# BioSimWare

Paolo Cazzaniga

July 6, 2010

BioSimWare is a novel software that provides a user-friendly framework for stochastic modelling, simulation and analysis of complex biological systems. It is based on a simple modelling approach, which consists in describing the system under investigation through "reagents → products" reactions, describing the interactions between molecules. Stochastic simulation of the emergent dynamics of the system can be easily run by setting the initial conditions, such as amounts of molecular species and kinetic constants, and by modulating the reaction parameters to test different conditions.

BioSimWare also implements optimization techniques to perform the estimation of unknown parameters, which can be performed by providing an experimental/synthetic time-series curve of the dynamics of one or more molecular species. Statistical analysis and characterization of the dynamical properties can finally be performed.

## System requirements

The system requirements needed to run BioSimWare are:

- Linux, Windows or Mac OS system
- Java 5.0 or a higher version
- MPI (to run parallel simulations)

The user interface is written in Java language, the executable files of the simulation algorithms (written in C language) can be found on the web page.

The executable of the simulation algorithms can be inserted into the "bin" folder. Alternatively, in the "Simulation → Settings" menu, it is possible to specify a different folder.

Note that the current BioSimWare version can be used to implement only single volume models. It is possible to write multi-volume models and generate the correct input files by using the parser (written in perl language) available on the web page.

## Single volume input files

To run the simulator, a number of files, contained inside an `input` folder, is needed. These files, generated by using the java inteface, are used to describe the implemented model and the parameters of the algorithm:

- `left_side`: contains a matrix which describes the left-hand side of the reactions
- `right_side`: contains a matrix which describes the right-hand side of the reactions
- `c_vector`: contains the reactions constants
- `M_0`: contains the initial quantities for the molecular amounts
- `M_feed`: contains the values of the molecular species whose quantities are kept constant during the simulation (the values are otherwise left to 0)
- `indexes`: contains the indexes of the molecular species whose dynamics is printed to the output file
- `eps`: contains the value of the error control parameter, and it is usually set to 0.03 (this value is used only by the tau-leaping based algorithms)
- `every`: contains the sampling frequency and the buffer dimension
- `time_max`: contains the maximum simulation time

To run n tau-leaping simulation (with the executable `tau-leaping-mean-dynamics`) and compute the average dynamics, an additional file, called "data" is required. This file contains the parameters of the algorithm and has to be added to the input folder.

The file must contain two rows:
(1) the number of tau-leaping executions
(2) the sampling frequency of the average dynamics.

An example is reported here:

```
Runs 10
Samplings 1000
EOF
```

# Multi volume parser

The description of multi-volume systems is achieved by creating different input files for each volume. These files are labelled with the volume id $(1, \ldots, N)$; for instance, we can define the files `left_side_0` and `right_side_0` for volume 0.

On BioSimWare web page, a parser written in perl language, can be downloaded and used to parse a text document and automatically create the input files for a multi-volume system.

The model file is structured as follows. First, the description of the volumes has to be indicated: the list of reactions written in the form $reagent_1 + reagent_2 + \cdots + reagent_n = product_1 + product_2 + \cdots + product_n$, along with the constant and the target (in the case of communication reactions). Then, the initial amount for the chemical species is specified only for those species whose value is greater than zero. In addition, the output species and the chemicals whose quantity has to be kept fixed throughout the simulation have to be indicated.

After the description of the reaction volumes, the algorithm parameters are listed. These parameters are the same as in the case of single volume algorithms.

An example of model file is the following:

```
Volume 0

#Reaction Constant Target
Reactions
a=b 0.1 1

Initial Conditions
+a= 1000
+b

Volume 1

Reactions

b=c 0.025 0


Initial Conditions
+c

Algorithm Parameters

time=100
eps=0.03
every=1
buffer=100
```

Note that comment lines begin with #. In the list of reactions, in the case of communication reactions the target volume has to be indicated: n=products sent to volume n. Otherwise, one can specify a different targets for different products. For instance, with the rule a=b+c, if b and c go to different targets, then the reaction will be indicated as follows:
`a=b+c 0.1 b=2 c=3`,
so doing, b is sent to volume 2 and c to volume 3

In the list of initial conditions, + is added before the species name to indicate those species whose dynamics is printed to the output files. If no species is tagged, then all species are saved. The symbol * after a species value is used to indicate the species whose quantity has to be kept fixed throughout the simulation.

## Parameter optimization input files

In order to perform a parameter estimation of a single-volume system, three additional files are required. The first one is the target dynamics (file `target`),

composed of $n+1$ columns: the time and the target dynamics of the $n$ chemical species.

The second file contains the parameters specification of the algorithm used for the optimization: Genetic Algorithms or Particle Swarm Optimizer.

For what concerns the application of Genetic Algorithms, the file `GAdata`, which contains the algorithm parameters specification, is added to the input folder. This file contains the following information:

- PopDim: the number of individuals in the population
- IndLength: the length of the individuals, i.e., the number of parameters to optimize
- Tourn: the size of the tournament, that is, the method used to select individuals
- Elite: the number of best individuals copied to the next generation without being modified by means of crossover and mutation
- MaxGen: maximum number of generations
- Exec: the number of execution of the tau-leaping algorithm to compute the fitness value
- Pcross: crossover probability (value taken from the interval [0,1])
- CrossType: crossover type, it can be set to Mean (Default), Exchange, Laplace or NonDet
- Pmut: mutation probability (value taken from the interval [0,1])
- MutType: mutation type, it can be set to Range (Default), Gauss, GaussW, Reinit or NonDet
- Eps: mutation strength

The different crossover and mutation types are briefly described hereafter:

- Mean crossover: the values of the offspring are computed as the average of the values of the parent individuals
- Exchange crossover: the values of the parent individuals are exchanged and two offspring are generated
- Laplace crossover: select new offspring by using the Laplace distribution
- NonDet crossover: randomly select a crossover operator
- Range mutation: modify the selected value as value +/- eps * variation range (of the value)
- Gauss mutation: modify the selected value by adding a random number selected from a gaussian distribution with mean=value and sigma= 0.5 * eps * variation range

- GaussW mutation (weak gaussian mutation): modify the selected value by adding a random number selected from a gaussian distribution with mean=value and sigma= 0.05 * eps * variation range
- Reinit: reinitialize (randomly) the selected value
- NonDet: randomly select a mutation operator

An example of GAdata is reported here:

```
PopDim 100
IndLength 3
Tourn 5
Elite 1
MaxGen 100
Exec 5
Pcross 0.95
CrossType NonDet
Pmut 0.05
MutType NonDet
Eps 0.1
EOF
```

For the application of Particle swarm optimizers, the file `PSOdata`, which contains the algorithm parameters specification, is added to the input folder. This file contains the following information:

- SwarmDim: the number of particles of the swarm
- ParticleDim: the number of dimensions of the particles, i.e., the number of parameters to optimize
- VelocityDim: a value in (0,1] which represents the fraction of the position range that will be considered as the velocity range (value taken from the interval (0,1])
- C1: the constant related to the local search (value taken from the interval (0,4])
- C2: the constant related to the global search (value taken from the interval (0,4])
- Coevolution: coevolution of the C's values (flag = 0 or 1): add gaussian noise to the initial values
- SigmaCoev: sigma value of the gaussian distribution used in the coevolution (value $> 0$)
- InitialInertia: initial value of the inertia used to damp the velocity value (value from the interval (0,1])
- FinalInertia: final value of the inertia (value from the interval (0,1])

- SigmaInertia: sigma value of the gaussian distribution used to add noise to the inertia value (value $> 0$)
- Iterations: maximum number of iterations
- Exec: the number of execution of the tau-leaping algorithm to compute the fitness value

An example of PSOdata is reported here:

```
SwarmDim 100
ParticleDim 3
VelocityDim 0.5
C1 2
C2 2
Coevolution 1
SigmaCoev 0.1
InitialInertia 0.9
FinalInertia 0.4
SigmaInertia 0.1
Iterations 100
Exec 5
EOF
```

The last file needed to run a parameter estimation is called `ind_ranges`, it is composed by $m$ rows, where $m$ is the number of constants to optimize. Each row has two floating point values: the lower-bound and the upper bound of the variation range of the corresponding constant of the chemical system under investigation.

An example of ind_ranges is reported here:

```
1 100
5 15
0.3 0.9
```