DelaySim

A Java library to simulate stochastic systems with delays

Giulio Caravagna
`caravagn@di.unipi.it`

Dipartimento di Informatica, Università di Pisa
Largo Bruno Pontecorvo 3, 56127 Pisa, Italy

January 10, 2011

Abstract

`DelaySim` is a library, written in Java, allowing easy coding of Java reaction-based models of biological systems with delays. Such models can be simulated by two different Delay Stochastic Simulation Algorithms: the DDA and the PDA. `DelaySim` is released under the terms of the GNU GPL license.

Preamble

Along the line of other simulation tools, `DelaySim` is a library which provides the programmer the required data structures and methods to easily describe and simulate reaction-based models of biological systems with delays.

`DelaySim` has been used to perform stochastic simulations of such systems, as discussed in the Ph.D. thesis

"Formal Modeling and Simulation of Biological Systems With Delays"

by Giulio Caravagna. Nomenclature and notation used in the following are taken directly from such a thesis, which can be found at

`http://www.di.unipi.it/~caravagn/`

or can be directly asked to the author. Accordingly to this, the Delay Stochastic Simulation Algorithms algorithms implemented in the library are the DDA and the PDA, as discussed in Chapters 4 and 5, respectively. In the former reactions with delay follow the delay-as-duration approach whereas in the latter follow a purely delayed approach.

`DelaySim` is written in Java 1.6 and is released under the terms of the GNU GPL license.

Packages

`DelaySim` is a pretty simple library consisting of only 2 packages: a package *DelaySim* containing the core of the library, and package *DelaySim.examples* containing some programming examples.

The main package *DelaySim* contains the definition of the following Java types: *Reaction*, *Entry*, *DDA*, and *PDA*.

- *Reaction* is a class representing a generic reaction happening in the modeled system. The reaction is represented as a 5-tuple

  $$(s, \nu_r, \nu_p, k, \sigma)$$

  where

  - s is the *name* of the reaction;
  - $\nu_r \in \mathbb{N}^n$ is the *stoichiometry vector for reactants*;
  - $\nu_p \in \mathbb{N}^n$ is the *stoichiometry vector for products*;
  - $k \in \mathbb{R}$ with $k > 0$ is the *kinetic constant* of the reaction;
  - $\sigma \in \mathbb{R}$ with $\sigma \geq 0$ is the *delay* of the reaction.

  Notice that ν_r must contain *negative* entries since it represents only consumed reactants.

- *DDA* is a class representing a generic algebraic reaction-based system which can be simulated by the DDA. Such a system is represented as a pair

  $$(\mathbf{R}, \mathbf{x})$$

  where

  - $\mathbf{R}$ is a *set of reactions*

    $$\mathbf{R} = \{R_1, \ldots, R_m\}$$

    which can happen in the system, these objects are typed *Reaction*;
  - $\mathbf{x} \in \mathbb{N}^n$ is the state-vector representing initial state of the system to be simulated.

  Notice that the *consistency* of the system as defined by the size of the state-vector and the vectors used by the reaction is to be checked by the programmer. On consistent systems DDA stochastic simulation can be performed starting from an initial time $t_0 \in \mathbb{R}$ up to a maximum time $T \in \mathbb{R}$, which can be chosen by the programmer.

- *PDA* is a class representing a generic reaction-based system which can be simulated by the PDA. This class is defined similarly to the DDA one.

- *Entry* is a class describing a generic entry in the scheduling lists of both the algorithms. This class should be of no interest for the library user.

Detailed information about the methods and variables used in the classes can be found in the attached Javadoc documentation.

Examples

`DelaySim` comes with two three examples of programmed models to clarify the use of the library. The first two examples are single reaction and reversible-reaction models, respectively. The third example is the model of the cell cycle with a delay presented in Chapters 4 and 5. Examples are contained in the package *DelaySim.examples*.

Example I [SingleReactionExample.java]

In this model we consider a population of two species A and B, hence the system state is described by a two-dimensional vector. We consider a single reaction

$$R_1 : \; A \xmapsto{k,\sigma} B$$

with kinetic constant k and delay σ. Default values are $k = 10.0$ and $\sigma = 3.0$, we write the following code

```
int [] v1_R = {-1, 0};
int [] v1_P = {0, 1};
Reaction r1 = new Reaction("R1", v1_R, v1_P, 10.0, 3.0);
```

The state is coded as

```
int [] x0 = {INITAL_A_MOLECULES, 0};
```

where `INITAL_A_MOLECULES` is a variable whose value is the initial number of molecules A in the system. Such a state represents a two-dimensional vector

$$\mathbf{X}(t_0) = \texttt{x0}.$$

with $\mathbf{x}_0 =$`x0`. Initial time t_0 can be specified once a simulation is started.

A whole DDA system can be coded as

```
Reaction [] reactions = {r1};
DDA DDA_system = new DDA(reactions, x0);
```

and it can be simulated from $t_0 = 0$ to $T = 10$ by executing method `simulate` from class *DDA*

```
DDA_system.simulate(0, 10);
```

With a very similar coding style the same system can be simulated by the PDA.

```
PDA PDA_system = new PDA(reactions, x0);
PDA_system.simulate(0, 10);
```

When executed such a method the output is of the form

```
1.3517503200690113              12              4
1.5342960570253148              12              4
1.6365428550462302              11              4
1.6644229295208675              10              4
1.6860151782990693              9               4
....
```

where the first column represents the system clock, and column i-th represent the value of the i-th component of the state vector.

Example II [TwoReactionsExample.java]
This is the same of Example I where the reaction is reversible

$$R_1 :\ A \xmapsto{10.0,3.0} B \qquad\qquad R_2 :\ A \xmapsto{1.0,5.0} B$$

Reactions are coded as

```
int [] v1_R = {-1, 0};
int [] v1_P = {0, 1};
Reaction r1 = new Reaction("R1", v1_R, v1_P, 10.0, 3.0);

int [] v2_R = {0, -1};
int [] v2_P = {1, 0};
Reaction r2 = new Reaction("R2", v2_R, v2_P, 1.0, 5.0);
```

and systems as

```
int [] x0 = {INITAL_A_MOLECULES, INITAL_B_MOLECULES};
Reaction [] reactions = {r1, r2};

DDA DDA_system = new DDA(reactions, x0);
PDA PDA_system = new PDA(reactions, x0);
```

so that systems can be simulated as in Example I.

Example III [CellCycleModel.java]
This is the model of the cell cycle with a delay in the passage of a cell from the interphase to the mitotic phase. Such a model is discussed in Chapters 3, 4 and 5, where both DDA and PDA simulations of the model are presented.