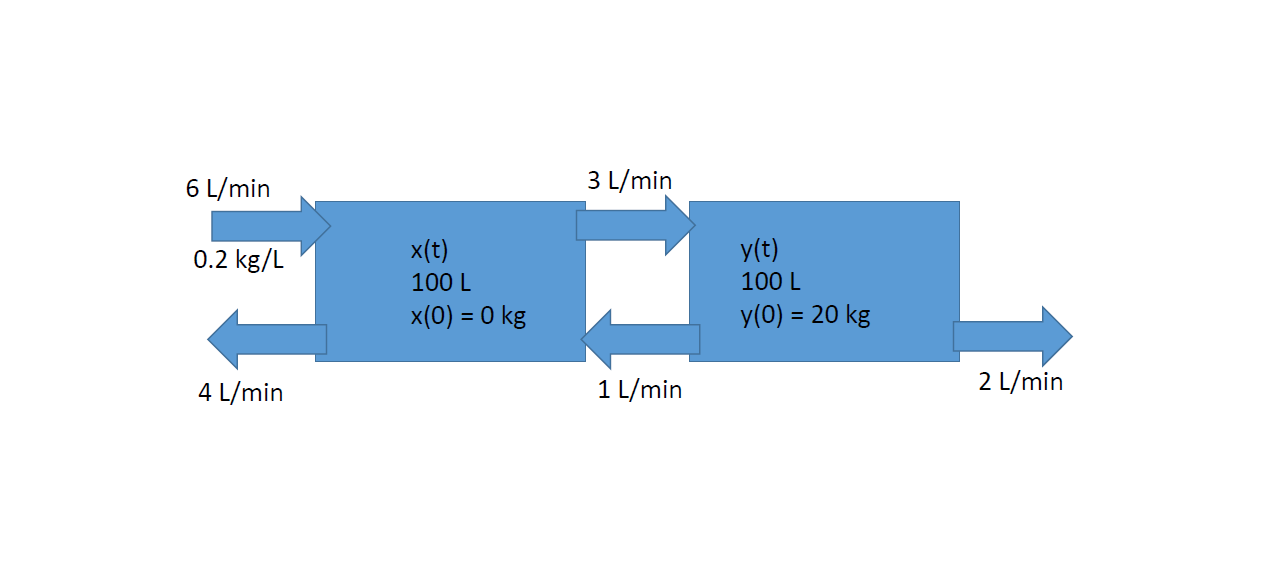
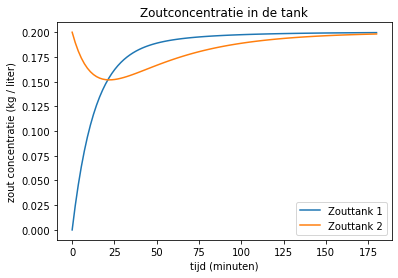
Verslag opdracht Zout tanken Koen van Heertum





Contents

[Differentiaalvergelijking 3](#_Toc57743865)

[Numerieke Berekening 4](#_Toc57743866)

[Grafiek 5](#_Toc57743867)

[Interactivity 5](#_Toc57743868)

# Differentiaalvergelijking

In dit stukje zal ik kort proberen uit te leggen wat bepaalde delen van mijn code doen. Bovendien leg ik hierdoor uit wat de differentiaalvergelijking is in de ingeleverde code.

# Ingevulde instroom/uitstroom voor beide tanks.  
constInstroom\_1 = CONCENTRATIE\_INSTROOM \* INSTROOM  
instroom\_1 = INSTROOM + UITSTROOM2\_A # Wat tank 1 instroomt  
instroom\_2 = INSTROOM2 + UITSTROOM\_B # Wat tank 2 instroomt  
instroom\_kg\_per\_s = CONCENTRATIE\_INSTROOM \* (INSTROOM + UITSTROOM2\_A)  
instroom\_kg\_per\_s\_2 = CONCENTRATIE\_INSTROOM \* (INSTROOM2 + UITSTROOM\_B)

Bovenin begin ik (na de imports) met het defineren van een aantal standaardwaarden. Onder andere de zoutconcentratie van de instroom van de 1e tank, de verhoudingen van het wegstromen van het water en de inhoud van de tanks. Vervolgens heb ik een stuk code geschreven die bijhoudt in welke verhouding het zoutwater van tank 1 naar 2 stroomt, welke verhouding van het zoutwater wat tank 1 instroomt van buitenaf komt, etc. etc.

def simLoop(stapgrootte=100):

De rest van het programma staat in een functie omdat ik ga spelen met de stapgrootte. Hij wordt hier standaard ingesteld op 100 maar kan uiteindelijk veranderd worden met de schuifbar.

zout = np.zeros(AANTAL\_STAPPEN + 1)  
zout2 = np.zeros(AANTAL\_STAPPEN + 1)  
zout[0] = 0 # beginconditie, niet echt nodig hier, maar toch.  
zout2[0] = 20 # beginconditie, hier wel nodig

Ik maak hier 2 arrays aan, een voor beide zout tanks. Deze komen later van te pas. Zout2 heeft 20 als startwaarde aangezien het begint met 20kg zout.

# Zouttank 1  
zout[stap] = zout[stap - 1] + stapgrootte \* (((concentratie\_t\_min1\_2 \* INSTROOM\_B) + constInstroom\_1) - concentratie\_t\_min1 \* uitstroom\_1)

Dit is de for-loop voor zouttank 1, hij stelt de waarde van de huidige stap in als volgt:  
1. Pakt de vorige stap en telt de stapgrootte erbij op  
2. Berekent de binnenkomende hoeveelheid zoutwater. In dit geval is het de concentratie van zouttank 2 keer de hoeveelheid van zouttank 2 (1/60). Hierna telt het de constante binnenstroom daar bovenop (constInstroom\_1).  
3. Vervolgens haalt hij de uitstroom hier vanaf. Dit is het zoutwater wat naar tank 2 loopt (3/60) en het water wat naar buiten de opdracht loopt (4/60).

# Zouttank 2  
zout2[stap] = zout2[stap - 1] + stapgrootte \* ((concentratie\_t\_min1 \* INSTROOM2\_A) - concentratie\_t\_min1\_2 \* uitstroom\_2)

Bij zouttank 2 gebeurt eigenlijk hetzelfde als zouttank 1, maar dan zonder de consistente binnenstroom.

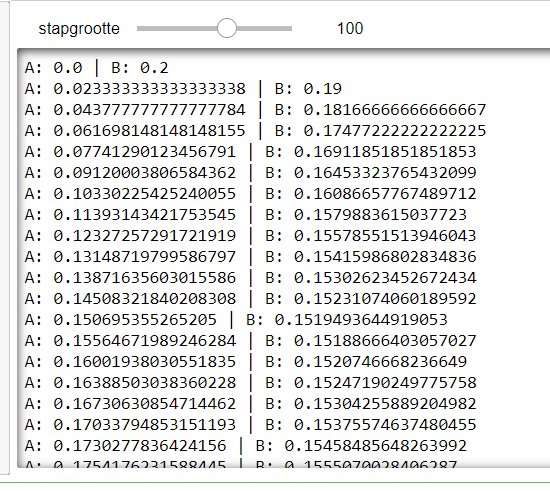
Als laatst wil ik nog even meegeven dat ik de tijd omgezet heb in minuten i.p.v. seconden. Dit komt omdat ik 0 t/m 175 minuten mooier vind staan dan 0 t/m 35.000 seconden. Het geeft een wat beter beeld van de zoutloop over de tijd. Het is makkelijker om te bedenken hoe lang 175 minuten is in vergelijking met uitrekenen van 35000 seconden.

# Numerieke Berekening

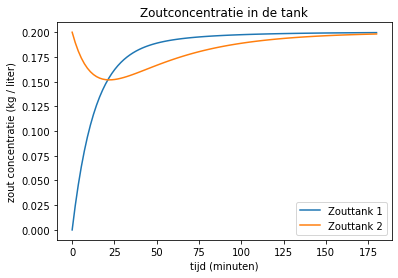
Als ik de code:

*print("A: {} | B: {}".format(concentratie\_t\_min1, concentratie\_t\_min1\_2))*

Aanzet tijdens de for-loop van mijn programma, kan ik de numerieke output van mijn programma zien. Hierdoor kan ik een tabel tonen met alle X coördinaten van beide zout tanks. Omdat deze lijst ontzettend lang is heb ik alleen de bovenste resultaten genoteerd. Dit gaat in werkelijkheid echter veel langer door.



# Grafiek



Deze afbeelding is gemaakt bij stapgrootte 100. Omdat ik mijn code interactive had gemaakt kun je die stapgrootte vergroten of verkleinen, wat andere resultaten zal geven.

# Interactivity

Zoals hierboven beschreven, heeft mijn uitwerking een slider voor de stapgrootte. Hierdoor kun je zien wat de stapgrootte zal doen met de uitwerking van mijn code. Dit was voor mij aardig snel te doen aangezien ipywidgets een handige python-library is om interactivity mee te maken.