modFossa Documentation

Release 0.1

Gareth Ferneyhough

August 01, 2013

CONTENTS

Contents:

CHAPTER

ONE

MODFOSSA PACKAGE

1.1 modFossa Package

1.2 experiment Module

addConcentration(*cpName*, ***args*)

Add a concentration value to the concentration protocol *cpName*.

Parameters

- **cpName** – name of concentration protocol to append the value to. It must be declared first using `concentrationProtocol()`.
- **args** – required arguments specifying the ligand and its concentration. See example below for more information.

  The following example adds a calcium concentration of 200 nM to *myConcentrations*

  ```
  addConcentration('myConcentrations', Ca=200E-9)
  ```

concentrationProtocol(*name*)

Declare a new concentration protocol with the given name.

Concentration values must be added using `addConcentration()`

experiment(*name*, *voltageProtocolName*, *concentrationProtocolName*)

Create an experiment using the given voltage and concentration protocols.

For each concentration in the concentration protocol, the voltage protocol will be run through. This is equivalent to using the method `experimentSweep()` to add each concentration value manually.

Individual experiment sweep results can be obtained and plotted using the generated name *name_ligand_concentration*. Additionally, the experiment module provides a dictionary, *_experimentSweepNames*, keyed on the experiment name. The values in the dictionary are lists of the experiment sweep names associated with a given experiment.

For example,:

```
concentrationProtocol('concentrations')
concentrationProtocolAddStage('concentrations', Ca=20E-9)
concentrationProtocolAddStage('concentrations', Ca=100E-9)
experiment('myExperiment', 'myVoltageProtocol', 'concentrations')
validate()
run()
```

will run *myVoltageProtocol* using a Ca concentration of 20 nM, followed by a Ca concentration of 100 nM. The two experiment sweeps will be named *myExperiment_Ca_20E-9*, and *myExperiment_Ca_100E-9*. These two names can be used to plot the experiment sweeps using the *single* plotting methods, such as `plotSingleIV()` and `plotSingleCurrents()`.

The *multiple* plotting methods will plot all the experiment sweeps from the given experiment on a single plot. For example, the following snippet will plot the two individual IV curves from the experiment defined above on a single plot:

```
plotMultipleIV(myExperiment, time=1100)
```

isValid()
Return the validity of the model.

This will return *true* after a successful call to `validate()`.

run()
Run all experiments defined with `experiment()`.

When this method completes, the results will be available using the results module.

startAtSteadyState(*value*)
Tell the simulator whether or not to start the simulation at steady state.

This is *true* by default. If set to *false*, the method `initialState()` in the markovModel module can be used to set the initial state at which to begin the simulation.

validate()
Check the validity of the model and notify the user of any problems.

voltageProtocol(*name*)
Declare a new voltage protocol with the given name.

Voltage protocol stages must be added using `voltageProtocolAddStage()`

voltageProtocolAddStage(*vpName*, *stageName*, ***args*)
Add a voltage protocol stage voltage protocol protocol *vpName*. The stage can either be stepped, or constant.

Parameters

- **vpName** – name of voltage protocol to append the value to. It must be declared first using `voltageProtocol()`.
- **args** – required arguments specifying the voltage stage. Units are in milliVolts and milliSeconds. See example below for more information.

  The following example adds a stepped voltage protocol stage with the name *step* to *vp* ::
  voltageProtocolAddStage('vp', 'step', start=-100, stop=140, step=20, duration=1000)

  The following example adds a constant voltage protocol stage with the name *hold* to *vp* ::
  voltageProtocolAddStage('vp', 'hold', voltage=-50, duration=100)

1.3 `markovModel` Module

connect(*fromState*, *toState*, *rate*)
Connect two states with the given rate constant.

initialState(*name*)
Set the initial state.

For this to have any effect, `startAtSteadyState()` must be *false*.

maxChannelConductance(*conductance*)
Set the max channel conductance in nS.

membraneCapacitance(*capacitance_pf*)
Set the membrane capacitance in pF.

This command currently has no effect on the simulation and will be removed.

rate(*name*, ***args*)
Create a rate constant.

Parameters

- **name** – name of the rate constant. Must be unique.
- **args** – required arguments specifying the type and parameters of the rate constant.

The type of rate constant to create is specified by the *type* argument. This is required. The various rate constant types and their required arguments are listed below.

constant

A rate constant with with a single rate, not dependent on voltage or ligand concentration.

Required arguments:

- k

Implementation: $rate = k$.

Example:

```
rate('myConstantRate', type='constant', k=10)
```

boltzman

A voltage dependent rate constant using the Boltzman equation.

Required arguments:

- a
- v_half
- k

Implementation: $rate(V) = \frac{a}{1+exp[(V-V\_0.5)/k]}$.

Example:

```
rate('myBoltzmanRate', type='boltzman', a=1, v_half=-30, k=10)
```

exponential

An exponential voltage dependent rate constant

Required arguments:

- a
- k

Implementation: $rate(V) = a * exp(k * V)$.

Example:

```
rate('myExponentialRate', type='exponential', a=1, k=10)
```

ligandGated

A ligand gated rate constant

Required arguments:

- ligand
- ligand_power
- k

Implementation: $rate([ligand]) = k * [ligand]^{ligand\_power}$ where $[ligand]$ is the concentration of *ligand*.

Example:

```
rate('myLigandGatedRate', type='ligandGated', ligand='Ca', ligand_power=3, k=50)
```

reversalPotential (*reversalPotential*)
Set the reversal potential in mV.

state (*name*, *conducting=False*, *gating=1.0*)
Add a new state to the markov model.

Parameters

- **name** – name of the state. Must be unique.
- **conducting** – whether or not the state is a conducting state. False by default.
- **gating** – scales the conductance of the state. Only used *conducting* is True. Default value is 1.0.

1.4 `plotting` Module

plotCurrents (*experimentSweepName*)
Plot the current traces for a given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

plotGvsConcentration (*experimentName*, *time_ms*)
Plot the channel conductance versus concentration for a given experiment sweep at the specified time in milliSeconds.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

plotGvsV (*experimentName*, *time_ms*)
Plot the channel conductance versus membrane voltage for a given experiment sweep at the specified time in milliSeconds.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

plotIV (*experimentSweepName*, *time_ms*)
Plot the IV curve for a given experiment sweep at the specifed time in milliSeconds.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

plotMultipleCurrents(*experimentName*)
Plot all of the current traces for a given experiment.

A figure will be generated with several subplots, one for each experiment sweep in the experiment.

plotMultipleIV(*experimentName*, *time_ms*, *ymin*, *ymax*, *labelHeight*)
Plot all of the IV curves for a given experiment at the given time in millieSeconds.

A figure will be generated with several subplots, one for each experiment sweep in the experiment.

plotStates(*experimentSweepName*)
Plot the state probablities for a given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

plotVoltageProtocol(*experimentSweepName*)
Plot the voltage protocol for a given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

1.5 `results` Module

getConductances(*experimentSweepName*)
Get the conductances for the given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

getCurrents(*experimentSweepName*)
Get the currents for the given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

getIV(*experimentSweepName*, *time_ms*)
Get the IV curve for the given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

getStateNames()
Get the state names for the given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

getStateProbabilities(*experimentSweepName*)
Get the state probabilities for the given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

getVoltageProtocol(*experimentSweepName*)
Get the voltage protocol for the given experiment sweep.

See `experiment()` in the experiment module on how experiment sweep names are generated for a given experiment.

INDEX