wide. Discretize the domain with 1/10 of the length length-wise and 1/5 of the width width-wise. Assure that flow in the model is from left to right direction (ca. 3 points).

c. How is a no-flow boundary condition mathematically defined? (ca. 1 point)

Solution 9a (L11/04-06)

A model or also a *conceptual model* is a representation, an image or a description of a real system.

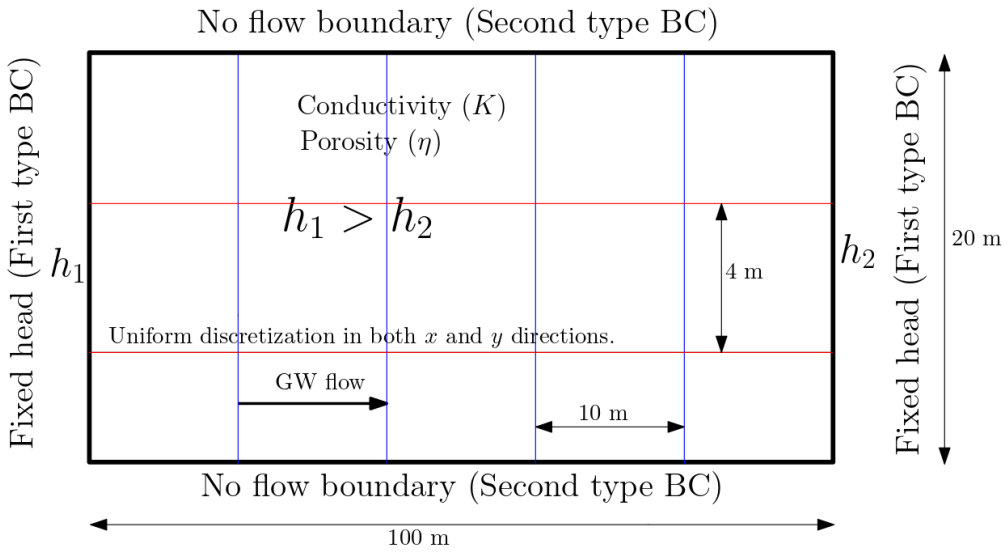
example for a real system: porous medium with water flowing through the pores (Darcy experiment)

A *mathematical model* provides a quantitative representation of the relevant system components, processes and impacts in the area of investigation. The quantitative representation is based on mathematical equations.

Analytical solution : These are exact mathematical expressions solving the model equations.

Emperical solution : These are solution based on experimental results.

Solution 9b - (L14/12)



Solution 9c (L13/16)

A no-flow boundary condition is special case of second type or Neumann boundary condition. For no flow condition head gradient is equated to zero, i.e., there is no gradient and thus no flow (water flows from high to low head). Mathematically, this is:

$\frac{dh}{dx} = 0$ for no-flow along $x-$axis, with h representing head.

Good Luck.