Delft University of Technology

Modelling Coupled Processes for Engineering Applications

CIE4365-16

Assignment 2: Heat and unsaturated water flow in soils

Authors
Group CP2022 14:
Akhilesh Soodan (5594758)
Gheylla Liberia (4583825)
Mazen Alqadi (4568478)
Priyadarshini Thiruvenkatachari (5522722)

June 2, 2022

Abstract

Water flow in unsaturated soil is non linear because of the dependence of soil water pressure and hydraulic pressure on water content in the soil. So it is important to model this flow to get numerical solutions to predict the real nature of unsaturated zone under various conditions. This assignment aims to model and compare a zero pressure head gradient against a Robbins Boundary condition with an external head of -1 m and a resistance term of 0.005 1/s. The top boundary condition is a situation a zero flux condition for a period of 25 days and after that a constant flux of -0.001 m/day for 200 days.

Contents

1 Introduction

The purpose of this assignment is to model different water flows in soil and specifically compare a gravity flow with a Richards flow. In order to simulate and model the water flow, the Richard's partial differential equation for unsaturated flow is used. 'The partial differential flow equation can be interpreted numerically by a finite difference, a finite element or a boundary element technique. Then a discretization scheme is applied for a system of nodal points that is superimposed on the soil depth-time region under consideration'.

Furthermore initial and boundary conditions for the soil need to be set in order for the equations to be solved. The results of this simulation include different variables such as pressure, soil moisture content, water flux and diverging water flux, all as a function of soil depth and time in days.

The methods and results are presented in the following sections.

2 Method

The equations required for the modelling/simulation of the unsaturated water flow are described in the following section.

2.1 Model Equations

To model a coupled process, water flow model is created using the following equations. Firstly, the pressure head from Richardson's equation is as follows:

$$[C(h_w) + S_w * S_{sw}]\frac{\partial h_w}{\partial t} - \nabla \cdot [K_{sat} * k_{rw}(\nabla h_w + \nabla z)] = 0 \tag{1}$$

In order to couple the water content to the capillary pressure head the "van Genuchten" equation (eq. 2) is used:

$$S_{effective}(h_c) = \begin{cases} [1 + (\alpha * h_c)^n]^{-m}, & \text{if } h_c > 0 \\ 1, & \text{if} h_c \leq 0 \end{cases} \tag{2}$$

In this equation the effective storage can be calculated using equation 3.

$$S_{effective}(h_c) = \frac{\theta_w - \theta_{res}}{\theta_{sat} - \theta_{res}} \tag{3}$$

$$h_c = h_a - h_w \tag{4}$$

In the equations presented above θ_w is the volumetric water content, h_w is the pressure head, θ_{res} is the residual volumetric water content, θ_{sat} is the saturated volumetric water content, h_c is the capillary pressure head, where h_a is taken as 0 resulting in h_c = -h_w and lastly α, m and n are empirical parameters.
Furthermore a permeability term has to be used, the simplified equation is presented below:

$$k_{rw} = (S_{effective})^3 \tag{5}$$

In order to derive the equation for the differential water capacity $C(h_w) = \frac{\partial \theta_w}{\partial h_w}$ the imaginary approximation as presented in the paper is used. And lastly, to solve the model and simulate the unsaturated water flows numerical solutions from python need to be used. A Jacobian matrix is constructed and the difference between the solution with and without the Jacobian is determined.
This is combined with the heat flow equations which are as follows:

2.2 Discretization

The estimation of the water flow in a soil column is done through its discretization into small elements and it is then calculated iteratively in the code.The space is being discretized by defining the soil column through a staggered grid. The soil column is divided into 100 layers, on which 100 nodes are defined in the middle of the layer and 101 inter-nodes are defined on the boundary between the layers. Figure 1 is a schematic overview of the discretization. Inter-nodes (zIN) are positioned from -1 to 0 in hundred equal steps. The first node (zN) is set equal to the first internode while the last node is set equal to the last inter-node. The boundary nodes are determined based on the boundary conditions. Inter-nodes are positioned between

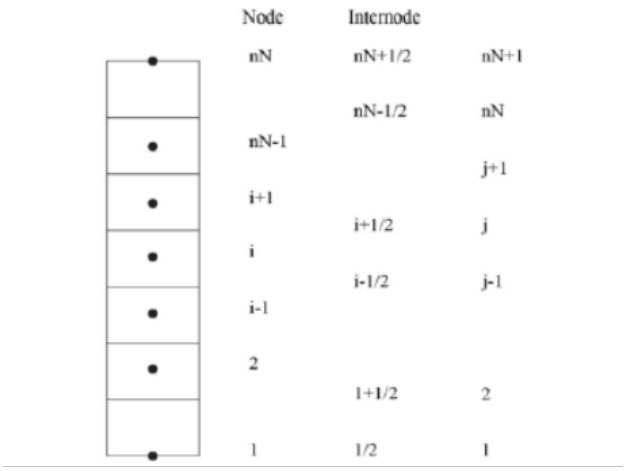

Figure 1: Discretization of soil column

the nodes, the position of the inter-nodes is based on the average position of the nodes above and below the inter-node.
The positions of the nodes and inter-nodes are used to calculate values of various physical parameters in either nodes or inter-nodes throughout the whole model. Equations used in the code are given by

$$q_{w_{i+1/2}} = -K_{sat_{i+1/2}} k_{rw_{i+1/2}} \left(\frac{hw_{i+1} - hw_i}{\Delta z_i} + 1\right) \tag{6}$$

$$q_{w_{i-1/2}} = -K_{sat_{i-1/2}} k_{rw_{i-1/2}} \left(\frac{hw_i - hw_{i-1}}{\Delta z_{i-1}} + 1\right) \tag{7}$$

2.3 Parameter Values

To be able to compute the model some parameters as well as the boundary and initial conditions have to be set. these values are presented in table 1. An explanation of the chosen values/ranges is given below.

Parameter	**Value**
Density water	1000 kg/m3
Density soil (clay)	2650 kg/m3
Compressibility Clay	0.00000001
Saturated Water content	0.4
Residual saturated water content	0.03
Saturated Hydraulic Conductivity	0.05
n	3
Power factor van Genuchten	0.5
Alpha	2

Table 1: Parameter values

- Density of water a known value that does not need to be calculated.

- Assuming that the simulation is run for Dutch soils the density of the soil can be taken as the density of clay knowing that the majority of the soil in the Netherlands is comprised of clay.
- The compressibility of clay is very low, nearly zero. This is due to the fact that clay is already very densely packed, as a result the value of compressibility is 0.00000001.
- The saturated water content is equal to the pore space of the soil. In this case the porosity of clay is approximately 0.4.
- Lastly, the saturated hydraulic conductivity for clay is approximately 0.048.

2.4 Pseudo-code

To make the as uncluttered as possible two different codes are constructed. First of all a class code containing all of the equations is made. Within this class the following equations are defined:

- Effective storage
- Volumetric water content
- Complex derivative
- The Jacobian
- Water flux
- The Diverging water flux

Firstly a separate class code is written defining functions 2, 3, 4 and 5. The functions can be written directly in the script code, however to keep the code looking clear it is useful to make a separate code containing all of the functions and later import this class code into the official script. The complex derivative, water flux and diverging water fluxes are also defined in the class code. These functions make use of matrix (array) notations in order to compute them for the entirety of the soil layer.

```
def SeffFun(self, hw):#Se = Effective Saturation = Seff
def ThetaFun (self, hw):
def CComplxFun(self, hw):
def CPrimeFun (self, hw):
def WaterFlux(self, t, hw, nOut):
def DivWaterFlux(self, t, hw, nOut):
```

Figure 2: Defined functions in class file

Upon completion of the class code, the script is defined. Firstly the layer is divided into different nodes and internodes, these are the nodes in the middle of each layer.To be exact the soil is divided into 100 layers and 101 nodes and 100 internodes . Each layer has a height of dz, which is defined as the location of the node before minus the location of the node after.

```
#the amount of internodes
nIN = 101

# soil profile until 15 meters depth internodes
zIN = np.linspace(-1, 0, num=nIN).reshape(nIN, 1)


#defining the internodes
zN = np.zeros(nIN - 1).reshape(nIN - 1, 1)
zN[0, 0] = zIN[0, 0]
zN[1:nIN - 2, 0] = (zIN[1:nIN - 2, 0] + zIN[2:nIN - 1, 0]) / 2
zN[nIN - 2, 0] = zIN[nIN - 1]
nN = np.shape(zN)[0]

ii = np.arange(0, nN - 1)
dzN = (zN[ii + 1, 0] - zN[ii, 0]).reshape(nN - 1, 1)
dzIN = (zIN[1:, 0] - zIN[0:-1, 0]).reshape(nIN - 1, 1)
```

Figure 3: Nodes definition

All of the layer dimensions are saved into a tuple that is made into a pandas series. This makes it more efficient to work with the different dimension parameters.
Just like the dimensions, the soil characteristics are also defined into a tuple, which is then later converted into a pandas series. The values values for the parameters are explained in section 2.3.
After all of the required parameters and conditions have been set the class can be imported and solved using the integrated solver.
The soil is divided into 150 nodes and so the matrix dimensions for the soil can be defined and the height between the layers can also be defined.

Upon defining the equations and discretization of the soil into layers, the boundary and the initial conditions are set and the simulation can be initialized by using the water table location.

```
# Initial Conditions
zRef = -0.75
hwIni = zRef - zN
MyWF = WF.WaterDiffusion(sPar, mDim, bPar)
```

Figure 4: Initialization

Lastly, the time is set and the simulation can be run.

3 Specific functions

3.1 Richard's Equation for water flow

The Richard's equation represents the movement of water in unsaturated soils. The mass balance can be numerically written as:

$$\frac{\partial(\rho\varepsilon S_w)}{\partial t} + \nabla.(\rho_w q_w) + (\rho_w q_w) = \rho_w Q^w \tag{8}$$

where,
ε = porosity[-]
ρ_w = density of water [kg/ m^3]
S_w = saturation of water [-]
Q^w = source or sink
Flux is written as

$$q_w = \frac{-kk_{rw}}{\mu_w}(\nabla p_w - \rho_w g) \tag{9}$$

where, k = saturated permeability [m/day]
k_{rw} = relative permeability [m/day]
g = gravitational acceleration [m/s^2]
p_w = water pressure [KPa]
Combining the Darcy's law with the mass balance leading to

$$\frac{\partial}{\partial t}(\rho_w \varepsilon S_w) - \nabla \cdot \left(\frac{-kk_{rw}}{\mu_w}(\nabla p_w - \rho_w g)\right) = \rho_w Q^w$$

can be simplified to

$$\varepsilon\frac{\partial S_w}{\partial t} - \nabla \cdot \left(\frac{kk_{rw}}{\mu_w}(\nabla p_w - \rho_w g) = 0\right.$$

by assuming a constant porosity, water density and no source or sink
We get the mixed form by using water content θ, K_sat and the hydraulic head

$$h_w = \frac{p_w}{\rho_w g}$$

$$\frac{\partial}{\partial t}(\rho_w \varepsilon S_w) - \nabla \cdot \left(\frac{-kk_{rw}}{\mu_w}(\nabla p_w - \rho_w g)\right) = \rho_w Q^w \tag{10}$$

3.2 Jacobian Implementation

The Jacobian can be calculated by using the following approach:

$$C'_{wi}\frac{\partial h_{wi}}{\partial t} = a_i h_{i-1} + b_i h_i + c_i h_{i+1} + d_i$$

a_i,b_i,c_i and d_i are the coefficients which construct the diagonals of the matrix. These coefficients are multiplied by the known values of the water head, $h_i inthecells$.
The coefficients are:

$$
\begin{aligned}
a_i &= \frac{k_{i-0.5}}{\Delta z_{i-0.5}\Delta z_{i-1}} \\
b_i &= -\left(\frac{k_{i-0.5}}{\Delta z_{i+0.5}\Delta z_{i-1}} + \frac{k_{i+0.5}}{\Delta z_{i-0.5}\Delta z_i}\right) \\
d_i &= \left(\frac{k_{i-0.5}}{\Delta z_{i-0.5}} - \frac{k_{i+0.5}}{\Delta z_{i-0.5}}\right) \\
c_i &= \frac{k_{i+0.5}}{\Delta z_{i-0.5}\Delta z_i}
\end{aligned}
\tag{11}
$$

k is the permeability which consists of the relative permeability and the saturated permeability.

$$
k = K_{sat} k_{rw} \tag{12}
$$

$$
\begin{bmatrix}
C'_{w_1}\frac{\partial h_{w_1}}{\partial t} \\
C'_{w_2}\frac{\partial h_{w_2}}{\partial t} \\
C'_{w_i}\frac{\partial h_{w_i}}{\partial t} \\
C'_{w_{nn-1}}\frac{\partial h_{w_{nn-1}}}{\partial t} \\
C'_{w_{nn}}\frac{\partial h_{w_{nn}}}{\partial t}
\end{bmatrix}
=
\begin{bmatrix}
b_1 & c_1 & & & & & & \\
a_2 & b_2 & c_2 & & & & & \\
 & & a_i & b_i & c_i & & & \\
 & & & & a_{nn-1} & b_{nn-1} & c_{nn-1} & \\
 & & & & & a_{nn} & b_{nn}
\end{bmatrix}
\begin{bmatrix}
h_1 \\ h_2 \\ h_i \\ h_{nn-1} \\ h_{nn}
\end{bmatrix}
+
\begin{bmatrix}
d_1 \\ d_2 \\ d_i \\ d_{nn-1} \\ d_{nn}
\end{bmatrix}
$$

Figure 5: Approximation of Jacobian matrix

4 Results Discussion

4.1 Results

The model was run for both the 'gravity' condition as well as the 'robins' condition. The gravity condition requires a zero pressure head gradient while for the 'robins' condition an external head of -1 meters is required and a resistance term of 0.005 1/s is used.

The results of both water simulation flows can be seen in the following section. (Note that in all of the figures, the left figure is attributed to the gravity condition and the right figure to the robins condition.

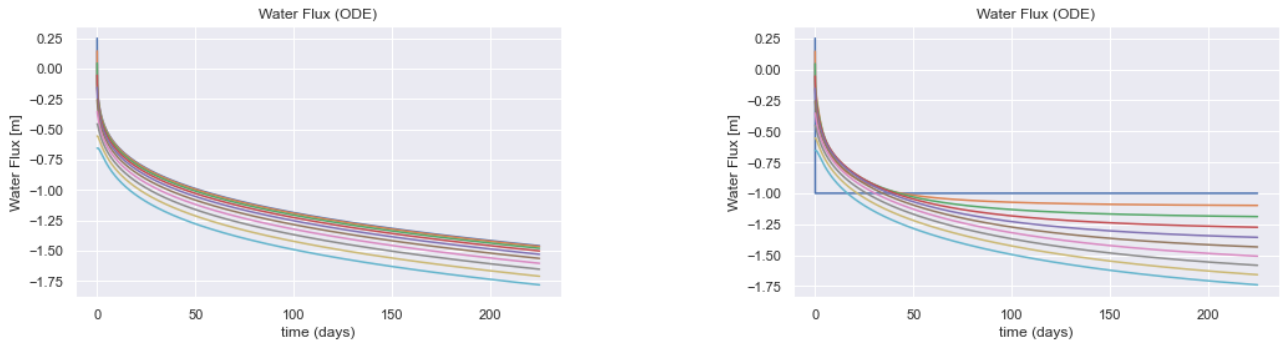

Figure 6: Water flux [Left = gravity, Right = robins]

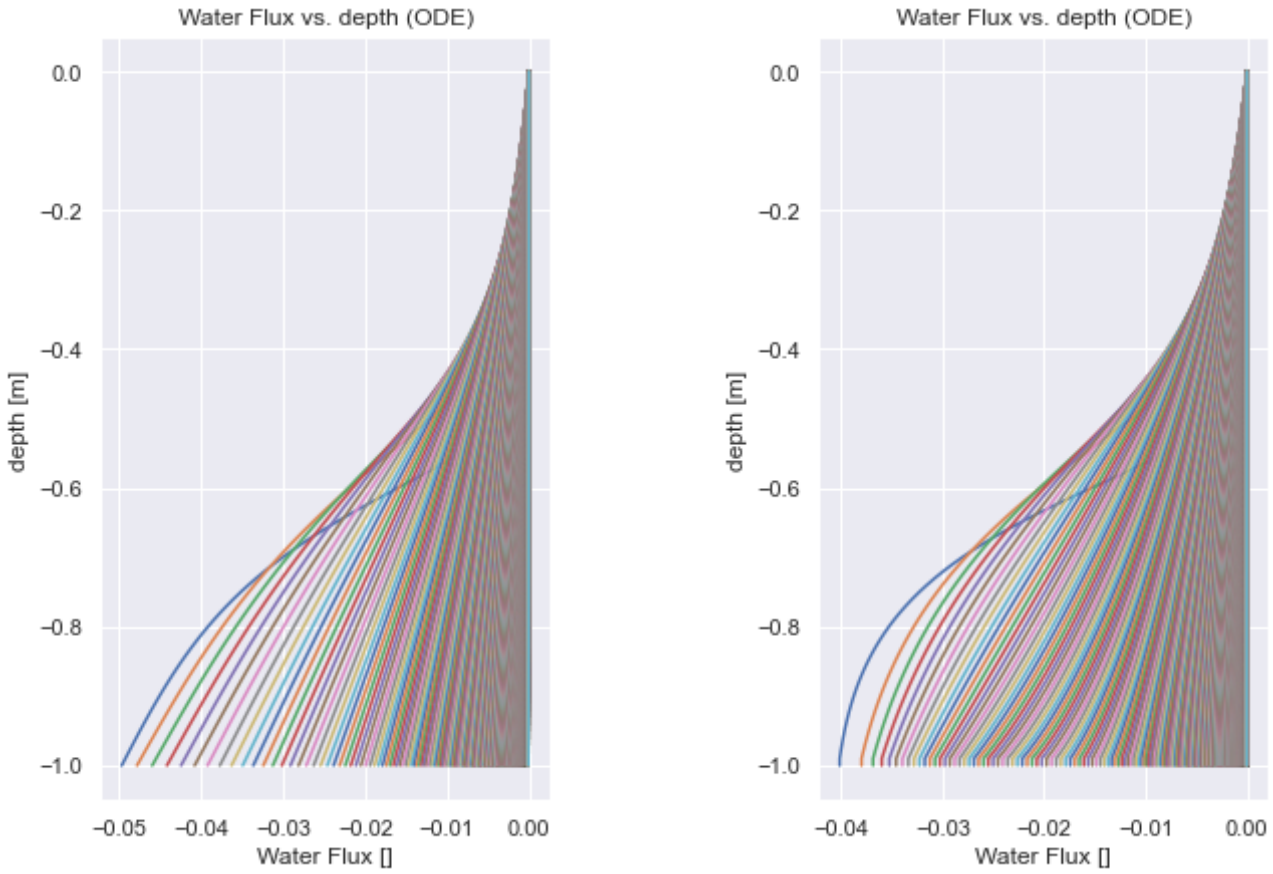

Figure 7: Water flux vs Depth [Left = gravity, Right = robins]

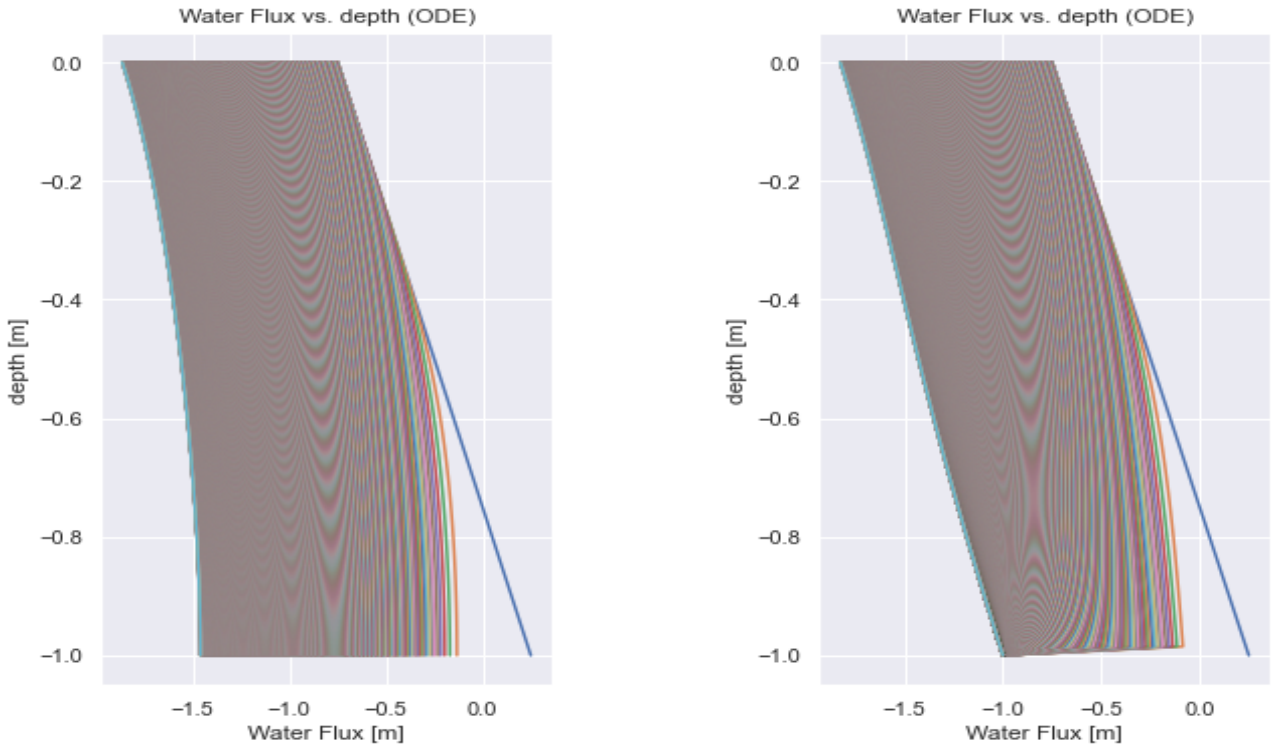

Figure 8: Water flux [Left = gravity, Right = robins]

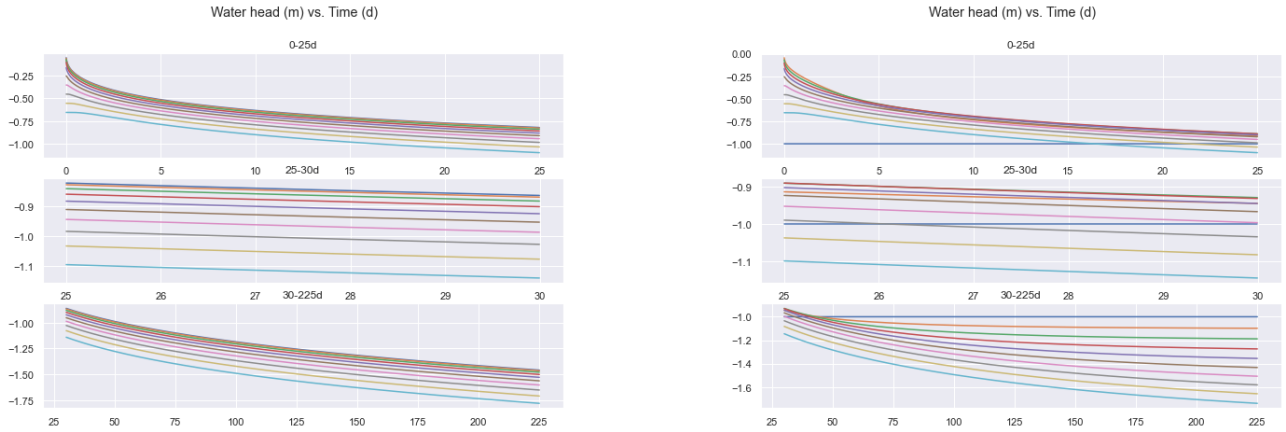

Figure 9: Water content vs Depth

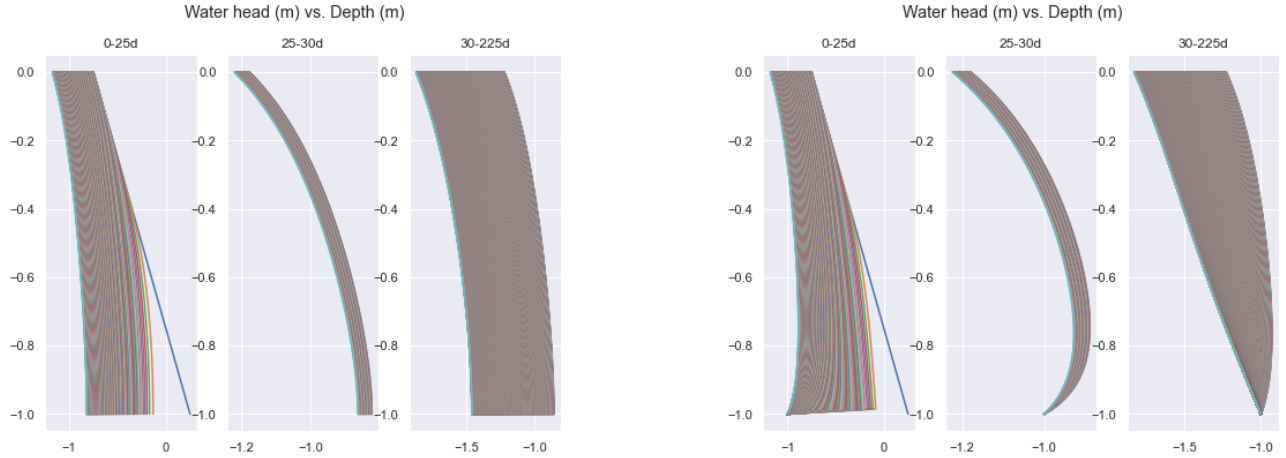

Figure 10: Water head vs Time [Left = gravity, Right = robins]

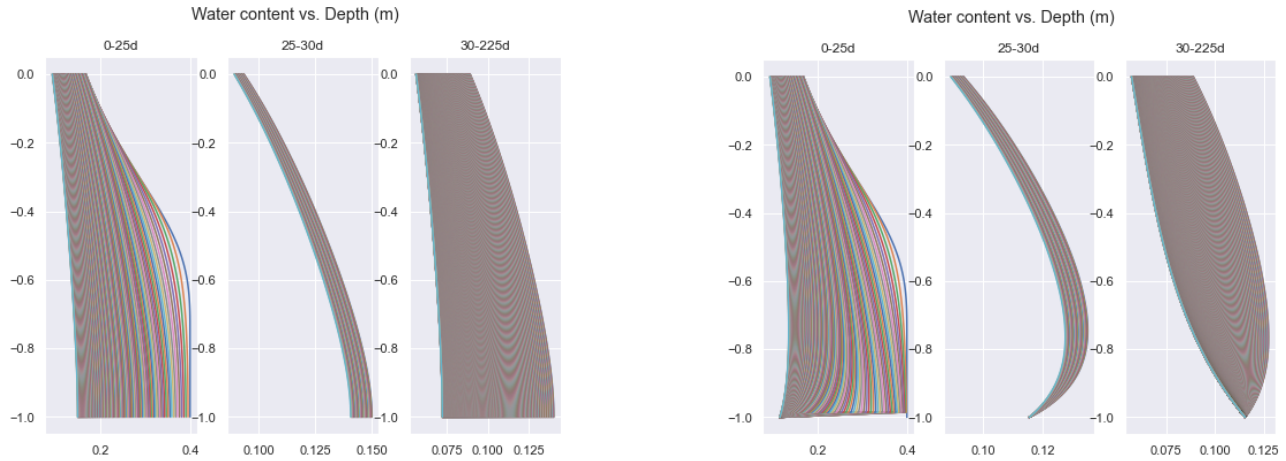

Figure 11: Water Content vs Depth [Left = gravity, Right = robins]

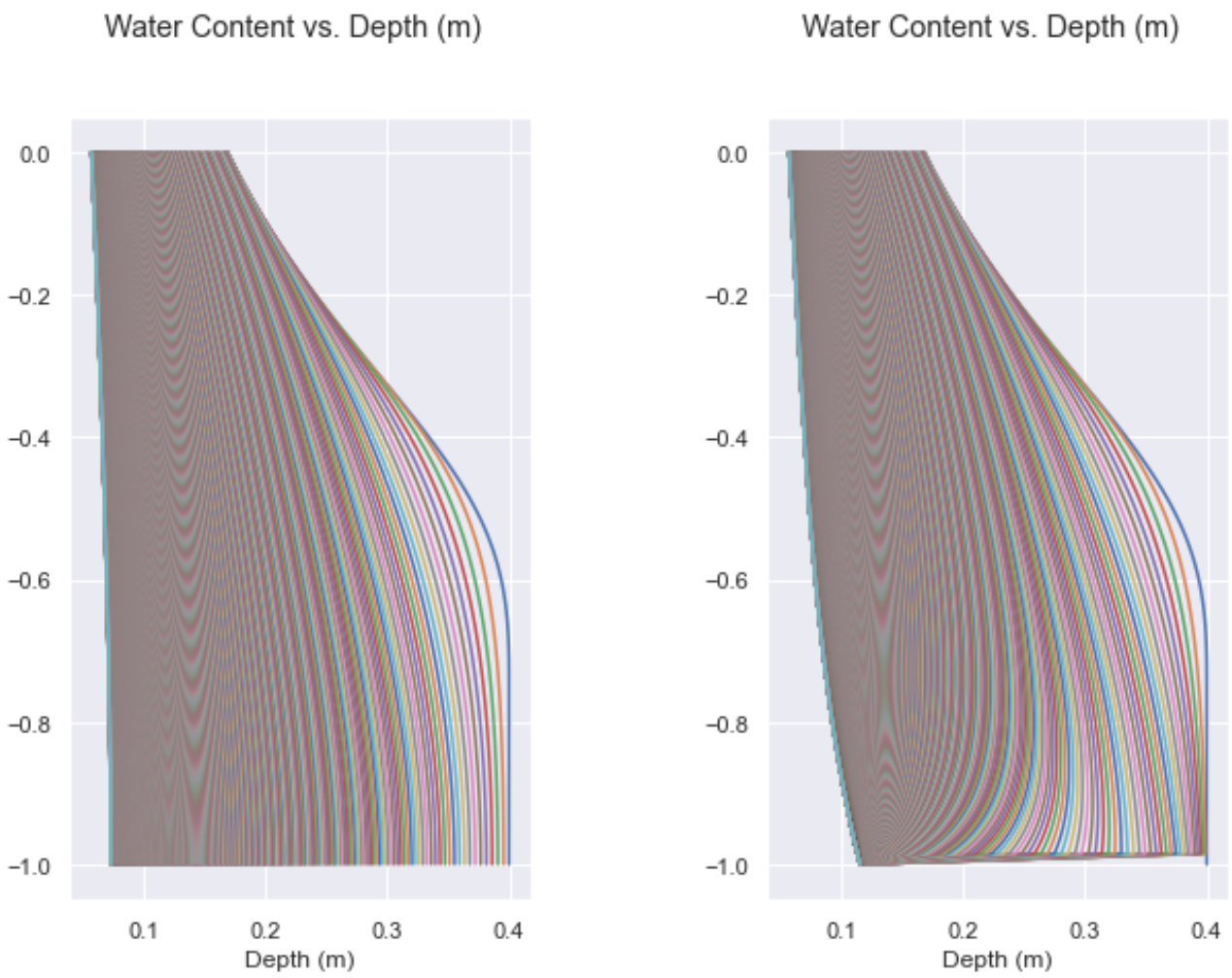

Figure 12: Water content vs Depth [Left = gravity, Right = robins]

4.2 Discussion

4.2.1 Jacobian

The jacobian matrix is used for efficiency of the code. When using the jacobian matrix in the code, the simulation is expected to run faster.
However, when actually running the code with the jacobian matrix the results were obtained after approximately 2.99 seconds, where as when the jacobian matrix was removed the results were obtained after approximately 1.97 seconds.

4.2.2 Gravity vs Robins

The results for gravity and robins conditions can be seen in section 4.1. From these results the following can be noted.

For a gravity flow which has no pressure head gradient, the water flux versus the depth is not much different from the robins flow, figure 7. In both conditions the water flux decreases as time goes by. At the start there is a drastic decrease in both conditions, this can be seen from the ' pretty vertical' lines around day '0'. After this the rate of reduction (slope) is reduced. It can be seen that for the robins condition the water flux becomes relatively constant the more days pass by, while the gravity flow continues to decrease.

The water content and pressure head profiles present the same structure, this is logical as the water content determines the pressure head in the soil. The higher the water content, the higher the pressure head will be. For both conditions it can be seen that the water content, figure 12, increases with depth until about 0.5 meters (-0.5 meters in depth) , the increase in water content with depth is due to the drainage of the water towards the bottom of the soil. After this depth, of approximately 0.5 meters, the water content stays constant/reaches a maximum. This means that the soil cannot hold more water and has become saturated. The saturation water content is approximately 0.39 for the gravity condition and 0.4 for the robins condition. This is in accordance to the volumetric saturation water content that was defined in the code.

Because the soil becomes saturated with water at the depth of approximately 0.5 meters, there is no more additional water added below this level. As a result the pressure head will not follow an increasing tendency as one moves further down into the soil. For gravity conditions the slope of the pressure head with depth decreases slightly, whereas for the robins condition this is more drastic.

5 Conclusions

This program predicts the behavior of water flow in unsaturated zones of the soil. Jacobian matrix is used to potentially make the code more efficient but doesn't succeed. The effects of changes in boundary conditions and initial conditions is explored.
Mass Balance Mass balance helps in checking the accuracy of the program. The water inflow should be equal to water outflow and storage in the soil. This storage results in saturation of the soil. Theoretically, mass balance has been defined in equation (8). However, some error is expected since the code is not the exact representation of natural conditions and discretization in space and time is not small enough. The mass error will be given by,

$$\Delta M = q_{In} - q_{Out} + storage \tag{13}$$

References

Benettin, Paolo, et al. (2015). *Modeling chloride transport using travel time distributions at Plynlimon, Wales.* Water Resources Research 51.5 (2015): 3259-3276.