ClC_MKM v0.1 Manual

Austen Bernardi

November 22, 2021

Contents

1 Introduction **2**

2 License **2**

3 Requirements **2**

4 Installation **2**

5 Executables **3**
- 5.1 **`run_opt.py`** . . . 3
- 5.2 **`run_rpa.py`** . . . 3
- 5.3 **`parse_cycles.py`** . . . 3
- 5.4 **`opt_flows.py`** . . . 4
- 5.5 **`run_mkm.py`** . . . 4

6 Configuration files **4**
- 6.1 Systems configuration file . . . 5
- 6.2 Optimization configuration file . . . 6
- 6.3 Residual configuration file . . . 8

7 Examples **9**

8 References **9**

1 Introduction

ClC_MKM contains Python libraries and executables for steady-state kinetic modeling and optimization a ClC-ec1 Markov State Model. ClC-ec1 is a secondary-active Chloride/Proton transmembrane antiporter [4]. ClC_MKM supports optimization of kinetic rate coefficients between biologically relevant transitions against experimentally derived unitary turnover rates [3]. ClC_MKM has built-in support for optimization using SpotPy [2] or SciPy [5]. This manual describes how to use ClC_MKM to perform optimization.

2 License

ClC_MKM uses the GNU General Public License v3. See `ClC_MKM/LICENSE` for more information.

3 Requirements

ClC_MKM requires the following:

- Python 3
- SciPy [5]
- SpotPy [2]
- NumPy
- CycFlowDec [1] (https://github.com/austenb28/CycFlowDec)
- MatPlotLib

4 Installation

Download the top directory `ClC_MKM` to a directory that will store the module. Update your `$PYTHONPATH` environment variable to include the storage directory if it is not already included. Update your `$PATH` environment variable to include `ClC_MKM/bin`. For example, on a Unix based system using a bash shell, if the storage directory is `$HOME/modules`, then adding

```
export PYTHONPATH=$PYTHONPATH:"$HOME/modules"
export PATH=$PATH:"$HOME/modules/ClC_MKM/bin"
```

to `$HOME/.bashrc` properly installs ClC_MKM.

5 Executables

This section describes the executables provided in `ClC_MKM/bin`.

5.1 `run_opt.py`

`run_opt.py` `<config>`

Description
Performs optimization on the ClC-ec1 system specified by `<config>`. See section 7 for a usage examples.

Parameters
`<config>`: A configuration file containing parameters for the ClC-ec1 system. Also specifies the optimization configuration file. The format of this configuration file is specified in section 6.1.

5.2 `run_rpa.py`

`run_rpa.py` `<config>`

Description
Performs reaction path analysis using cyclic flow decomposition [1] on the first ClC-ec1 system specified by `<config>`. See section 7 for a usage examples.

Parameters
`<config>`: A configuration file containing parameters for the ClC-ec1 system. The format of this configuration file is specified in section 6.1. Only performs RPA on the first system.

5.3 `parse_cycles.py`

`parse_cycles.py` `-bn <basename> -ci <init_coeffs> -d <dir> --opp`

Description
Parses cycles generated by `run_rpa.py`, generating various output files describing the flow cycles. See section 7 for a usage examples.

Parameters
`-bn <basename>`: Basename for output filenames (default `"biological"` or `"opposite"` depending on `-opp`).

`-ci <init_coeffs>`: Name of initial coeffs file (.csv).

`-d <dir>`: Directory containing `cycle_dat.pickle` (default `"./"`).

`--opp`: Use to flip stoichiometry for opposite orientation.

5.4 opt_flows.py

opt_flows.py <config> -f <opt_dat>

Description
Writes stepwise flow values to "ion_flows_sys<j>.dat", where <j> corresponds to the index of the system specified in the configuration file.

Parameters
<config>: A configuration file containing parameters for the ClC-ec1 systems. The format of this configuration file is specified in section 6. Should directly correspond to the <config> used to generate opt_dat.

<opt_dat> (optional): Filename of output optimization coefficients. Default is "opt.dat".

5.5 run_mkm.py

run_mkm.py <config>

Description
Generates instances of the ClC-ec1 systems specified by <config>. See section 7 for a usage example.

Parameters
<config>: A configuration file containing parameters for the ClC-ec1 system. The format of this configuration file is specified in section 6.

6 Configuration files

This section describes the various configuration files used by ClC_MKM. Representative example configuration files are provided in the ClC_MKM/examples directory, and discussed in section 7. Lines may be commented using the '#' symbol. The parameter specification format for all configuration files is param = <param_list>, where <param_list> is a comma-seperated (without spaces) list of values, ex. <val1,val2,val3,...>. Space seperated text after <param_list> is ignored and can be used as comments (ex. units). If any <param_list> has length greater than one, all other <param_list> must have the same length or length one. If a <param_list> is length one and another <param_list> has length greater than one, then the value is used for all systems. In some cases, <param_list> must be length one, indicated by scalar. A listed param is considered to be required unless specified as optional.

6.1 Systems configuration file

This is the main configuration file that is used as the argument for both executables listed in section 5. See below for the full list of accepted parameters.

input_rate_file (scalar)
The input rate coefficient filename for the kinetic model. File should be comma-seperated. Line 1: parameter identifiers. Line 2: parameter values. Line 3: lower bounds for optimization. Line 4: upper bounds for optimization. If lines 3 and 4 are not present, lower and upper bounds are uniformly set to zero and infinity, respectively. See `ClC_MKM/examples/opt/custom/seed_dru_san.csv` for an example with bounds specified. Units are 1/ms.

rate_map_file (scalar)
The rate coefficient map filename for the kinetic model. Used to map coefficients to applicable transitions. See examples for formatting.

internal_pH
The pH(s) inside the vesicles of the modeled systems.

external_pH
The pH(s) outside the vesicles of the modeled systems.

internal_Cl_conc
The chloride concentration(s) in mol/m^3 inside the vesicles.

external_Cl_conc
The chloride concentration(s) in mol/m^3 outside the vesicles.

enzyme_MW
The molecular weight of the antiporter in g/mol.

lipid_MW
The molecular weight of the lipids that make up the vesicles in g/mol.

area_per_lipid
The average surface area per lipid in m^2 of the lipids that make up the vesicles.

enzyme_lipid_wtfrac
The weight fraction of enymes to lipids of the vesicles.

h_rxn_bl

The approximate height of the reactive boundary layer for vesicle surface uptake reactions.

diffusivity_Cl
The bulk diffusivity of Chlorides.

diffusivity_H
The bulk diffusivity in m^2/ms of protons.

vesicle_diam
The average diameter in m of the vesicles.

enzyme_surf_conc_sim
The surface concentration of enzymes in mol/m^2 for the simulations used to model the uptake coefficients.

opt_config_file (scalar)
Required only for **run_opt.py**. The filename of the optimization configuration file. See section 6.2 for details.

6.2 Optimization configuration file

This configuration file is specified by the main configuration file as `opt_config_file`. For use with **run_opt.py**. See below for the full list of accepted parameters.

opt_package (optional, scalar)
The name of the optimization package to use. A custom combined steepest descent/conjugate gradient method is used if unspecified. Supported packages are "scipy" and "spotpy". See section 7 for example use cases.

opt_residuals_file (scalar)
The optimization residuals filename. This file contains residual targets for specified flows, and is used to build the objective function using a sum of square residual differences. See section 6.3 for details.

opt_dat_file (optional,scalar)
Filename for the output optimization data (step, parameters, objective). Default value is `"opt.dat"`.

n_steps (scalar)

The number of steps for optimization (outermost level).

output_interval (scalar)
The interval between consecutive output records. A value of 1 records every step, a value of 2 records every other step, etc.

local_method (scalar)
Only used when opt_package is "scipy". The local optimization method. Accepted values are listed under the "method" parameter of the scipy.optimize.minimize documentation.

local_options_file (optional, scalar)
Only used when opt_package is "scipy". The local optimization options configuration filename. Format is consistent with the generic configuration file format specified in section 6. Only supports scalar parameters. Accepted values are consistent with the arguments listed under the specific SciPy local method documentation, with exception to the arguments maxiter and bounds, which are automatically specified by ClC_MKM. See section 7 for an example use case.

global_method (optional,scalar)
Only used when opt_package is "scipy". The global optimization method. Accepted values are listed under the Global Optimization section of SciPy's optimize documentation. Method "brute" is not supported.

global_options_file (optional, scalar)
Only used when opt_package is "scipy". The global optimization options configuration filename. Format is consistent with the generic configuration file format specified in section 6. Only supports scalar parameters. Accepted values are consistent with the arguments listed under the specific SciPy global method documentation, with exception to the arguments niter/maxiter and bounds, which are automatically specified by ClC_MKM. See section 7 for an example use case.

algorithm (scalar)
Only used when opt_package is "spotpy". Specifies the algorithm used for SpotPy optimization. See SpotPy's Algorithm Guide for a list of accepted values (lowercase abbreviations).

limp (scalar)
Only used when opt_package is not specified. Specifies the target lower bound for improvement in the objective for a single step.

uimp (scalar)

Only used when opt_package is not specified. Specifies the target upper bound for improvement in the objective for a single step.

max_limp_steps (scalar)
Only used when opt_package is not specified. Specifies the maximum number of steps in which the improvement is below limp before switching from steepest descent to conjugate gradient.

6.3 Residual configuration file

The residual configuration file details ion flow targets for optimization, which are combined using sum of squared residual differences. Accepted parameters are listed below. Note the length of <param_list> must be consistent with the number of systems specified in the main configuration file, following the <param_list> rules specified in section 6. If a value is specified as NaN, then the corrosponding residual flow is ommitted from the objective function calculation.

net_Cl_flow (optional)
The net chloride flow(s) directed from external to internal in ions/ms per enzyme.

net_H_flow (optional)
The net proton flow(s) directed from external to internal in ions/ms per enzyme.

bio_Cl_flow (optional)
The chloride flow(s) directed from external to internal in ions/ms per enzyme for biologically oriented enzymes.

bio_H_flow (optional)
The proton flow(s) directed from external to internal in ions/ms per enzyme for biologically oriented enzymes.

opp_Cl_flow (optional)
The chloride flow(s) directed from external to internal in ions/ms per enzyme for oppositely oriented enzymes.

opp_H_flow (optional)
The proton flow(s) directed from external to internal in ions/ms per enzyme for oppositely oriented enzymes.

no_flow_sys (optional)

Special boolean residual parameter (`"True"` or `"False"`). Designates a system to have no flow anywhere (microscopic reversibility), generally for use with zero gradient boundary conditions. See `ClC_MKM/examples/opt/custom/opt_residuals.txt` for a usage example.

7 Examples

See `ClC_MKM/examples` for some usage examples. `ClC_MKM/examples/mkm` provides a single instance of the ClC-ec1 kinetic system to be executed with `"run_mkm.py config.txt"`. All other examples are designed to be executed within their respective directories with `"run_opt.py config.txt"`. See below for commands that exhibit a traditional optimization/analysis workflow.

```
cd ClC_MKM/examples/opt/custom
run_opt.py config.txt
run_rpa.py config_rpa.txt
cd biological
parse_cycles.py -ci ../seed_dru_san.csv
cd ../opposite
parse_cycles.py -ci ../seed_dru_san.csv --opp
```

Executing the above commands should generate analysis files in subdirectories of the custom optimization. The same workflow should work for the other optimization examples provided. Note these examples are designed to execute quickly, and would require significantly more iterations to achieve complete optimization.

8 References

[1] Austen Bernardi and Jessica MJ Swanson. CycFlowDec: a python module for decomposing flow networks using simple cycles. *SoftwareX*, 14:100676, 2021.

[2] Tobias Houska, Philipp Kraft, Alejandro Chamorro-Chavez, and Lutz Breuer. SPOTting model parameters using a ready-made python package. *PloS one*, 10(12):e0145180, 2015.

[3] Hyun-Ho Lim and Christopher Miller. Intracellular proton-transfer mutants in a CLC Cl-/H+ exchanger. *Journal of General Physiology*, 133(2):131–138, 2009.

[4] Heather B Mayes, Sangyun Lee, Andrew D White, Gregory A Voth, and Jessica MJ Swanson. Multiscale kinetic modeling reveals an ensemble of Cl–/H+ exchange pathways in ClC-ec1 antiporter. *Journal of the American Chemical Society*, 140(5):1793–1804, 2018.

[5] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. SciPy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.