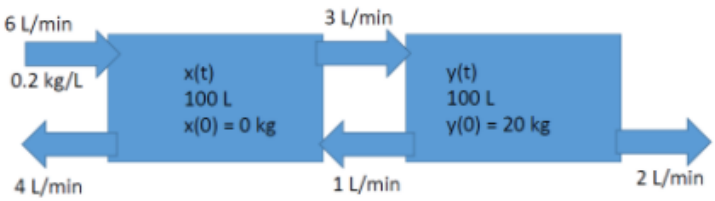
Final Assignment Simulation Modelling

Opdracht Gekoppelde zouttanks luidt: *Twee grote tanks, elk gevuld met 100 Liter vloeistof, zijn met pijpleidingen aan elkaar verbonden. De vloeistof stroomt van tank A in tank B met een snelheid van 3 L/min en van B in A met 1 L/min. Een zoutoplossing met een concentratie van 0.2 kg/L stroomt met een snelheid van 6 L/min tank A in. De oplossing stroomt met een snelheid van 4 L/min tank A uit en verlaat met een snelheid van 2 L/min tank B. Hoe verloopt de concentratie van zout door beide tanks?*

Doorgelopen Stappen:

- Stap 1: Wat weten we van de opdrachtbeschrijving?
- Stap 2: Wat weten we van de eerder gemaakte opdrachten?
- Stap 3: Wat is de moeilijkheidsgraad/nieuwe kennis?
- Stap 4: Hoe moet de output verwerkt worden?
- Stap 5: Welke formules/stappen moeten worden doorlopen om de output te maken?

<u>Stap 1:</u> De opdracht omschrijft dat 2 tanks aan elkaar vastzitten met een stroming waarbij de eerste tank alleen vloeistof binnenkrijgt (zonder van een andere tank). Deze vloeistof gaat weg of naar de tweede tank. De tweede tank heeft dezelfde inhoud, maar krijgt op een verminderde snelheid de vloeistof binnen, maar heeft al een volle zoutconcentratie. De vloeistof kan dan ook weg of naar de eerste tank terug maar in kleinere maten. Maar alle hoeveelheden vloeistoffen gaat met dezelfde hoeveelheid er ook uit, waardoor de tank niet overstroomt. Zo ziet de foto er

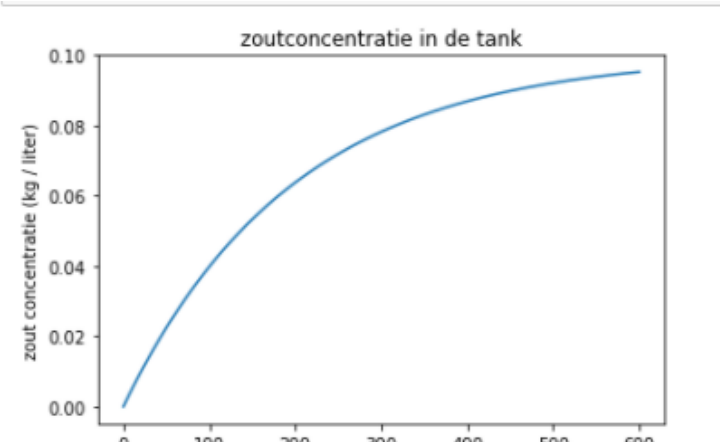


dan uit:

<u>Stap 2:</u> De 2 opdrachten die hierbij vergelijkbaar is, zijn S(E)IR en de zouttank.

Zouttank, laat zien hoe er je zout concentratie kan meten in 1 tank. De concentratie is dan 0.1 kg/l. De code hieronder laat zien welke stappen zijn ondergaan en hoe de euler methode is toegepast. De grafiek laat zien hoe de zoutconcentratie langzaam naar de 0.1 gaat en dan op het randje blijft.

```
# Nulpunten
TANK_INHOUD = 1000          # liter
INSTROOM = 5 / 1            # liter per min
CONCENTRATIE_INSTROOM = 0.1 # kg / liter
UITSTROOM = 5 / 1           # liter per min
instroom_kg_per_s = CONCENTRATIE_INSTROOM * INSTROOM

# instellingen voor simulatie:
DUUR = 10*60 # 60 minutes = 1 hour
AANTAL_STAPPEN = 100
stapgrootte = DUUR / AANTAL_STAPPEN # in seconden

tijd = stapgrootte * np.arange(AANTAL_STAPPEN + 1)

zout = np.zeros(AANTAL_STAPPEN + 1)

# de simulatieloop:
for stap in range(1, AANTAL_STAPPEN + 1):
    concentratie_t_min1 = zout[stap - 1] / TANK_INHOUD # kg / liter
    zout[stap] = zout[stap - 1] + stapgrootte * (instroom_kg_per_s - concentratie_t_min1 * UITSTROOM)


# plot
fig, ax = plt.subplots()
ax.plot(tijd, zout/1000)
ax.set_title('zoutconcentratie in de tank')
ax.set_xlabel('tijd (minuten)')
ax.set_ylabel('zout concentratie (kg / liter)')
plt.show()
```



S(E)IR, Gaat over 4 categorieën die ieder een vaste hoeveelheid input en output hebben. Deze opdracht is vooral handig omdat het gaat om meerdere categorieën in plaats van 1 bij de zouttank. Want je kan vooral van de opdracht leren hoe meerdere objecten van elkaar leren zoals als de ene als output is en voor de ander een input. De euler code is daarom het belangrijkste en leerzaamste gedeelte om te begrijpen.

```
def seir_model():

    s = numpy.zeros(num_steps + 1)
    e = numpy.zeros(num_steps + 1)
    i = numpy.zeros(num_steps + 1)
    r = numpy.zeros(num_steps + 1)

    s[0] = 1e8 - 1e6 - 1e5
    e[0] = 0
    i[0] = 1e5
    r[0] = 1e6

    for stap in range(1, num_steps + 1):
        s[stap] = s[stap - 1] + h * -1 * transmission_coeff * s[stap - 1] * i[stap - 1]
        i[stap] = i[stap - 1] + h * (transmission_coeff * s[stap - 1] * i[stap - 1]- 1 / infectious_time * i[stap - 1])
        e[stap] = e[stap - 1] + h * (transmission_coeff * s[stap - 1] * i[stap - 1]- 1/latency_time * e[stap - 1])
        r[stap] = r[stap - 1] + h / infectious_time * i[stap - 1]

    return s, e, i, r

s, e, i, r = seir_model()
```

Bij de euler/for loop kan je zien s ook gebruik maakt van i als negatieve waarde. Dat wil zeggen als er meer aantallen in i zijn dan zijn er dus minder in s. Het verloop tussen deze 2 categorieën komt door de waarde transmission_coeff. Dat geld voor elke categorie namelijk, i->e = latency_time & e->r = infectious_time. Dus als we gebruik moeten maken van meerdere categorieën dan heb je een variabele nodig die het verloop bepaalt en maak je met behulp van waar bepaalde aantallen NIET of juist WEL zijn om de totale waardes te weten van een categorie.

Stap 3: Het is duidelijk dat je S(E)IR en zouttank moet combineren om gekoppelde zouttanks op te lossen. Maar het lastigste is dat er 2 verschillende vloeistoffen zijn. Namelijk zout met water en water zonder zout. Zo kan het lastig worden als tank 1 wel zout heeft en tank 2 niet, waardoor er tank 1 ook water zonder zout erbij krijgt. En om te weten wanneer tank 1 klaar is (dus geen zoutconcentratie meer krijgt). Omdat tank 2 de hele tijd opnieuw water vraagt (meer dan tank 1 vraagt aan tank 2). Dus tank 2 moet eerder klaar zijn dan tank 1, toch?

Stap 4: De output, dus het verloop van de zoutconcentratie in beide tanks, moet worden weergegeven in grafiek. Het makkelijkste is dan om 2 lijnen te maken die ieder de zoutconcentratie laat zien, waarbij je kan vergelijken welke tank eerder klaar is dan de ander. We hebben uit stap 3 geconcludeerd dat dat tank 2 is, dus we hebben een grote kans dat het werkt tank 2 eerder klaar is dan tank 1. Natuurlijk is als uitbreiding ook fijn om de stapgrootte handmatig kunnen veranderen daarvoor kunnen we gebruik maken van ipywidgets, die een interact library heeft die we kunnen gebruiken om een slider te maken.

Stap 5:
Zouttank 1,

Tank 1 bestaat uit de volgende stromingen:
Positief: Instroom1 (met zoutconcentratie), Instroom2_1
Negatief: Uitstroom1, Uitstroom2_1
Dus de inhoud van tank 1 op tijd x en stap t is dus:
*x + t * (Instroom1 (met zoutconcentratie) + Instroom2_1 - Uitstroom1 - Uitstroom2_1)*
Dit is natuurlijk zonder zoutconcentratie
De zoutconcentratie wordt bij het begin in alleen Instroom1 aangebracht, maar later ook Instroom2_1 en verminderd bij uitstroom1 en Uitstroom2_1, deze concentraties zijn dan opgeslagen in arrays bij tijd t, deze moet worden verhoogd of verlaagd bij de uitstroom of instroom van de tank, wat dus de volgende vergelijking geeft:
*x + t * ((instroom_salt_tank1 + (concentratie_t_min1_2 * Instroom2_1)) - ((concentratie_t_min1_1 * uistroom1_2)+ (concentratie_t_min1_1 * Uitstroom1)))*

Zouttank 2,
Tank 2 bestaat uit de volgende stromingen:
Positief: Instroom1_2
Negatief: Uitstroom2, Uitstroom1_2
Dus de inhoud van tank 2 op tijd x en stap t is dus:
*x(met begin 20kg) + t * (Instroom1_2 – Uitstroom2 – Uitstroom1_2)*
Dit is natuurlijk zonder zoutconcentratie
De zoutconcentratie wordt bij het begin alleen verloren, maar later komt het er meer bij door tank 1. Deze latere concentraties zijn opgeslagen in arrays bij tijd t, deze moet worden verhoogd of verlaagd bij de uitstroom of instroom van de tank, wat dus de volgende vergelijking geeft:
*x + t * ((concentratie_t_min1_1 * Instroom1_2) - ((concentratie_t_min1_2 * uistroom2_1)+ (concentratie_t_min1_2 * uitstroom2)))*