Systems Biology Workbench - Simulation API

Herbert Sauro and Frank Bergmann

{hsauro,fbergmann}@kgi.edu
Systems Biology Workbench Development Group
ERATO Kitano Systems Biology Project
Keck Graduate Institute,
535 Watson Drive, Claremont, CA 91171, USA

http://www.sys-bio.org

November 18, 2005

Contents

1 Introduction **3**
1.1 Plugin Category 3
2 Simulator API - Level 1 **4**
3 Simulator API - Level 2 **5**
4 Simulator API - Level 3 **8**
5 Simulator API - Level 4 **9**
6 Simulator API - Level 5 **11**
7 Simulator API - Level 6 **11**

1 Introduction

This document describes the SBW simulator standard API. This API supersedes the original Simulator API published by Finney et. al. in 2002.

The new simulator API will be described in levels, each level becoming more comprehensive. Thus the lowest level will be the simplest and will provide only the most basic facilities for supporting a simulation API.

All levels above level 1 (the lowest level) must implement methods in earlier levels.

To specify levels to SBW, please use the category feature of SBW. To support level 1, register you module with category:

`plugin/simulator/level1`

For level 2 use:

`plugin/simulator/level2`

and so on.

1.1 Plugin Category

The simulator API descends from the Plugin category API. This API is described here. The purpose of the plugin API is to provide basic information about the underlying module, information such as author, version, etc.

string getName()

Returns the name of the module as a string

string getVersion()

Returns the version string for the module

string getAuthor()

Returns the name of the author of this module

string getDescription()

Returns a description of the module

string getDisplayName()

Returns a string which could be used to populate a menu, list box etc.

string getCopyright()

Returns the copyright and licence type for this module

string getURL()

Returns the URL where further information can be obtained

2 Simulator API - Level 1

This section describes Level 1 for the simulator API.

void loadSBML(string)

Loads the SBML string into the simulator. The method throws an exception in the event of an error

string getSBML()

Returns the currently loaded model as SBML, returns empty if no model loaded

void setTimeStart (double)

Sets the time start for the simulation, defaults to 0.0 if not set

void setTimeEnd (double)

Sets the time end for the simulation, defaults to 25.0 time units if not set

void setNumPoints (int)

Sets the number of data points to generate during the simulation, defaults to 50 points if not set

void reset()

Resets the simulator to the initial conditions (initial concentrations) specified in the SBML model.

double[][] simulate ()

Simulates the model as given in the loadSBML() method, Returns a 2D array containing time in the first column and species concentrations in the remaining columns. Note that at the end of a run, the values of the state variables are set to the final time point of the simulation. This means that a second call will commence the simulation from the last point. To start the simulation from the original initial conditions, call the reset() method.

double[][] simulateEx (double timeStart, double timeEnd, int numPoints)

Simulates the model as given in the loadSBML() method, allows the caller to specify directly the time start, time end and number of points to generate, otherwise acts the same as the simulate() method. Note that at the end of a run, the values of the state variables are set to the final time point of the simulation. In addition, simulateEx() makes a call to reset() before carrying out the simulation (this is in contrast to simulate() which will continue a simulation at the current values of the species.

void setFloatingSpeciesConcentrations (double[] values)

Sets the floating species values to the array vector given in the argument. The number of values in the array must equal the number of species in the model.

double[] getReactionRates ()

Returns a double array containing the values of the reaction rates for the currently loaded model. Reaction rates are computed at the current values for the floating species.

double[] getRatesOfChange ()

Returns a double array containing the values for the rates of change for each species in the model. Values are computed at the current values for the floating species.

{} **getFloatingSpeciesNames ()**

Returns a list containing the names of all the floating species. getSpeciesNames() has been deprecated.

{} **getSpeciesNames ()**

Returns a list containing the names of all the floating species. This method has been deprecated, please use instead getFloatingSpeciesNames().

{} **getReactionNames ()**

Returns a list containing the names of all the reactions in the model.

2.0.1 Example

```
loadSBML (SBMLStr);
m = simulateEx (0.0, 10, 100);
plot (m);
```

3 Simulator API - Level 2

int getNumberOfCompartments()

Returns the number of compartments in the model.

double getCompartmentByIndex (int index)

Gets the value of a parameter by its index value.

void setCompartmentByIndex (int index, double value)

Sets the volume of a compartment by its index in the list returned by getCompartment-Names()

void setCompartmentVolumes (double[] values)

Sets the volumes for all compartments using the vector values

{} **getCompartmentNames ()**

Returns a list of compartment names.

int getNumberOfBoundarySpecies()

Returns the number of boundary species in the model.

double getBoundarySpeciesByIndex (int index)

Gets the concentration of a boundary species by its index value.

void setBoundarySpeciesByIndex (int index, double value)

Sets the concentration of the floating species by its index in the list returned by getBoundarySpeciesNames()

double[] getBoundarySpeciesConcentrations ()

Returns the an array of boundary species concentrations

void setBoundarySpeciesConcentrations (double[] values)

Sets the concentrations for all the boundary species.

{} **getBoundarySpeciesNames ()**

Returns a list of boundary species names.

int getNumberOfFloatingSpecies()

Returns the number of floating species in the model.

double getFloatingSpeciesByIndex (int index)

Gets the concentration of a floating species by its index value.

void setFloatingSpeciesByIndex (int index, double value)

Sets the concentration of the floating species by its index in the list returned by getFloatingSpeciesNames()

double[] getFloatingSpeciesConcentrations ()

Returns the an array of floating species concentrations

int getNumberOfGlobalParameters()

Returns the number of global parameters in the model.

void setGlobalParameterByIndex (int index, double value)

Sets the value of a parameter by its index in the list returned by getParameterNames()

double getGlobalParameterByIndex (int index)

Gets the value of a global parameter by its index)

void setGlobalParameterValues (double[] values)

Set the values for all global parameters

double[] getGlobalParameterValues ()

Returns a list of global parameter values

{} **getGlobalParameterNames ()**

Returns a list of global parameter names in the model.

{} **getAllGlobalParameterTupleList ()**

Returns a list of global parameters as a list of 2-tuples. Each tuple contains the name of the global parameter and it's value, eg $\{\{\text{“}k_1\text{”}, 1\}, \{\text{“}k_2\text{”}, 1\}, \{\text{“}k_3\text{”}, 1\}\}$

int getNumberOfLocalParameters(int reactionId)

Returns the number of local parameter in a given reaction.

void setLocalParameterByIndex (int reactionId, int index, double value)

Sets the value of a parameter by its index in the list returned by getParameterNames()

double getLocalParameterByIndex (int reactionId, int index)

Gets the value of a local parameter by its index and reaction Id

void setLocalParameterValues (int reactionId, double[] values)

Set the values for all local parameters in a particular reaction

double[] getLocalParameterValues (int reactionid)

Returns the values for all local parameters in a particular reaction

{} **getLocalParameterNames (int reactionId)**

Returns a list of local parameter names for a particular reaction.

{} **getAllLocalParameterTupleList ()**

Returns a list of local parameters as a list of 3-tuples. Each tuple contains the reaction number, the name of the local parameter and it's value, eg $\{\{0, \text{“}k_1\text{”}, 1\}, \{1, \text{“}k_2\text{”}, 1\}, \{2, \text{“}k_3\text{”}, 1\}\}$

int getNumberOfReactions()

Return the number of reactions in the model.

double getReactionRate (int index)

Returns the reaction rate by its index

double[] getReactionRatesEx (double[] speciesArray)

Returns the reaction rates given an array of floating species concentrations

double getRateOfChange (int index)

Returns the rate of change by its index

double[] getRatesOfChangeEx (double[] speciesArray)

Returns the rates of change given an array of floating species concentrations

{} **getRateOfChangeNames ()**

Returns the names given to the rate of change of the floating species

void setSelectionList ()

Sets the selection list for the simulation. The selection list contains the names of the quantities that should be returned in the array after a call to simulate. Valid names include names of species and reaction names. Reaction names return the reaction rate for the specified reaction. Rates of change can be returned by appending an apostrophe to the end of a species name, eg ATP', p53'. The special symbol name, time, is interpreted as the independent variable time.

double oneStep(double currentTime, double stepSize)

Compute one integration step over the interval stepSize. Returns the value of the new time (usually currentTime + stepSize)

3.0.2 Example

```
loadSBML (SBMLStr);
setSelectionLists ({'Time', 'S1', 'S2', 'J1'})
m = simulateEx (0.0, 10, 100);
plot (m);
```

4 Simulator API - Level 3

double[][] getReducedJacobian ()

Returns the Jacobian matrix for the current state of the model. The Jacobian should be the reduced Jacobian, that is the Jacobian of model where the conserved cycles have been removed.

double[][] getFullJacobian ()

Returns the Jacobian matrix for the current state of the model. The Jacobian should be the full Jacobian.

double steadyState()

Attempts to compute the nearest steady state using the current model concentrations as a starting point. The function returns a value indicating how close the solution was reached. Usually the returned value is computed from $\sum(dS_i/dt)^2$.

5 Simulator API - Level 4

double[][] getStoichiometryMatrix()

Returns the Reordered Stoichiometry matrix for the currently loaded model.

double[][] getLMatrix()

Returns the Reder Link Matrix for the currently loaded model.

double[][] getNrMatrix()

Returns the Reduced Stoichiometry matrix for the currently loaded model.

double[][] getL0Matrix()

Returns the Lo portion of the link matrix for the currently loaded model.

double[][] getConservationMatrix()

Returns the conservation matrix, each row corresponds to a single conservation law.

double[][] getScaledElasticityMatrix()

Returns the matrix of scaled pathway elasticity coefficients.

double[][] getUnscaledElasticityMatrix()

Returns the matrix of unscaled pathway elasticity coefficients.

public double getUnscaledFloatingSpeciesElasticity (string reactionName, string speciesName)

Returns the unscaled species elasticity for a given reaction and given floating species.

public double getScaledFloatingSpeciesElasticity (string reactionName, string speciesName)

Returns the scaled species elasticity for a given reaction and given floating species.

public double getUnscaledConcentrationControlCoefficient (string speciesName, string localReactionName, string parameterName)

Returns the unscaled concentration control coefficient for a local parameter

public double getScaledConcentrationControlCoefficient (string speciesName, string localReactionName, string parameterName)

Returns the scaled concentration control coefficient for a local parameter

public double getUnscaledConcentrationControlCoefficient (string speciesName, string parameterName)

Returns the unscaled concentration control coefficient for a global parameter, a boundary species or conservation total.

public double getScaledConcentrationControlCoefficient (string speciesName, string parameterName)

Returns the scaled concentration control coefficient for a global parameter, a boundary species or conservation total.

public double getUnscaledFluxControlCoefficient (string fluxName, string localReactionName, string parameterName)

Returns the unscaled flux control coefficient for a local parameter

public double getUnscaledFluxControlCoefficient (string reactionName, string parameterName)

Returns the unscaled flux control coefficient for a global parameter, a boundary species or conservation total.

public double getScaledFluxControlCoefficient (string reactionName, string localReactionName, string parameterName)

Returns the unscaled flux control coefficient for a global parameter, a boundary species or conservation total.

public double getScaledFluxControlCoefficient (string reactionName, string parameterName)

Returns the scaled flux control coefficient for a global parameter, a boundary species or conservation total.

double[][] getUnScaledConcentrationControlCoefficientMatrix()

Returns a matrix of unscaled concentration control coefficients.

double[][] getUnScaledFluxControlCoefficientMatrix()

Returns a matrix of unscaled flux control coefficients.

double[][] getScaledConcentrationControlCoefficientMatrix()

Returns a matrix of scaled concentration control coefficients.

double[][] getScaledFluxControlCoefficientMatrix()

Returns a matrix of scaled flux control coefficients.

double getUnscaledParameterElasticity (string reactionName, string parameterName)

Returns the parameter elasticity for a given reaction with respect to a given parameter. Allowed parameters include global parameters, local parameters, boundary species and conservation totals.

double[][] getFrequencyResponse(double startFrequency, int numberOfDecades, int numberOfPoints, string parameterName, string variableName, bool useDB, bool useHz)

Returns the frequency response of a given variable with respect to a given parameter. The boolean options useDB and useHz when set to true indicate that the results should be returned using Decibels and Hertz respectively. The response is returned in three columns, the first column indicates the frequency, the second column the amplitude and the third column the phase of the response.

6 Simulator API - Level 5

Things like time dependent MCA

7 Simulator API - Level 6

Parameter Optimization?