ClC_MKM v0.1 Manual

Austen Bernardi

October 29, 2021

Contents

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 [2]. ClC_MKM supports optimization of kinetic rate coefficients between biologically relevant transitions against experimentally derived unitary turnover rates [1]. This manual describes how to use ClC_MKM to perform optimization.

2 Requirements

ClC_MKM depends on the following requirements:

- Python3 (python3)
- SciPy (python package)
- SpotPy (python package)
- NumPy (python package)

3 Installation

Download the top directory `ClC_MKM` to a directory that will store the module. Update your `$PYTHONPATH` enviornoment variable to include the storage directory if it is not already included. Update your `$PATH` environment variable to include `ClC_MKM/bin`.

4 Executables

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

4.1 `run_opt.py`

`run_opt.py` `<config>`

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 5.

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

4.2 `run_mkm.py`

`run_mkm.py` `<config>`

Parameters
`<config>`: A configuration file containing parameters for the ClC-ec1 system. The format of this configuration file is specified in section 5.

Description
Generates an instance of the ClC-ec1 system specified by `<config>`. See section **??** for a usage example.

5 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 **??**. 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`.

5.1 Systems configuration file

This is the main configuration file that is used as the argument for both executables listed in section 4. See below for the full list of accepted parameters.

input_rate_file (scalar)
The input rate coefficient filename for the kinetic model. See examples **??** for formatting. Units of 1/ms.

input_rate_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 5.2 for details.

5.2 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 examples **??** 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 5.3 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 5. Only supports scalar parameters. Accepted values are listed 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 examples **??** 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 5. Only supports scalar parameters. Accepted values are listed 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 examples **??** 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.

5.3 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 5. 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.

6 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`".

7 References

[1] 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.

[2] 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.