Sutra

Release 0.2.2

Martin Korevaar, Martin van der Schans, Steven Ros, Alex Hockin

Jul 01, 2022

CONTENTS

Sutra - A python package to calculate the removal of microbial organisms in the subsoil.

CHAPTER

ONE

GETTING STARTED

Sutra is a python package to calculate the advective removal of microbial organisms (also called 'pathogens') from source to end_point.

Main features:

- Includes database of removal parameters for microbial organisms.
- Calculate the removal and concentration of the microbial organism over distance, and with time

1.1 Installation

To get the latest stable version, use:

```
python -m pip install git+https://github.com/KWR-Water/sutra.git@main
```

or:

```
pip install git+ssh://git@github.com/KWR-Water/sutra.git@main
```

1.2 Philosophy

Sutra calculates the subsurface removal of microbial organisms over a distance and with time using an analytical approach. The aim is to allow for a quick assessment of subsurface removal of microbial organisms for a growing selection of species. A database was added, starting with plant pathogens 'solani' (Dickeya solani), 'carotovorum' (Pectobacterium carotovorum), and 'solanacearum' (Ralstonia solanacearum). Additional species will be added once more data will become available.

CHAPTER

TWO

FAQ

Frequently Asked Questions

CHAPTER

THREE

GLOSSARY

Key definitions and abbreviations

alpha0: ‘reference sticky coefficient’, for a reference pH [pH0]
alpha: ‘sticky coefficient’, pH corrected
pH0: reference pH for which alpha0 was determined
mu1: inactivation coefficient [day-1]
k_att: attachment coefficient [day-1]
lambda: removal coefficient [day-1], equals k_att + mu1
mbo: abbreviation for microbial organisms

CHAPTER

FOUR

API REFERENCE

4.1 sutra.removal_functions module

class sutra.removal_functions.**MicrobialRemoval**(*organism:* Organism = *'carotovorum'*, *alpha0_suboxic=None*, *alpha0_anoxic=None*, *alpha0_deeply_anoxic=None*, *pH0_suboxic=None*, *pH0_anoxic=None*, *pH0_deeply_anoxic=None*, *mu1_suboxic=None*, *mu1_anoxic=None*, *mu1_deeply_anoxic=None*, *organism_diam=None*)

Bases: object

Class to calculate removal (rate) for a given microbial organism.

organism: object
The microbial organism object with the microbial organism (mbo) of interest

organism_dict: dictionary

'alpha0': float
reference_collision_efficiency [-] per redox zone ('suboxic', 'anoxic', deeply_anoxic')

'pH0': float
reference pH for calculating collision efficiency [-] per redox zone ('suboxic', 'anoxic', deeply_anoxic')

'organism_diam': float
diameter of pathogen/species [m]

'mu1': float
inactivation coefficient [1/day] per redox zone ('suboxic', 'anoxic', deeply_anoxic')

calc_advective_microbial_removal(*grainsize=0.00025*, *temp_water=11.0*, *rho_water=999.703*, *pH=7.5*, *por_eff=0.33*, *conc_start=1.0*, *conc_gw=0.0*, *redox='anoxic'*, *distance_traveled=1.0*, *traveltime=100.0*, *mu1=None*, *alpha0=None*, *pH0=None*, *organism_diam=None*)

Calculate the advective microbial removal of microbial organisms from source to end_point.

For more information about the advective microbial removal calculation:
BTO2012.015: Ch 6.7 (page 71-74)

Parameters

- **lambda** (*float*) – 'removal rate' [day-1] (redox dependent) –> calculated
- **redox** (*str*) – redox condition ['suboxic','anoxic','deeply_anoxic']

- **mu1** (*float*) – inactivation coefficient [day-1]
- **por_eff** (*float*) – effective porosity [-]
- **grainsize** (*float*) – grain diameter of sediment [m]
- **pH** (*float*) – pH of the water [-]
- **pH0** (*float*) – reference pH for which alpha0 was determined
- **temp_water** (*float*) – Water temperature [degrees celcius]
- **rho_water** (*float*) – Water density [kg m-3]
- **alpha** (*float*) – 'sticky coefficient' [-], pH corrected
- **alpha0** (*float*) – 'reference sticky coefficient', for a reference pH [pH0]
- **organism_diam** (*float*) – organism/species diameter [m]
- **v_por** (*float*) – porewater velocity [m/d]
- **conc_start** (*float*) – starting concentration
- **conc_gw** (*float*) – initial groundwater concentration
- **distance_traveled** (*float*) – distance between points [m]
- **traveltime** (*float*) – time between start and endpoint [days]
- **Calculates** –
- **----------** –
- **C_final** (*float*) – final concentration [N/L]

Return type
C_final

calc_lambda(*redox='anoxic'*, *mu1=0.149*, *mu1_std=0.0932*, *por_eff=0.33*, *grainsize=0.00025*, *pH=7.5*, *temp_water=10.0*, *rho_water=999.703*, *alpha0=0.001*, *pH0=7.5*, *organism_diam=2.33e-08*, *v_por=0.01*)

For more information about the advective microbial removal calculation: BTO2012.015: Ch 6.7 (page 71-74)

Calculate removal coefficient lambda [/day].

Parameters

- **redox** (*str*) – redox condition ['suboxic','anoxic','deeply_anoxic']
- **mu1** (*float*) – inactivation coefficient [day-1]
- **por_eff** (*float*) – effective porosity [-]
- **grainsize** (*float*) – grain diameter of sediment [m]
- **pH** (*float*) – pH of the water [-]
- **pH0** (*float*) – reference pH for which alpha0 was determined
- **temp_water** (*float*) – Water temperature [degrees celcius]
- **rho_water** (*float*) – Water density [kg m-3]
- **alpha** (*float*) – 'sticky coefficient' [-], pH corrected
- **alpha0** (*float*) – 'reference sticky coefficient', for a reference pH [pH0]

- **organism_diam** (*float*) – Organism/species diameter [m]
- **v_por** (*float*) – porewater velocity [m/d]
- **const_BM** (*float*) – Boltzmann constant [1,38 × 10-23 J K-1]
- **Calculates** –
- **----------** –
- **lambda** (*float*) – k_att + mu_1 'removal rate' [day-1]
- **k_att** (*float*) – attachmant rate [day-1]

Return type
lambda, k_att

class sutra.removal_functions.**Organism**(*organism_name*, *removal_function='mbo'*)

Bases: object

Placeholder class which includes removal parameters for a selection of microbial organisms ('mbo'). For now dictionary includes the plant pathogens: 'solani' (Dickeya solani), 'carotovorum' (Pectobacterium carotovorum), and 'solanacearum' (Ralstonia solanacearum).

removal_paramaters_dict: dict .. attribute:: organism_name

species_name of the substance

type
String

'alpha0'

reference_collision_efficiency [-] per redox zone ('suboxic', 'anoxic', deeply_anoxic')

Type
float

'pH0'

reference pH for calculating collision efficiency [-] per redox zone ('suboxic', 'anoxic', deeply_anoxic')

Type
float

'organism_diam'

diameter of pathogen/species [m]

Type
float

'mu1'

inactivation coefficient [1/day] per redox zone ('suboxic', 'anoxic', deeply_anoxic')

Type
float

CHAPTER

FIVE

ABOUT

SUTRA is developed by KWR Water Research Institute. The code is based on the equations given in chapter 6.7 of BTO 2012.015 “Gezondheidsrisico’s van fecale verontreiniging” [in Dutch], pp 71-74.

5.1 License

5.2 Acknowledgements

This tool was co-financed by the Horizon 2020 research and innovation programme under grant number 869171 -B-Watersmart and by the BTO Joint Research Programme of the Dutch and Flemish Water Utilities. The main contributors are: Steven Ros, Alex Hockin, Martin Korevaar, Carina Eisfeld and Martin van der Schans.

CHAPTER

SIX

KNOWN ISSUES

1. A database was added, starting with plant pathogens ‘solani’ (Dickeya solani), ‘carotovorum’ (Pectobacterium carotovorum), and ‘solanacearum’ (Ralstonia solanacearum). Additional species are not yet available, but will be added once more data will become available.

CHAPTER

SEVEN

TUTORIAL

7.1 Overview

Sutra is a python package to calculate the advective removal of microbial organisms (also called 'pathogens') from source to end_point.

Main features:

- Includes database of removal parameters for microbial organisms.
- Calculate the removal and concentration of the microbial organism over distance, and with time

The code is based on the equations given in chapter 6.7 of BTO 2012.015 "Gezondheidsrisico's van fecale verontreiniging" [in Dutch], pp 71-74. The equations used are described below.

7.2 Background information

During transport in the subsurface, microbial organism removal takes place by both attachment to the soil matrix and by inactivation. The virus concentration 'C' [m-3] through steady-state transport of microbial organisms along pathlines in the saturated groundwater can be approximated by:

$$\frac{dC}{dx} + \frac{k_{att}+\mu_1}{v} C = 0$$

Where k_att + mu1 –> equal to removal rate 'lambda' 'k_att': attachment coefficient [day-1] 'mu1': inactivation coefficient [day-1] x: the distance traveled [m] v: the porewater velocity [m day-1] or 'darcyflux divided by the effective porosity'

Assuming that the background concentration of the relevant microbial organism equals 0, the relative removal 'C_x/C0' can be calculated as follows.

$$log\left(\frac{C_x}{C_0}\right) = -\frac{(k_{att}+\mu_1)}{\ln(10)}\frac{x}{v}$$

The attachment coefficient 'k_att' depends on the effective porosity 'epsilon', the grain diameter of the sediment 'd_c', 'sticky coefficient' alpha [day-1], the porosity dependent Happel's parameter 'A_s', diffusion constant 'D_BM' [m2 day-1], and the porewater velocity [m day-1].

$$k_{att} = \frac{3}{2} \cdot \frac{(1-\varepsilon)}{d_c} \cdot \alpha 4 A_s^{1/3} \cdot \left(\frac{D_{BM}}{d_c \cdot \varepsilon \cdot v}\right)^{2/3} \cdot v$$

The sticky coefficient alpha is determined by coefficient 'alpha0', which depends on both the soil type and the type of organism. Alpha0 is being determined for a reference pH [pH0], e.g. pH=7.5. Alpha relates to alpha0 as follows [corrected for different pH].

$$\alpha = \alpha_0 \cdot 0{,}9^{\left(\frac{pH-pH_0}{0{,}1}\right)}$$

The other parameters are calculated as follows:

Happel's porosity dependent parameter

$$A_s = 2 \cdot \frac{(1-\gamma^5)}{(2-3\gamma+3\gamma^5-2\gamma^6)}$$

Where:

$$\gamma = (1-\varepsilon)^{1/3}$$

Boltzmann diffusion coefficient:

$$D_{BM} = \frac{K_B \cdot (T+273)}{3\pi \cdot d_p \cdot \mu} \cdot 86400$$

with Boltzmann constant K_B [1,38 × 10-23 J K-1], organism diameter d_p [m], water temperature T [degr C], and conversion factor 86,400 [s day-1].

The dynamic viscosity 'mu' [kg m-1 s-1] depends on the groundwater density 'rho'. The water density is assumed to be 999.7 [kg m-3], representative for fresh groundwater in the Netherlands under a reference temperature of 12 degrees centigrade.

$$\mu = \frac{\rho \cdot 497 \cdot 10^{-6}}{(T+42{,}5)^{3/2}}$$

7.3 Steps

Operating the microbial organism removal involves 2 steps:

#. Run/load the removal_functions.MicrobialRemoval class to retrieve the default microbial (removal) parameters, if present in the database. Otherwise, an empty dataframe is returned. #. Run removal_functions.calc_advective_microbial_removal to calculate the final concentration after a distance and time traveled.

Now, let's try some examples. First we import the necessary python packages

```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: import os

In [4]: import sys

In [5]: from pathlib import Path

In [6]: import sutra.removal_functions as rf
```

Scenario A: Calculate removal of a microbial organism using default database parameters.

```
## Default removal parameters ##
In [7]: organism_name = "carotovorum"

# Redox condition: 3 options ['deeply_anoxic','anoxic','suboxic']
In [8]: redox_cond = 'anoxic'

# organism diameter [m]
In [9]: organism_diam = 1.803e-6

# Starting concentration
In [10]: conc_start = 1.

# Ambient groundwater concentration
In [11]: conc_gw = 0.

# effective porosity
In [12]: por_eff = 0.33

# Sediment grainsize
In [13]: grainsize = 0.00025

# pH of the groundwater
In [14]: pH_water = 7.5

# Water temperature
In [15]: temp_water = 10.

# Water density [kg m-3]
In [16]: rho_water = 999.703
```

(continues on next page)

(continued from previous page)

```
# Distance traveled along pathline [m]
In [17]: distance_traveled = 100.

# Time traveled [days]
In [18]: traveltime = 1.

# Porewater velocity [m day-1]
In [19]: porewater_velocity = distance_traveled / traveltime
```

First initialize a class for calculating the removal of an organism.

```
In [20]: mbo_removal_scenA = rf.MicrobialRemoval(organism = organism_name)

In [21]: removal_parameters = mbo_removal_scenA.removal_parameters

# Return the (default) removal parameter values
In [22]: print(removal_parameters)
{'organism_name': 'carotovorum', 'alpha0': {'suboxic': 0.3, 'anoxic': 0.577, 'deeply_
↪anoxic': 0.577}, 'pH0': {'suboxic': 7.5, 'anoxic': 7.5, 'deeply_anoxic': 7.5},
↪'organism_diam': 1.803e-06, 'mu1': {'suboxic': 1.2664, 'anoxic': 0.1279, 'deeply_anoxic
↪': 0.1279}}
```

Calculate final concentration after advective microbial removal

```
# Calculate final concentration and print it
In [23]: C_final_default = mbo_removal_scenA.calc_advective_microbial_removal(grainsize␣
↪= grainsize,
   ....:                                        temp_water = temp_water, rho_water = rho_
↪water,
   ....:                                        pH = pH_water, por_eff = por_eff,
   ....:                                        conc_start = conc_start, conc_gw = conc_gw,
   ....:                                        redox = redox_cond,
   ....:                                        distance_traveled = distance_traveled,
   ....:                                        traveltime = traveltime)
   ....:

In [24]: print(C_final_default)
8.2065781569924e-12

# Print lambda (default): removal rate [day-1]
In [25]: print(mbo_removal_scenA.lamda)
25.526085068992856
```

Scenario B1: Manual input of removal parameters, not included in default database

```
## Removal parmeters ##
# Organism name
In [26]: organism_name = "MS2"

# Redox condition: 3 options ['deeply_anoxic','anoxic','suboxic']
In [27]: redox_cond = 'anoxic'
```

(continues on next page)

(continued from previous page)

```
# alpha0: 'sticky coefficient' [-]
In [28]: alpha0 = 0.001

# Reference pH for calculating 'alpha' [-]
# --> if pH == pH0, then collision efficiency alpha equals the value of alpha0
In [29]: pH0 = 7.5

# time dependent inactivation coefficient 'mu1' [day-1]
In [30]: mu1 = 0.149

# organism diameter [m]
In [31]: organism_diam = 2.33e-8
```

Add remaining 'ambient' input parameters

```
# effective porosity
In [32]: por_eff = 0.33

# Sediment grainsize
In [33]: grainsize = 0.00025

# pH of the groundwater
In [34]: pH_water = 7.5

# Water temperature
In [35]: temp_water = 10.

# Water density [kg m^-3]
In [36]: rho_water = 999.703

# Distance traveled along pathline [m]
In [37]: distance_traveled = 100.

# Time traveled [days]
In [38]: traveltime = 1.

# Porewater velocity [m day-1]
In [39]: porewater_velocity = distance_traveled / traveltime

# Starting concentration
In [40]: conc_start = 1.

# Ambient groundwater concentration
In [41]: conc_gw = 0.
```

Initialize a class for calculating the removal of an organism

```
In [42]: mbo_removal_B1 = rf.MicrobialRemoval(organism = organism_name)
```

Calculate (relative) concentration following advective microbial removal

```
In [43]: C_final_B1 = mbo_removal_B1.calc_advective_microbial_removal(grainsize =␣
↪grainsize,
```

(continues on next page)

(continued from previous page)

```
   ....:                                        temp_water = temp_water, rho_water = rho_
↪water,
   ....:                                        pH = pH_water, por_eff = por_eff,
   ....:                                        conc_start = conc_start, conc_gw = conc_gw,
   ....:                                        redox = redox_cond,
   ....:                                        distance_traveled = distance_traveled,
   ....:                                        traveltime = traveltime,
   ....:                                        organism_diam = organism_diam,
   ....:                                        mu1 = mu1,
   ....:                                        alpha0 = alpha0,
   ....:                                        pH0 = pH0 )
   ....:

# print final concentration
In [44]: print(C_final_B1)
0.38739172625173746
```

Print the attachment coefficient 'k_att' and removal rate 'lambda'

```
# k_att, calculated
In [45]: print(mbo_removal_B1.k_att)
0.7993188853572424

# lambda, calculated
In [46]: print(mbo_removal_B1.lamda)
0.9483188853572424
```

Scenario B2: An alternative way to enter removal parameters and calculate the final concentration Should compare to previous input, be aware to enter the correct redox related values for 'anoxic' situation

```
In [47]: mbo_removal_B2 = rf.MicrobialRemoval(organism = organism_name,
   ....:             alpha0_suboxic=None,
   ....:             alpha0_anoxic=0.001,
   ....:             alpha0_deeply_anoxic=None,
   ....:             pH0_suboxic=None,
   ....:             pH0_anoxic=pH0,
   ....:             pH0_deeply_anoxic=None,
   ....:             mu1_suboxic=None,
   ....:             mu1_anoxic=mu1,
   ....:             mu1_deeply_anoxic=None,
   ....:             organism_diam=organism_diam,
   ....:             )
   ....:
```

Read removal parameters from 'removal_parameters'. Check these values as follows

```
In [48]: removal_parameters = mbo_removal_B2.removal_parameters

In [49]: print(removal_parameters)
{'organism_name': 'MS2', 'alpha0': {'suboxic': None, 'anoxic': 0.001, 'deeply_anoxic':␣
↪None}, 'pH0': {'suboxic': None, 'anoxic': 7.5, 'deeply_anoxic': None}, 'organism_diam
↪': 2.33e-08, 'mu1': {'suboxic': None, 'anoxic': 0.149, 'deeply_anoxic': None}}
```

Calculate the final concentration, removal parameters for redox condition 'anoxic' (given by 'redox_cond')

```
# Only include 'ambient'/'physical' parameters (removal parameters loaded preiously)
In [50]: C_final_B2 = mbo_removal_B2.calc_advective_microbial_removal(grainsize =␣
↪grainsize,
   ....:                                        temp_water = temp_water, rho_water = rho_
↪water,
   ....:                                        pH = pH_water, por_eff = por_eff,
   ....:                                        conc_start = conc_start, conc_gw = conc_gw,
   ....:                                        redox = redox_cond,
   ....:                                        distance_traveled = distance_traveled,
   ....:                                        traveltime = traveltime)
   ....:

# k_att, calculated
In [51]: k_att = mbo_removal_B2.k_att

In [52]: print(k_att)
0.7993188853572424

# lambda, calculated
In [53]: lamda = mbo_removal_B2.lamda

In [54]: print(lamda)
0.9483188853572424

# Final concentration
In [55]: print(C_final_B2)
0.38739172625173746
```

Notice that the output concentrations C_final_B1 & C_final_B2 are equal

```
In [56]: print(C_final_B1 == C_final_B2)
True
```

PYTHON MODULE INDEX

S

INDEX