Urban Simulation Platform Projects

MICROBIAL SYSTEMS

Written by: Alireza Adli

Project Reseacher: Narges Rahimi

Supervisor: Professor Ursula Eicker

Project Integrator: Alireza Adli

Next-Generation Cities Institute (NGCI) September 2022

This project is a part of The NGCI's Urban Simulation Platform project. The platform has integrated the application of NGCI's research projects in Python programming language. These projects are developed by graduate students and researchers of the institute towards urban sustainable development in six main areas: energy systems, building, transportation, vegetation, waste and recycling, liveability.

The integration of a project refers to further development of a research project in Object Oriented Programming (OOP) paradigm following the coding style of the platform. This is done in order to employ multiple projects in a single workflow.

For more information about this piece of software and documentation, contact Alireza Adli (alireza.adli@concordia.ca)
For detailed technical information of the project read Narges Rahimi Master's thesis report on the same subject (Link).

Contents

1 Model Description **3**

2 The Project Structure **3**

2.1 MicrobialSystem 3

2.2 SingleChamberElectrolysisCell 4

2.3 DualChamberElectrolysisCell 4

2.4 FuelCell 5

2.5 MicrobialSystemWorkflow 5

3 3rd Party Software **6**

4 Input Data **6**

5 Constants **7**

6 Output Data **10**

7 Limitations **11**

8 Technical Performance **11**

1 Model Description

A microbial system for wastewater treatment and energy recovery produces hydrogen and power from wastewater.

In this project, three microbial systems have been studied and implemented in Python programming: Single Chamber Microbial Electrolysis Cell (SCMEC), Dual Chamber Microbial Electrolysis Cell (DCMEC) and Microbial Fuel Cell (MFC). The project considers the treatment of one litre wastewater. But given the number of inhabitants and water consumption, the system can output results based on the corresponding amount of water.

In the first step, the wastewater should be collected from urban areas such as residential buildings. Then, collected wastewater will be injected into the bio-electrochemical systems, including SCMEC, DCMEC, and MFC to treat wastewater and generate hydrogen and electricity. In the first scenario power output of MFC can be used in two different applications such as providing energy for electric vehicle charging stations or urban areas electricity demand. In the second scenario, as the main product of SCMEC and DCMEC, hydrogen can be considered as a type of energy or to feed MFCs to generate power. It should be noted that to run the MECs, the applied voltage can be covered by renewable sources of green electricity such as solar panels and wind turbines.

2 The Project Structure

This project consists of five classes: four classes for modelling different microbial systems and one work-flow class to carry out different simulations.

2.1 MicrobialSystem

This class has been developed in microbial_systems_abc.py module. This is an abstract base class (ABC) which has been defined as a template for developing any microbial system

models. ABCs cannot be instantiated but they are being developed as the parent of other classes. This is being done when modelling different types of the same system. In this way, the parent class will be used as a template with attributes that are common in different types and main functionalities. These main functionalities are being defined as abstract methods. Any other sub system (different types) should define (override) all the abstract methods of the parent class otherwise, that subclass cannot be instantiated. ABCs raise the control of the main developer over the project.

For example, in this project all new child classes should include a mass_balance(t, values) method with the mentioned parameters. Otherwise, they cannot be instantiated. This is the interface (abstract method) of this Abstract Base Class. This rule is not applied to DualChamberElectrolysisCell class because it inherits from the SingleChamberElectrolysisCell class.

2.2 SingleChamberElectrolysisCell

This class has been developed in single_chamber_electrolysis_cell.py module. SingleChamberElectrolysisCell simulates the system by inheriting constants and abstract methods (interfaces) of the MicrobialSystem class of microbial_system_abc module. So the former should be imported for this module to work. To instantiate this class, it is only needed to assign the class to a variable. Example: single = SingleChamberElectrolysisCell()

2.3 DualChamberElectrolysisCell

This class has been developed in dual_chamber_electrolysis_cell.py module. A dual chamber electrolysis cell is very similar to a single chamber one, so in this project, the DualChamberElectrolysisCell inherits from the SingleChamberElectrolysisCell class for constants and some functionalities. So the SingleChamberElectrolysisCell class of single_chamber_electrolysis_cell.py module should be imported in

dual_chamber_electrolysis_cell.py. To instantiate this class, it is only needed to assign the class to a variable.

Example: dual = DualChamberElectrolysisCell()

2.4 FuelCell

This class has been developed in microbial_fuel_cell.py module. FuelCell simulates this microbial system by inheriting constants and abstract methods (interfaces) of the MicrobialSystem class of microbial_system_abc. So the former should be imported for this module to work. To instantiate this class, it is only needed to assign the class to a variable.

Example: microbial_fuel_cell = FuelCell()

2.5 MicrobialSystemWorkflow

This class has been developed in microbial_system_workflow.py module. The module is accessible through below Gitlab link:
https://rs-loy-gitlab.concordia.ca/alireza.adli/microbial-system-workflow

This class simulate desirable microbial system (currently three systems are available which have been mentioned in previous subsections). There are two parameters needed to instantiate this class: Name of the microbial system (its class should be instantiated either directly or by assigning it to a variable beforehand.) and the time-step.

Example: workflow_1 = MicrobialSystemWorkflow(SingleChamberElectrolysisCell(), (0, 152))

In above example, a single chamber microbial system with time-steps in range zero and 152 has been simulated. There are different outputs available: CSV file or a Pandas dataframe. It is only needed to call the method.

Example: workflow_1.system_output_csv()

The above example put out a CSV file including the resulted dataset.

The plotting() method can be called and used in different ways. The program will ask the user about the desirable plot. The plot can be saved if True has been assigned to the save_plot keyword argument. The default value is False.

Example: workflow_1.plotting(save_plot=True)

3 3rd Party Software

Following Python packages have been used to develop the project, and are needed to run the project's program:

- **pandas**
- **numpy**
- **scipy**
- **matplotlib**

Python packages are free and accessible from www.pypi.org. They can be also installed directly from pycharm, pip or Anaconda prompt.

4 Input Data

All the three systems have been designed based on one litre of wastewater as input, by default. But they can be also run with any other amount of wastewater. In this case number of inhabitants and water consumption should be input when instantiating a system. Influent flow is computed by multiplying number of inhabitants by the amount of water consumption. This has been developed as a method of the project's workflow (workflow will be explained in the XXX section). Below table shows project's parameters and their corresponding names in the Python program.

Table 1: Input parameters and their corresponding data members and/or method in the Python program.

Parameter (unit)	Python
Number of inhabitants	number_of_inhabitants
Water consumption (litre)	water_consumption
Influent flow (litre)	influent_flow

5 Constants

The systems are working at their best performance with a number of constants (based on the conducted research project). These constants are represented in four following tables: constants which are common among the three systems with same values (table 2), constants which are common among the systems but with different values (table 3), constants which are specific to microbial electrolysis systems (table 4), and constants which are specific to the microbial fuel cell (table 5).

Table 2: Common constants between the three microbial systems (which have the same values), along with their corresponding data members in the Python program.

Constant (unit)	Python
$F\ (C/mole)$	faraday
$R_1\ (J/mole.K)$	ideal_gas
$m\ (mole\ e^-/mole\ H_2)$	electrons_per_mole
$K_{MEC}\ (mg\ M/L)$	mediator_half
$K_{d,a}\ (1/d)$	andophilic_decay

$K_{s,a}\ (mg\ S/L)$	andophilic_half
$K_{s,m}\ (mg\ S/L)$	methanogenic_half
$X_{max,1}\ (mg\ S/L)$	andophilic_limitation
$\mu_{max,a}\ (1/d)$	andophilic_max_growth
$\gamma\ (mg\ M/mole\ M^{-1})$	mediator_molar_mass
α_1	andophilic_biofilm_retention
α_2	methanogenic_biofilm_retention
$S_A\ (m^2)$	anode_surface_area

Table 3: Common constants between the three microbial systems (which have different values), along with their corresponding data members in the Python program.

Constant (unit)	Python
$M_T\ (mg\ M/mg\ X^{-1})$	mediator_fraction
$K_{d,m}\ (1/d)$	methanogenic_decay
$K_R\ (L/mg\ X)$	curve_slope
$q_{max,a}\ (mg\ S/mg\ X\ d)$	andophilic_reaction_max
$q_{max,m}\ (mg\ S/mg\ X\ d)$	methanogenic_reaction_max
$R_{min}\ (\Omega)$	resistance_min
$R_{max}\ (\Omega)$	resistance_max
$V_r\ (L)$	reactor_volume
$Y_h\ (ml\ H_2/mg\ X)$	hydrogen_yield
$Y_M\ (mg\ M/mole\ A^{-1})$	mediator_yield
$\mu_{max,m}\ (1/d)$	methanogenic_max_growth
M_{Ox_0}	oxidized_mediator_initial

X_{a0} (mg/L)	andophilic_population_initial
X_{m0} (mg/L)	methanogenic_population_initial
I_{MEC}&I_{MFC} (A)	initial_current_density
F_{inlet}	influent_flow_initial

Table 4: Microbial Electrolysis Cell systems' constants and their corresponding data members in the Python program.

Constant (unit)	Python
F_1 $(A.d/mole)$	faraday_ec
R_1 $(J/mole.K)$	ideal_gas_ec
H_2 $(mg\,A/L)$	hydrogen_2_saturation
E_{app} (V)	applied_potential
E_{CEF} (V)	counter_electromotive_force
$[H_2]$ (mg/L)	hydrogen_2_dissolved
K_h (mg/L)	hydrogenotrophic_half
$K_{d,h}$ $(1/d)$	hydrogenotrophic_decay
$X_{max,2}$ $(mg\,X/L)$	max_biomass
Y_h $(ml\,H2/mg\,X)$	hydrogen_yield_methanogenic
Y_{H_2}	hydrogen_yield
β	oxidation_coefficient
S_0 (mg/L)	acetate_initial
X_{h0} mg/L	hydrogenotrophic_population_initial

Table 5: Constants and their corresponding data members in the Python program Python program Python program.

Constant (unit)	Python
E_{min} (V)	e_ocv_min
E_{max} (V)	e_ocv_max
K_X $(L/mg\,X)$	steepness
$X_{max,m}$ $(mg\,X/L)$	biofilm_space_limitation
E_{CEF} (V)	counter_electromotive_force
$[H_2]$ (mg/L)	hydrogen_2_dissolved
K_h (mg/L)	hydrogenotrophic_half
$K_{d,h}$ $(1/d)$	hydrogenotrophic_decay
$X_{max,2}$ $(mg\,X/L)$	max_biomass
Y_h $(ml\,H2/mg\,X)$	hydrogen_yield_methanogenic
Y_{H_2}	hydrogen_yield
β	oxidation_coefficient
S_0 (mg/L)	acetate_initial
X_{h0} mg/L	hydrogenotrophic_population_initial

6 Output Data

The output of each system can be reached either through a pandas dataframe or a Comma Separated Value (CSV). After instantiating MicrobialSystemWorkflow class, below methods return the corresponding system's output pandas dataframe:

- single_chamber_dataframe()

- dual_chamber_dataframe()
- fuel_cell_dataframe()

system_output_csv() return output CSV file, based on the system model that has been chosen in instantiation. Table 6 shows output data names as in the pandas dataframe and CSV files with their corresponding data members in Python and parameter names in the project report. It should be mentioned that, for all the three systems, acetate has been considered as the main substrate. This has been referred in the report's formulas with 'S'. Time is also a timestep from the whole number of iterations.

Table 6: Output data of a single chamber microbial electrolysis system

Parameter (unit)	Python	CSV
t	time	Time
(S) (mg_s/L)	acetate	Acetate (mg_s/L)
(X_a) (mg_x/L)	andophilic_population	Andophilic Population (mg_x/L)
S (mg/L)	methanogenic_popoulation	Methanogenic Popoulation (1/d)
S (mg/L)	acetate	Acetate (mg_s/L)

7 Limitations

Limitations have not been defined for the project.

8 Technical Performance

The output will be achieved almost less than five seconds. The performance can be slightly different based on different inputs for time intervals.