



User's Guide



SBMLsimulator

An efficient Java™ solver implementation for SBML

Alexander Dörr¹

Roland Keller¹

Andreas Zell¹

Andreas Dräger^{1,2,*}

July 16, 2014

Institutional affiliations:

¹Center for Bioinformatics Tuebingen (ZBIT), University of Tuebingen, Tübingen, Germany

²Systems Biology Research Group, University of California, San Diego, La Jolla, CA, USA

*Corresponding author: andraeger@eng.ucsd.edu

SBMLsimulator is a fast, accurate, and easily usable program for dynamic simulation and heuristic parameter optimization of models encoded in the Systems Biology Markup Language (SBML). In order to ensure a high reliability of this software, it has been benchmarked against the entire SBML Test Suite and all models from the BioModels database. It includes the large collection of nature-inspired heuristic optimization procedures for efficient model calibration from the framework EvA2. SBMLsimulator provides an intuitive Graphical User Interface (GUI) and several command-line options to be suitable for large-scale batch processing and model calibration. SBMLsimulator runs on all platforms that provide a standard Java Virtual Machine and is based on the open-source library JSBML. The simulation core library of SBMLsimulator can be obtained as a separate application programming library.

Contents

1	Introduction	1
2	Installation	2
2.1	Requirements	2
2.2	Starting the application	2
3	How to get started	4
3.1	Open a model file	4
3.2	Simulate a model	5
3.3	Explore the tree structure of the model	8
3.4	Save plot image	9
3.5	Save simulation data	10
3.6	Save model	10
3.7	Prepare experimental data	11
3.8	Open experimental data in SBMLsimulator	12
3.9	Parameter estimation	14
4	Command-line arguments and preferences	20
4.1	Simulator input/output options	20
4.2	Simulation options	20
4.3	Estimation Options	22
4.4	Options for the graphical user interface	24
4.5	Plot Options	25
4.6	Character-Separated Value (CSV) options	25
4.7	Garuda options	26
5	License	27
5.1	Included third-party libraries and packages	27
6	FAQ / Troubleshooting	28
A	Acknowledgments	30

B Release notes	31
B.1 Version 1.0	31
B.2 Version 1.1	31
B.3 Version 1.2	31
B.4 Version 1.2.1	32
C Acronyms	33
Bibliography	34
Index	36

Figures

3.1 Loading of a model	5
3.2 SBMLsimulator with a loaded model	6
3.3 Selection of quantities for plotting	7
3.4 Starting of a simulation in SBMLsimulator	9
3.5 Simulation results plotted in SBMLsimulator	10
3.6 Display of model tree	11
3.7 Table with simulation data	12
3.8 Example of the data import dialog	13
3.9 Expanded “CSV Options” allow to give details for the input file	13
3.10 SBMLsimulator after simulation and loading of experimental data	14
3.11 Selection of quantities to estimate	16
3.12 EvA2 settings window	17
3.13 Changing the optimization procedure	18
3.14 Intermediate estimation result	19

Tables

3.1 Solvers provided by SBMLsimulator	7
3.1 Solvers provided by SBMLsimulator (continued)	8

Listings

3.1 Input file example for experimental data	12
3.2 Input file example for parameter estimation file	15

Contents

1 Introduction

SBMLsimulator is a easily usable, portable, and powerful simulation and parameter optimization engine for biochemical network models in Systems Biology Markup Language (SBML) format (Hucka *et al.*, 2003). It uses the JSBML-based Systems Biology Simulation Core Library (SBSCL) (Dräger *et al.*, 2011; Keller *et al.*, 2013) as its computational core and combines its functions with EvA2 (Kronfeld *et al.*, 2010), a Java™ framework for nature-inspired heuristic optimization procedures. In this context, optimization means the estimation of model parameters, i.e., the calibration of a model with respect to given experimental data (Dräger *et al.*, 2009).

This document is intended to guide you through the function and use of SBMLsimulator. We will start by explaining how to install and launch the program, and how to use its more advanced features.

1.0.1 Main program features

SBMLsimulator has been designed to

- ✓ parse SBML files and display its content in a Graphical User Interface (GUI) for exploration.
- ✓ insert missing values into the model that are required to perform a dynamic simulation.
- ✓ interpret metabolic, gene regulatory, and signal transduction models in terms of an Ordinary Differential Equation (ODE) systems and solve those with numerical integration methods.
- ✓ fit the model to given experimental data and save the optimization results in the model.
- ✓ plot experimental and simulation data, i.e., temporal changes of all model components.
- ✓ export simulation results as CSV or image file.
- ✓ save your preferences and restore all your settings when launched next time.
- ✓ be launched as a Garuda gadget from the Garuda dashboard (Ghosh *et al.*, 2011).
- ✓ be used as a Java™ Web Start application without any local installation of software, directly from the website in your browser or by clicking at this link: [SBMLsimulator-latest.jnlp](#).
- ✓ be used in a command-line mode without Graphical User Interface (GUI).
- ✓ be embedded as an Application Programming Interface (API) in a third-party program, allowing to access all of its functions in more complex scripts and procedures.

2 Installation

SBMLsimulator comes as a Java™ Archive (JAR) file. You can download it from <http://www.cogsys.cs.uni-tuebingen.de/software/SBMLsimulator/>. It can run out-of-the-box on all systems where a Java™ Virtual Machine (JVM) is installed and does not require any further installations.

2.1 Requirements

2.1.1 Software

SBMLsimulator is entirely written in Java™ and runs on any operating system where a suitable Java™ Virtual Machine (JVM) (Java™ Development Kit (JDK) version 1.6 or newer) is installed. It has been successfully tested on Microsoft Windows 7, Linux (Ubuntu 10.04 LTS), and Mac OS X versions 10.6.8 (Snow Leopard) to 10.9.2 (Mavericks). See, for example, the Java™ SE download page¹. If you encounter any issues with your operating system, the chapter 6 on page 28 might provide answers to you. Otherwise, contact the developer team mailing list sbmlsimulator@googlegroups.com.

2.1.2 Hardware

With at least 1 GB main memory, you should be able to perform most tasks without any problem. An active internet connection is not required for simulations, but for some other application features (e.g., on-line update).

2.2 Starting the application



If you downloaded a ZIP-file, you need to unzip it before starting the application. In the most simple case, you can launch SBMLsimulator by double-clicking at the Java™ application icon (see image next to this text). You can also start the application on all operating system by typing

```
java -jar -Xms128m -Xmx1024m SBMLsimulator.jar
```

¹<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

on your command prompt. Please note that you might have to change `SBMLsimulator.jar` for the real name of the JAR file, e.g., `SBMLsimulator_v1.2.jar`. In this example, a minimum of 128 MB and a maximum of 1024 MB of memory will be available for the program. In most cases, SBMLsimulator needs more than 128 MB memory, so it might be convenient to create a shortcut and start the application with as much memory as available. If you have 2 GB RAM, for example, you might want to start the application with the following command:

```
java -Xms128m -Xmx1400M -jar SBMLsimulator.jar
```

To simplify matters, you could create a start-scripts to run the application with as much memory as possible. How much memory is actually needed strongly depends on the size of your input datasets.

SBMLsimulator is a bi-lingual program, which has been translated to German in addition to its English user interface. When launching the program, it uses English, unless your operating system has a German environment. You can select the language of the program by passing the parameter `-Duser.language=en` to the Java™ Virtual Machine (JVM) upon the start of the program if you like to have an English user interface, or `de` for the German interface.

3 How to get started


To demonstrate the use of SBMLsimulator, we use the model by Bucher *et al.* (2011). You can download this model from BioModels Database (Le Novère *et al.*, 2006) under accession N^o 328.

There are also many other published models available from this website. All published models that have been curated by the team of model curators are listed at the website of BioModels Database¹. If you like to download a model, click on the respective ID, which brings you to a page containing a description of the model and a button “Download SBML”. Position the cursor at this button and then you see the possible SBML levels (abbreviated with L) and versions (abbreviated with V) to download. If you click on the respective combination of level and version, the desired SBML file will be downloaded.

Alternatively, you can also create an SBML model yourself. To this end, dedicated software solutions like CellDesigner (Funahashi *et al.*, 2003) can be used. For downloading CellDesigner please go to the project’s website².

In order to perform model simulations, you first need to load an SBML model. If you additionally want to compare the simulation results to experimental data or run a parameter estimation, it is required to also load the experimental data into SBMLsimulator. In our example, we generate artificial data from the example model. You can obtain artificial data, too, by running a simulation with SBMLsimulator and exporting the numerical solution to a CSV file. This procedure is explained in detail below.

3.1 Open a model file

 You can simply drag&drop the model file (an SBML document) into the application or select **File** » **Open** » **Open model** to load a model (see fig. 3.1 on the next page). Another option would be to click at the folder icon in the tool-bar, which is displayed next to this text. The tool-bar contains the icon that is displayed next to this text. Depending on your operating system, you can also use the key stroke combinations **⌘** + **O** (Mac OS X) or **Ctrl** + **O** (Linux and Microsoft Windows). SBMLsimulator memorizes up to ten previously opened SBML documents in the menu under **File** » **Recent files**, where the most recently used file will appear at the top most position. You can also use the keystroke combinations **⇧** or **Alt** + **0** to **9** (operating system-dependent).

Now, the model is loaded and the window in fig. 3.2 on page 6 appears. The right part of the window is for plotting simulation results and experimental data points (if experimental data has

¹<http://www.ebi.ac.uk/biomodels-main/publmodels>

²www.celldesigner.org

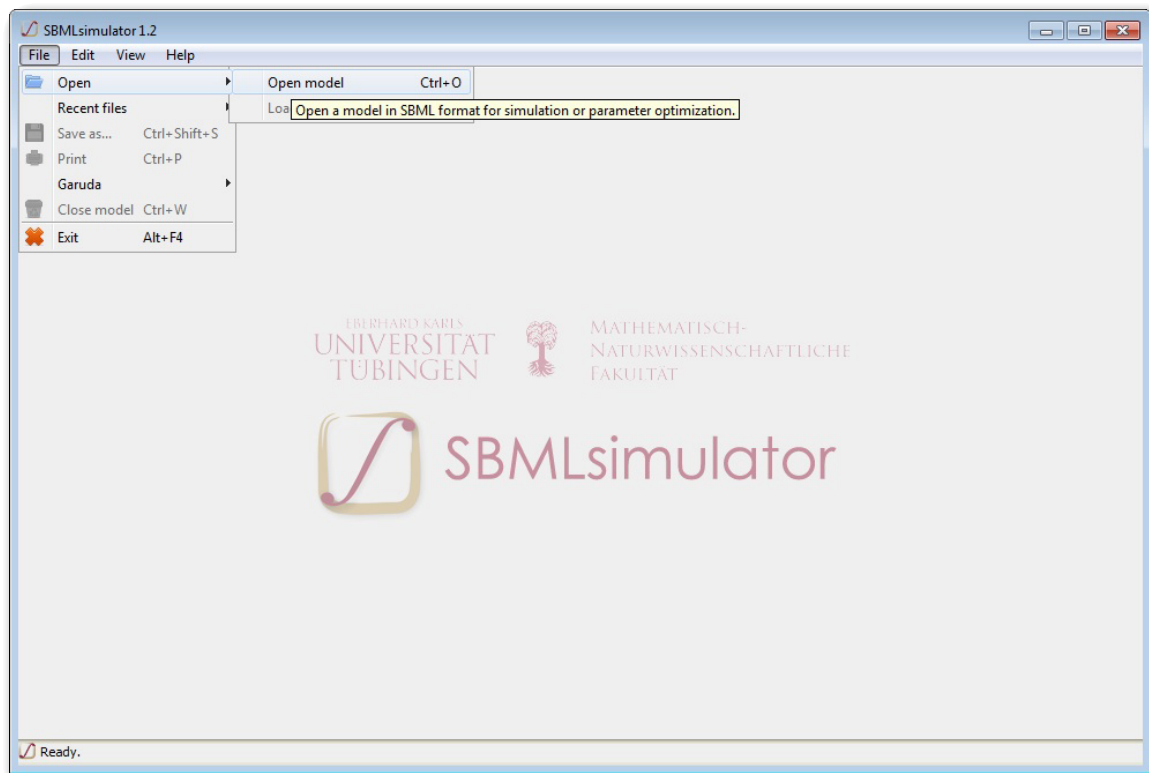


Figure 3.1: Loading of a model. One possibility to open a model is to select **File** **Open** **Open model**.

been loaded). In the bottom part of the window you can determine simulation preferences like the solver or the number of simulation steps and the quality function for comparing experimental data (if present) to simulation results. In the center left part of the window the compartment values, the initial values of species, and parameter values in the model can be changed. After typing a new value in, it is necessary to either press the return key , or the tabulator key . Otherwise the changes cannot take effect. The upper left part is for choosing the quantities that should appear in the plot.

3.2 Simulate a model

You can now select the simulations results of which elements should be plotted by clicking at the individual check boxes for model components of interest. The buttons **Select** and **Deselect** give you the choice to visualize or remove an entire group of elements from the plot, such as all species or all fluxes. For instance, select to plot all species values (see fig. 3.3 on

3 How to get started

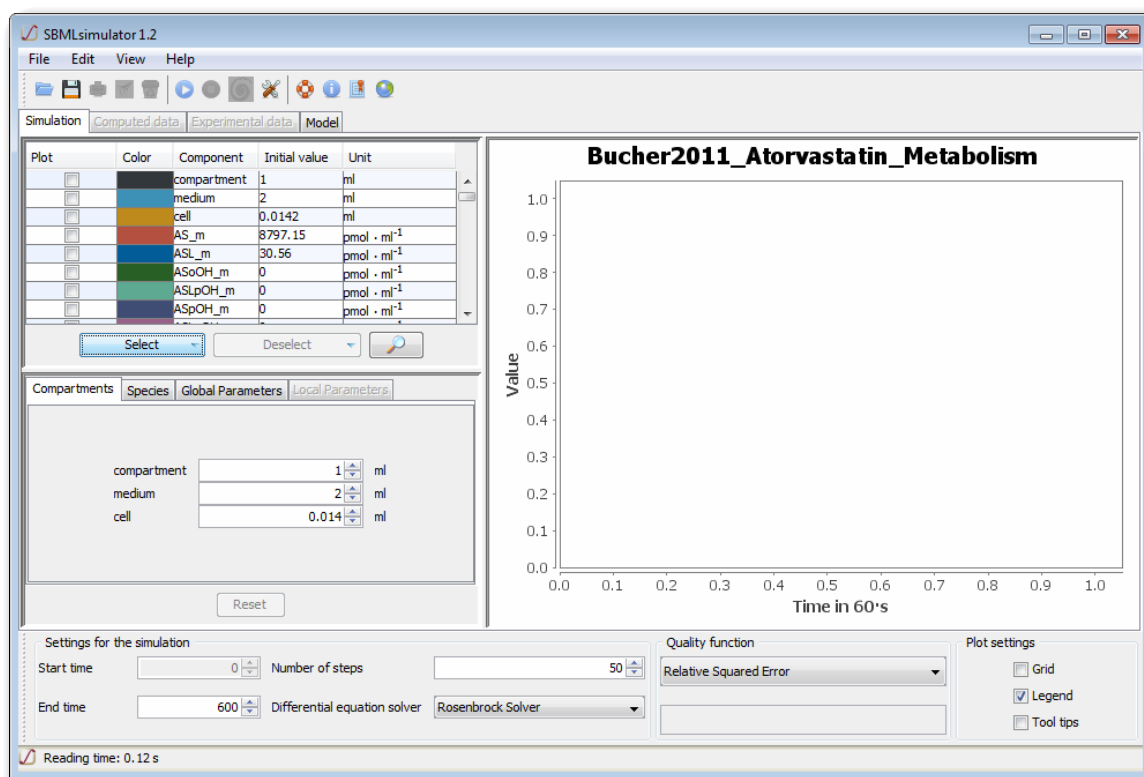


Figure 3.2: SBMLsimulator with a loaded model. The window lets the user choose the simulation settings (bottom), the parameter values (center left), and the quantities to plot (upper left). The plot section is on the right.

the next page). Selecting model components is necessary, because SBMLsimulator does not know which model components you like to display in the plot. Then you have to click on the simulation button (fig. 3.4 on page 9). The simulation results can then be displayed in the plot section (fig. 3.5 on page 10).

SBMLsimulator offers a large variety of integration routines. The solver, which should work best for all models, is the Rosenbrock solver. However, its high preciseness comes at the cost of a running time longer than that of the other solvers. Table 3.1 lists all solvers provided by SBMLsimulator together with a short description and a reference for more information.

■ If in some cases a simulation run consumes too much time, you can always interrupt it by clicking at the stop button in the tool-bar (see image at the side of this text), which is activated whenever you launch a simulation.

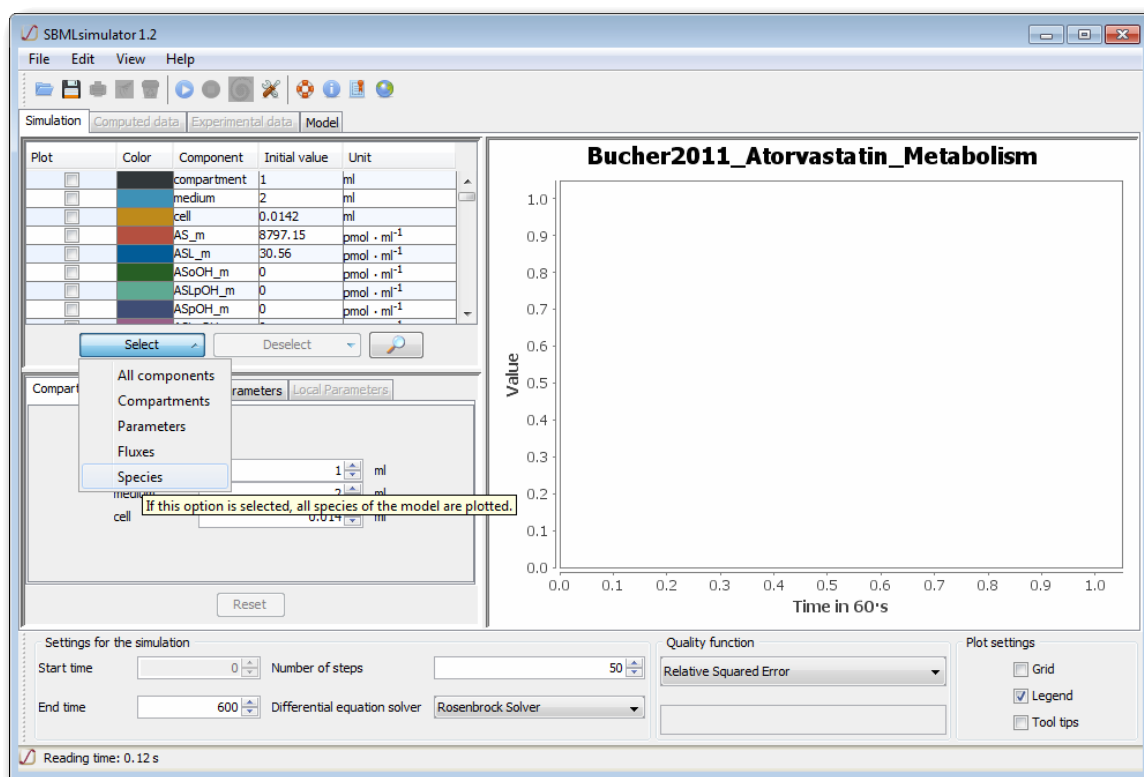


Figure 3.3: Selection of quantities for plotting. The user can click on **Select** **Species** in order to choose all species for plotting.

Table 3.1: Solvers provided by SBMLsimulator

Name	Description	Reference
Adams-Bashforth	This is an explicit multi-step integration method. The solver uses a step-size adaptation. It is faster than Rosenbrock, but not applicable for very stiff differential equation systems. This solver is provided by Apache Commons.	Apache Software Foundation (2013)
Adams-Moulton	The method by Adams and Moulton is an implicit multi-step integration method, which uses a step-size adaptation. It is also faster than Rosenbrock, but not applicable for all models. This solver is provided by Apache Commons.	Apache Software Foundation (2013)

Continued on the next page...

Table 3.1: Solvers provided by SBMLsimulator (continued)

Name	Description	Reference
Dormand-Prince 54	The Dormand Prince in an explicit integration method of order 5, which belongs to the family of Runge-Kutta methods. Step-size adaptation is included in the routine. This solver works for most models, but can have problems with extremely stiff differential equation systems. It is provided by Apache Commons.	Apache Software Foundation (2013)
Dormand-Prince 853	This solver is similar to Dormand-Prince 54, but it is of order 8. It comprises more function evaluations than Dormand-Prince 54 and is therefore more precise, but also slower. It is provided by Apache Commons.	Apache Software Foundation (2013)
Euler	The explicit Euler method. This is a very fast and simple solver, that lacks a step-size adaptation and might therefore be imprecise.	Press <i>et al.</i> (1992)
Gragg-Bulirsch-Stoer	This method belongs to the most efficient and accurate methods with step-size adaptation for non-stiff differential equations. Its use is not recommended for stiff differential equations. This solver is provided by Apache Commons.	Apache Software Foundation (2013)
Higham-Hall 54	The method by Higham and Hall is a Runge-Kutta method of order 5 with step-size control. The solver is similar to the Dormand-Prince 54 solver. It is provided by Apache Commons.	Apache Software Foundation (2013)
Rosenbrock	This adapted solver comprises Rosenbrock's method, which includes an adaptation of step-size. This implementation is specifically dedicated to precise simulation of SBML models and is applicable for very stiff differential equation systems. But it might be significantly slower for some models than the other solvers.	Press <i>et al.</i> (1992)
Runge-Kutta	The classical fourth order Runge-Kutta method. It is fast, but imprecise for many models, as it does not contain a step-size adaptation.	Press <i>et al.</i> (1992)

3.3 Explore the tree structure of the model

Choose the tab **Model** and you see SBML document in a tree structure that you can explore (fig. 3.6 on page 11).

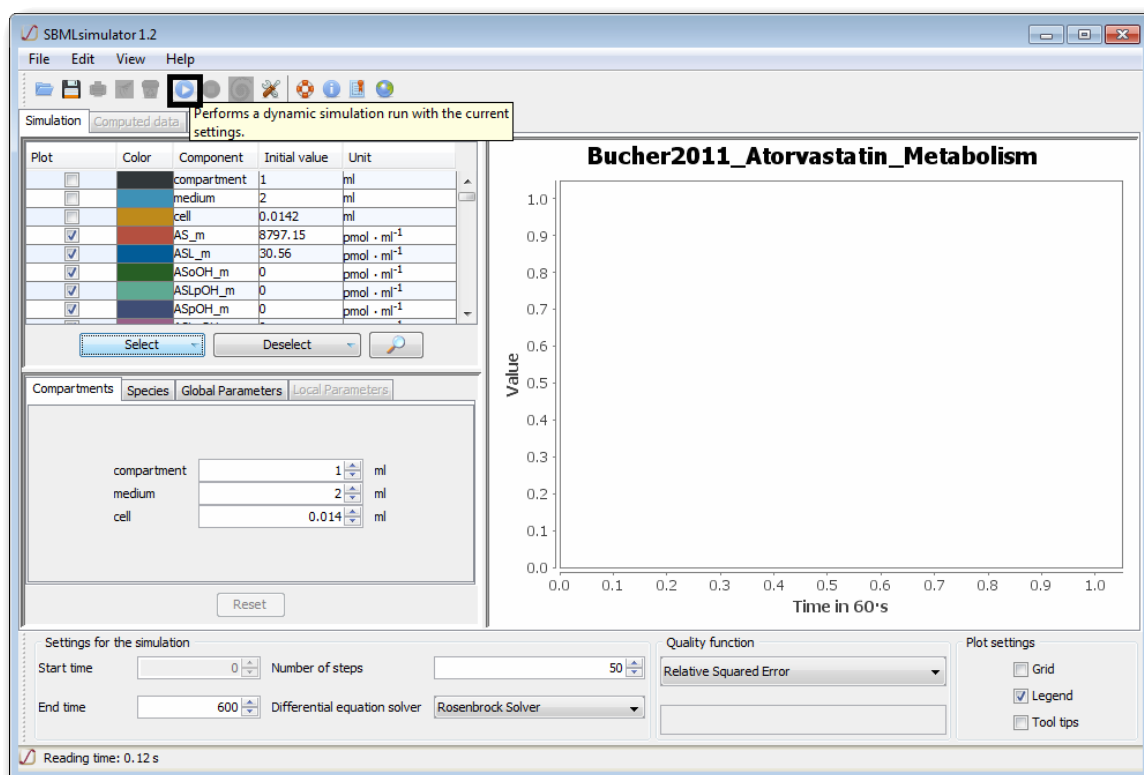


Figure 3.4: Starting of a simulation in SBMLsimulator. By clicking on the simulation button, the simulation of the model is started.

A search function at the bottom gives you the opportunity to browse the model for certain components. Here you can search for Identifiers (Ids) or names of components. The tree will be expanded and elements that do not match your filter criteria will be removed from the view. When clicking at individual model components, the program displays detailed information about the selected element.

3.4 Save plot image

If you like to save the current plot as an image, just select the tab **Simulation**. You can then save the image of the results plot by clicking at **File > Save as**. For users who prefer keystroke combinations, it is also possible to save the plot by hitting the keys **⌘ + ⬆ + S** if you are working under Mac OS X, or **Ctrl + ⬆ + S** for Linux and Microsoft Windows. By clicking at the printer icon (displayed next to this text) you can print the current plot or save it as a Portable Document Format (PDF). The print function can also be used with the keystroke combination **⌘**

3 How to get started

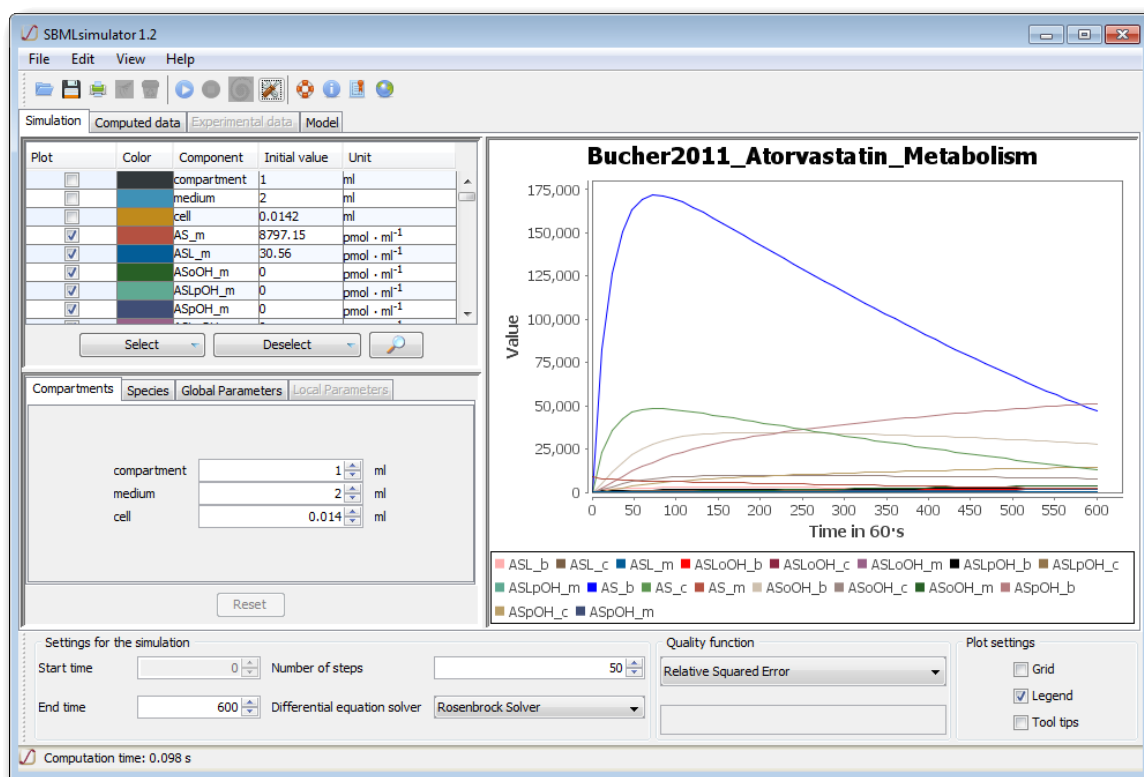


Figure 3.5: Simulation results plotted in SBMLsimulator. The simulation results are plotted in the right part of the window.

+ **P** (Mac OS X) or **Ctrl** + **P** (Microsoft Windows and Linux).

3.5 Save simulation data

You need to select the tab **Computed data** (see fig. 3.7 on page 12). Then you can save the simulation data under **File** > **Save as**. Just like for the plot data, you can also use keystroke combinations to save your simulation data. To this end, just select the tab **Computed data** and hit the keys **⌘** + **↑** + **S** if you are working under Mac OS X, or **Ctrl** + **↑** + **S** for Linux and Microsoft Windows.

3.6 Save model

In order to save the model with the parameter, species and compartment values that you have modified, you first have to select the tab **Model**. Then you can save the simulation data under **File** >

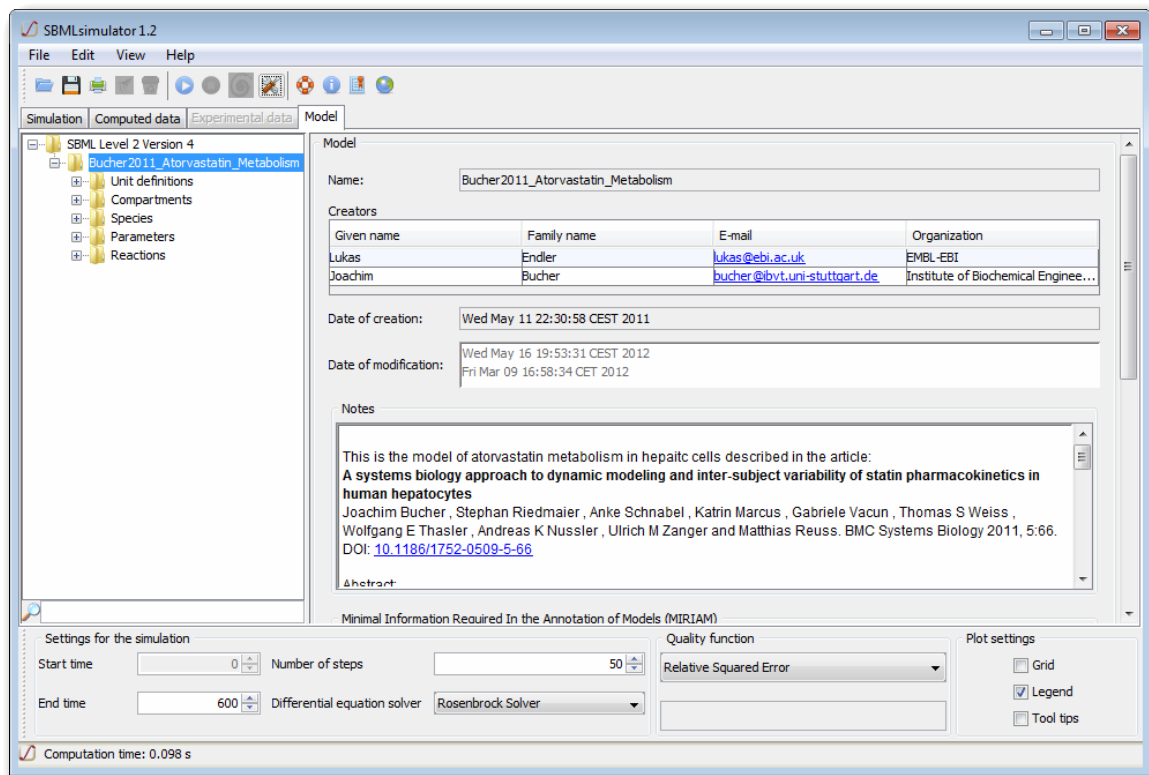


Figure 3.6: Display of model tree. The model tree is shown when clicking on the **Model** tab. The user can explore the model by clicking at elements within the tree.

Save as. Again, also for this purpose you can use keystroke the combinations **⌘** + **⇧** + **S** (for Mac OS X) or **Ctrl** + **⇧** + **S** (for Linux and Microsoft Windows) when the **Model** tab is selected.

3.7 Prepare experimental data

First of all, make sure that your experimental data is in one of the required file formats. The application can read CSV files, which are mostly tab-separated files with data. To use Microsoft Excel-data, you can simply open your Microsoft Excel spreadsheet, click **File** > **Save as** and select “Tab-separated text file”. For all files, we require one column called “Time” that should contain for each row the time point it refers to. The other columns should be called with the Id of the respective quantity in the model and should contain the measured values at the time points, or Not a Number (NaN) if the measurement for a time point is missing.

3 How to get started

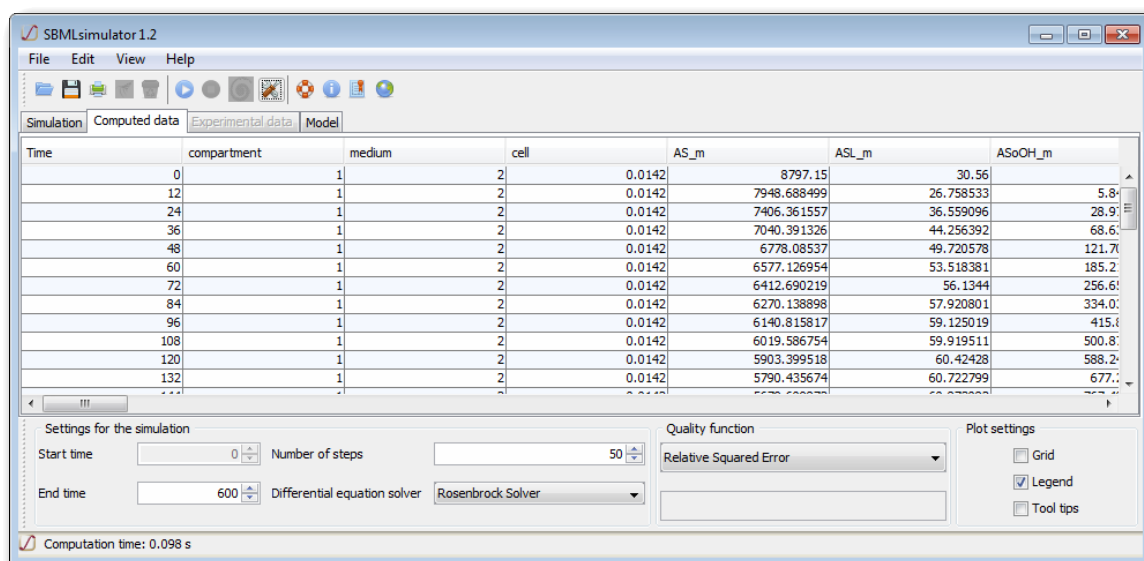


Figure 3.7: Table with simulation data. Under the tab **Computed data** the user can see the simulation results in a table. They can be saved under **File > Save as**.

3.7.1 Experimental data example

As described, you need a column with the time points and multiple columns that state the values (often concentrations) of each quantity at each time point. The following is an example how an input file may look like:

Listing 3.1: Input file example for experimental data

Time	s1	s2	s3
0	32.456E-12	7.02	12E32
4	5E-54	179.05E-14	0.005
...

3.8 Open experimental data in SBMLsimulator

In general, all data must be processed expression data in tabular text files (see section 3.7 on page 11 for descriptions and examples of the input file formats). Then, there are many possibilities to open datasets in SBMLsimulator. For example, you can simply drag&drop data into the application or select **File > Open > Load experimental data** to open a file. Just like in the case of model files, you can also use the key stroke combination **⌘ + O** or **Ctrl + O** depending on your operating system.

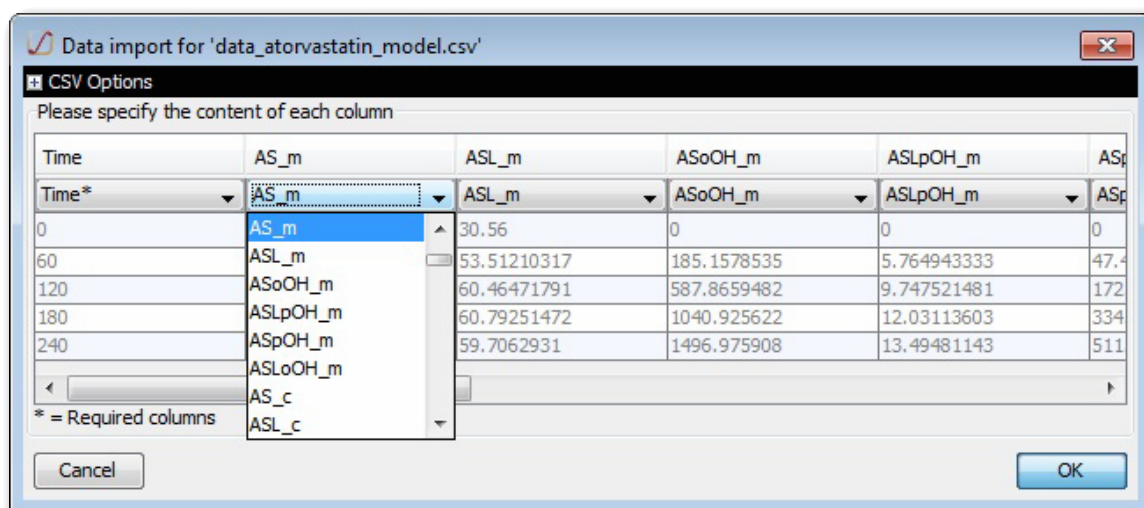


Figure 3.8: Example of the data import dialog. The “CSV Options” panel might be expanded to correct auto-detected input file properties. At the table in the lower half, the content of each column must be specified.

Please note that you cannot open experimental data before loading a model. This is because an SBML file can contain various meta information about model components, whereas the simple CSV format might not suffice to build the complex data structure for the display.

The fig. 3.8 shows an example of the file input dialog. Here we open the simulation data saved in section 3.5 on page 10. This data have been restricted to the quantities that were experimentally measured for the parameter estimation in the respective publication by Bucher *et al.* (2011). If the input file format has not been inferred automatically (which is not the case here), one may click on the black “CSV Options” label to specify further options (like “column separator char” or if the file contains headers—see fig. 3.9).

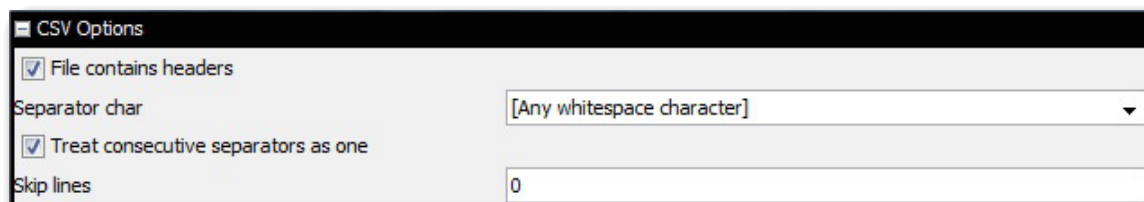


Figure 3.9: Expanded “CSV Options” allow to give details for the input file. These properties are auto-detected and only need to be changed if the auto-detection failed to correctly infer those properties.

3 How to get started

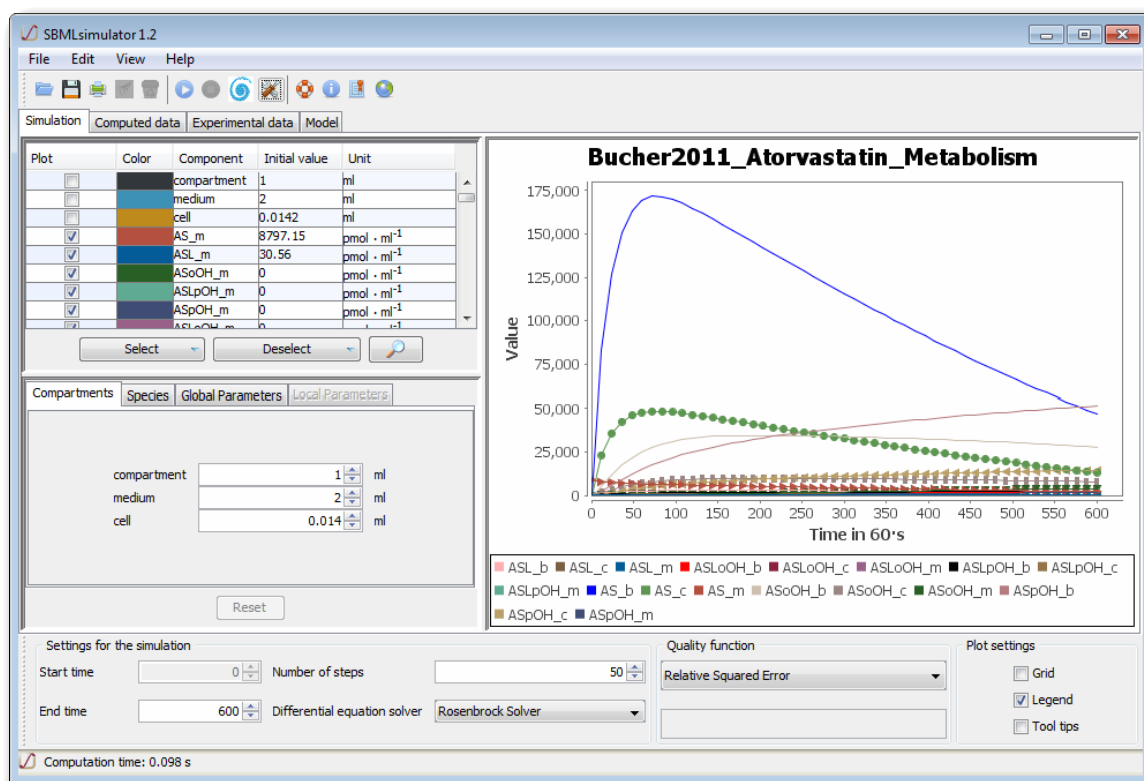


Figure 3.10: SBMLsimulator after simulation and loading of experimental data. The plot shows the simulation results and the experimental data points as dots.

In the table at the lower half of this dialog, one must specify the content of each column. This can be done by clicking on the combo box below the captions.

After the experimental data are uploaded, the plot shows the respective data points as dots (see fig. 3.10).

3.9 Parameter estimation

In many cases, biological models contain quantities with uncertain values. These values must be estimated by calibrating the model with respect to given experimental data. To facilitate this complicated procedure, SBMLsimulator comes with the optimization toolbox EvA2 that provides a large collection of nature inspired heuristic optimization procedures, e.g., evolution strategies (Rechenberg, 1973; Schwefel, 1975), genetic algorithms (Holland, 1975), differential evolution (Storn, 1996), or particle swarm optimization (Clerc and Kennedy, 2002; Clerc, 2005), as well as niche-based methods that have been found to be particularly useful for applications in

systems biology (Kronfeld *et al.*, 2009).

Click on the estimation button (see the icon next to this text). Then a window appears, in which the quantities to estimate and their ranges can be selected (see fig. 3.11 on the next page). This can be done manually or by uploading a CSV file. In order to upload a file, click on **Open**. You can also save your configuration by clicking on **Save**.

In a configuration file, you can set the minimum and maximum values for initialization of the estimated quantities at the beginning of the estimation (initial minimum/maximum) as well as the minimum and maximum value for each estimated quantity that is allowed during the whole estimation (minimum/maximum). Here is an example how such a file might look like:

Listing 3.2: Input file example for parameter estimation file

[Id]	[initialMinimum]	[initialMaximum]	[minimum]	[maximum]
Import_ASUpOH_k	1E-7	0.1	1E-7	0.1
Import_ASLoOH_k	1E-7	0.1	1E-7	0.1
Import_ASU_k	1E-7	1	1E-7	1
...

After you are satisfied with your selected quantities, click on **OK**. Then EvA2 is started and the respective window (see fig. 3.12 on page 17) lets you select the estimation method (“optimizer”) with specific settings (see fig. 3.13 on page 18) and the termination criterion (“terminator”). When you click on **Start**, the estimation begins. Please note: If there are initial values (i.e., values at time point 0) for some model quantities given in the data, SBMLsimulator takes these values as initial values for the simulations necessary during the estimation. In the case of multiple experimental datasets loaded, the medians of the initial values are taken. For the quantities without initial values given in the data, the initial values defined in the model are chosen.

The simulation result with the current best parameter values is always plotted during estimation and the current quality function value is displayed (fig. 3.14 on page 19).

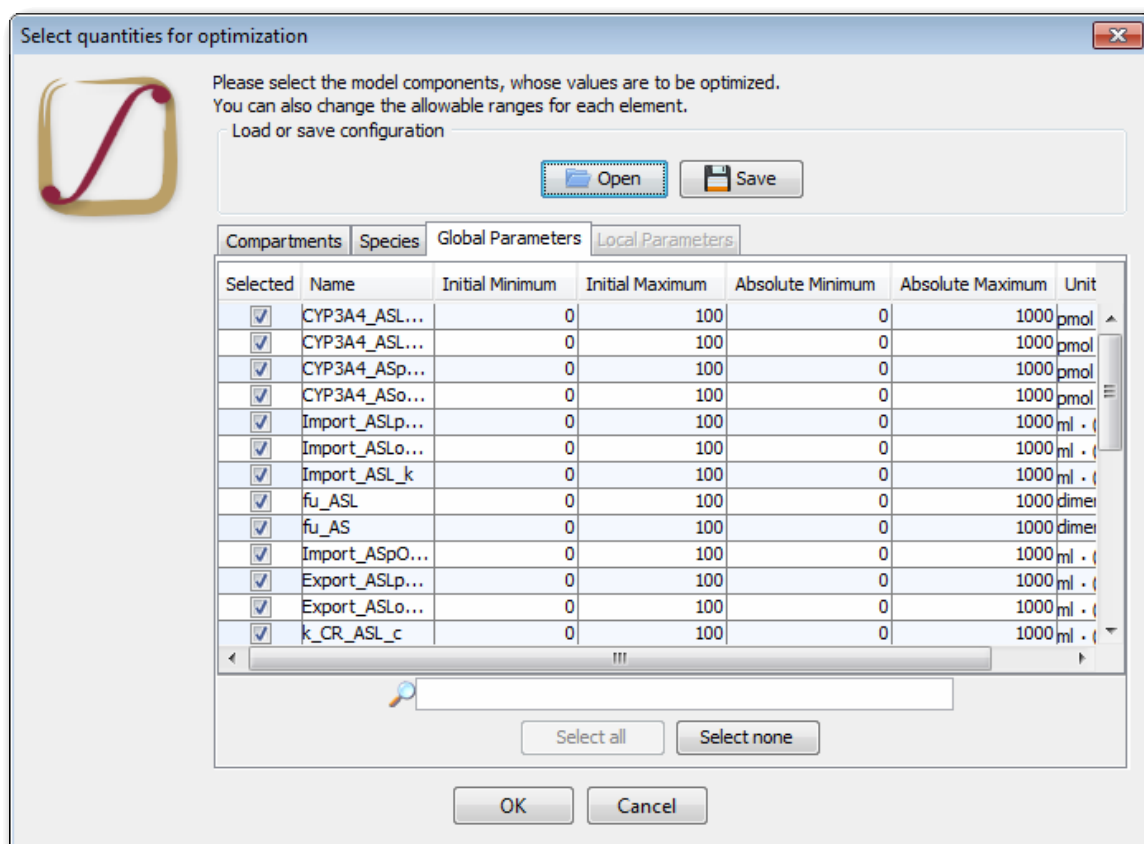


Figure 3.11: Selection of quantities to estimate. The user can select the quantities to estimate and their ranges manually by modifying the displayed table or via uploading a CSV file by clicking on **Open**. The current configuration of quantities and their ranges (for initialization as well as for the whole estimation process) can be saved by clicking the **Save** button. The search function at the bottom helps you to navigate through the names or identifiers that are potential optimization targets. When you type a name in the text field, the above table will be reduced to elements that contain this name you are typing. The buttons **Select all** and **Select none** are also helpful to choose all or none of the elements within one tab as optimization targets. Note that each tab has these buttons separately, so your choice in one tab does not affect others.

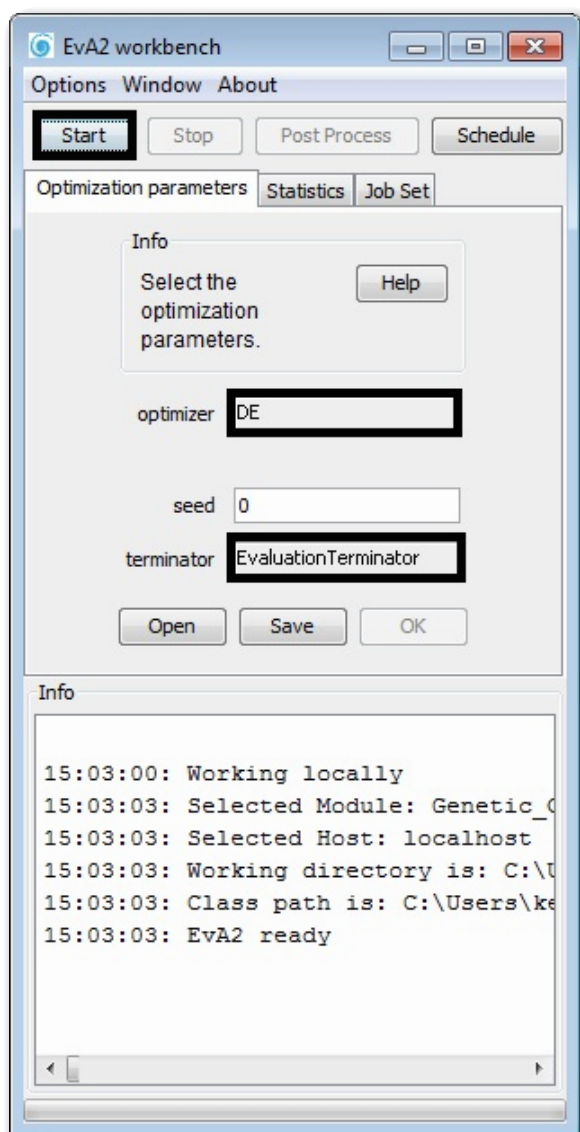
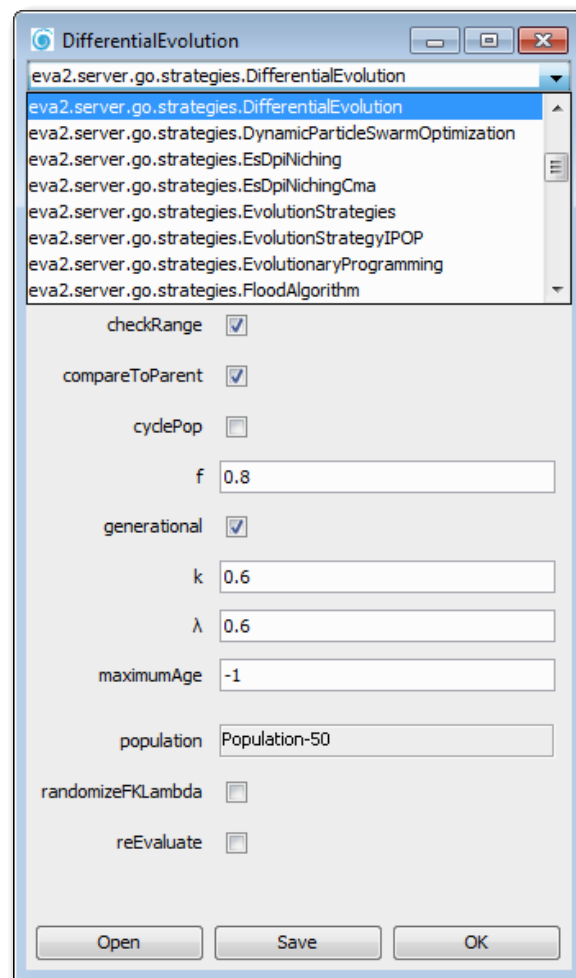


Figure 3.12: EvA2 settings window. EvA2 lets the user choose the parameter estimation method and the specific settings: just click at the text field next to the label “optimizer”. A separate dialog window will appear that offers a large number of optimization procedures (fig. 3.13 on the following page). In the same way, an appropriate termination criterion can also be set by clicking at the text field that is labeled “terminator”. After clicking on **Start**, the estimation begins. During the optimization, a window will appear that displays the quality improvement of the solution, the so-called *fitness*. This is the distance between your experimental data and the simulation result for the currently best parameter set. If the optimization is successful, the fitness will decrease as time proceeds. You can interrupt the optimization at any time by hitting the **Stop** button. When your selected termination criterion is reached, the optimization will stop and you can continue working with your optimized model in the main window of SBMLsimulator.

Figure 3.13: Changing the optimization procedure. The user can change the optimization procedure in this window. Apart from Differential Evolution, many other routines like Particle Swarm Optimization are provided. A parameter of a routine can be changed by clicking in the text field right of it and typing the new value or by clicking in the check box, respectively.



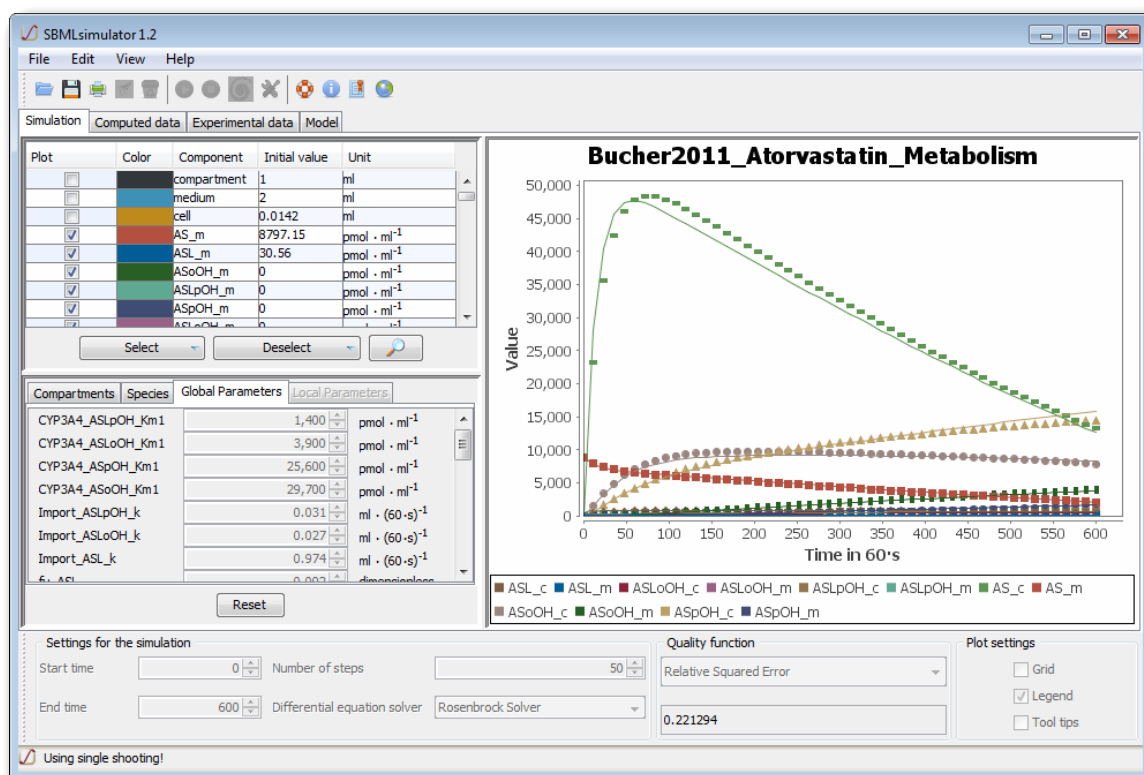


Figure 3.14: Intermediate estimation result. After each generation of solutions by EvA2 the simulation result with the current best parameter estimation is plotted. The current value of the quality function is also shown.

4 Command-line arguments and preferences

It now follows a short overview about all possible command-line options of SBMLsimulator. All functions of the program are also available from the command line. This means that you can also run SBMLsimulator on a cluster and use it for large-scale model calibration. Furthermore, you can use the command-line options in combination with the Graphical User Interface (GUI). One possible use-case scenario would be to launch the program with your desired configuration with a customized start script, e.g., to directly open a certain model file when launching the program.

4.1 Simulator input/output options

4.1.1 Select input files

These options are used to select all input files for the simulation.

`--sbml-input-file[|=]<File>` Select a model in SBML format that is to be simulated. Accepts SBML files (*.sbml, *.xml). Default value: none

`--time-series-file[|=]<File>` Path to a file with a time series of species, compartment, or parameter values. Accepts CSV files (*.csv). Default value: none

4.1.2 Select output files

Select output files for the results of simulation and parameter estimation.

`--sbml-output-file[|=]<File>` Select a file where to store the model in SBML format after parameter optimization. Accepts SBML files (*.sbml, *.xml). Default value: none

`--simulation-output-file[|=]<File>` Select a file where to store the results of a simulation. Accepts CSV files (*.csv). Default value: none

4.2 Simulation options

4.2.1 Missing values

Decide how to treat the values of compartments, species, and parameters if no initial value has been defined in the model. A missing initial value is to be distinguished from invalid, i.e., NaN, values. Furthermore, the elements may have very diverse the units associated with them. Here it is only possible to define one numeric value for each element group, irrespective of any unit.

-
- `--default-init-compartment-size [|=]<Double>` If not specified the value corresponding to this argument will be used to initialize the size of compartments. Arguments must fit into the range $(0.0, 9 \cdot 10^9]$. Default value: 1.0
- `--default-init-species-value [|=]<Double>` If not specified the value corresponding to this argument will be used to initialize species depending on their `hasOnlySubstanceUnits` property as initial amount or initial concentration. Arguments must fit into the range $(0.0, 9 \cdot 10^9]$. Default value: 1.0
- `--default-init-parameter-value [|=]<Double>` The default initial value that is set for parameters with undefined value. Arguments must fit into the range $(0.0, 9 \cdot 10^9]$. Default value: 1.0

4.2.2 Settings for the simulation

Here you can specify parameters for the simulation of the model.

- `--ode-solver [|=]<Class>` This gives the class name of the default solver for Ordinary Differential Equation (ODE) systems. The associated class must implement `AbstractDESSolver` and must have a constructor without any parameters. All possible values for type `<Class>` are:
- `org.simulator.math.odes.AdamsBashforthSolver`,
 - `org.simulator.math.odes.AdamsMoultonSolver`,
 - `org.simulator.math.odes.DormandPrince54Solver`,
 - `org.simulator.math.odes.DormandPrince853Solver`,
 - `org.simulator.math.odes.EulerMethod`,
 - `org.simulator.math.odes.GraggBulirschStoerSolver`,
 - `org.simulator.math.odes.HighamHall54Solver`,
 - `org.simulator.math.odes.RosenbrockSolver`, and
 - `org.simulator.math.odes.RungeKutta_EventSolver`.

Default value: `org.simulator.math.odes.RungeKutta_EventSolver`

- `--abs-tol [|=]<Double>` Allowed absolute vectorial error. Default value: 1.0E-10
- `--rel-tol [|=]<Double>` Allowed relative vectorial error. Default value: 1.0E-6
- `--sim-start-time [|=]<Double>` The double value associated with this key must, in case of SBML equal to zero. Generally, any start time would be possible. This is why this key exists. But SBML is defined to start its simulation at the time zero. Arguments must fit into the range $[0, 10^5]$. Default value: 0.0

- `--sim-end-time [|=]<Double>` With the associated non-negative double number that has to be greater than 0 when simulating SBML models, it is possible to perform a simulation. Arguments must fit into the range $(0, 10^5]$. Default value: 5.0
- `--sim-step-size [|=]<Double>` The greater this value the longer the computation time, but the more accurate will be the result. Arguments must fit into the range $(0, 10^5]$. Default value: 0.01

4.3 Estimation Options

4.3.1 Spline approximation

These options allow you to estimate parameters with respect to spline interpolation values between given measurement data and to configure how to calculate these splines.

- `--fit-to-splines` If this is selected, splines will be calculated from given experimental data and the parameter estimation procedure will fit the system to the splines instead of the original values. The advantage of this procedure is that the amount of available data is increased due to this form of interpolation, also ensuring that the shape of the resulting curves comes close to what could be expected. The disadvantage is that the influence of potential outliers on the overall fitness is increased. Default value: false
- `--number-of-spline-samples [|=]<Integer>` This defines the number of additional spline sampling points between the measurement data. If you select zero, only the real sampling points will be used. Default value: 50

4.3.2 Default optimization targets

Select the default optimization targets.

- `--est-all-compartments` Decide whether or not by default all compartments in a model should be considered the target of an optimization, i.e., value estimation. Default value: false
- `--est-all-global-parameters [|=]<Boolean>` Decide whether or not by default all global parameters in a model should be considered the target of an optimization, i.e., value estimation. Default value: true
- `--est-all-local-parameters [|=]<Boolean>` Decide whether or not by default all local parameters in a model should be considered the target of an optimization, i.e., value estimation. Default value: true
- `--est-all-species` Decide whether or not by default all species in a model should be considered the target of an optimization, i.e., value estimation. Default value: false

`--est-all-undefined-quantities` Estimates the values for all those quantities in the model whose values are either undefined or set to NaN. Default value: `false`

4.3.3 Ranges of all optimization targets

Define the initial and absolute ranges of all optimization targets.

`--est-init-min-value [|=]<Double>` The minimal value of the initialization range in a parameter estimation procedure. Default value: `0.0`

`--est-init-max-value [|=]<Double>` The maximal value of the initialization range in a parameter estimation procedure. Default value: `10.0`

`--est-min-value [|=]<Double>` The minimal value in the absolute allowable range in a parameter estimation procedure. Default value: `0.0`

`--est-max-value [|=]<Double>` The maximal value in the absolute allowable range in a parameter estimation procedure. Default value: `1000.0`

4.3.4 Integration strategy

Select whether or not to apply a multiple shooting strategy.

`--est-multi-shoot [|=]<Boolean>` Decide whether a model calibration should be done using multiple shoot technique. This should be the default. The other possibility is the so-called single shoot technique. This means that only one initial value is taken to integrate the ordinary differential equation system, whereas the multiple shoot technique restarts the integration in each time step given the values in this step. The aim is then to come as close as possible to the start value in the next time step. In many cases the fitness landscape becomes much more friendly when using a multiple shoot strategy. Default value: `true`

`--use-existing-solution` Select whether or not to use the parameters in the existing model for post-optimization. Default value: `false`

4.3.5 Quality function

Here you can specify how to evaluate the quality of a parameter set with respect to given experimental data.

`--quality-measure [|=]<Class>` This specifies the class name of the default quality function that evaluates the quality of a simulation with respect to given (experimental) data. All possible values for type `<Class>` are:

- `org.simulator.math.EuclideanDistance,`
- `org.simulator.math.ManhattanDistance,`

- `org.simulator.math.N_Metric`,
- `org.simulator.math.PearsonCorrelation`,
- `org.simulator.math.RelativeEuclideanDistance`,
- `org.simulator.math.RelativeManhattanDistance`,
- `org.simulator.math.RelativeSquaredError`, and
- `org.simulator.math.Relative_N_Metric`.

Default value: `org.simulator.math.RelativeSquaredError`

`--quality-default-value [|=] <Double>` The default return value of a quality function that can be used if for some reason a quality cannot be computed. For example, this might avoid a division by zero. Default value: `1000.0`

`--quality-n-metric-root [|=] <Double>` The root parameter in the distance function for n -metrics: in case of the n -norm this is at the same time also the exponent. For instance, the Eulidean distance has a root value of two, whereas the Manhattan norm has a root of one. In the Relative Squared Error (RSE), the default root is also two, but this value may be changed. Default value: `3.0`

4.3.6 Additional options

`--est-targets [|=] <String>` The file with the values to estimate and their initial setting.

4.4 Options for the graphical user interface

4.4.1 Additional options

`--check-for-updates [|=] <Boolean>` Decide whether or not this program should search for updates at start-up. Default value: `true`

`--gui` If this option is given, the program will display its Graphical User Interface (GUI). Default value: `false`

`--log-level [|=] <String>` Change the log-level of this application. This option will influence how fine-grained error and other log messages will be that you receive while executing this program. All possible values for type `<String>` are: `ALL`, `CONFIG`, `FINE`, `FINER`, `FINEST`, `INFO`, `OFF`, `SEVERE`, and `WARNING`. Default value: `INFO`

4.5 Plot Options

4.5.1 Plotting panel options

Options for the visible rectangle of the plot panel.

- `--show-plot-grid` Decide whether or not to draw a grid in the plot area. Default value: `false`
- `--show-plot-legend[|=]<Boolean>` Add or remove a legend in the plot. Default value: `true`
- `--show-plot-tooltips` Let the plot display tool tips for each curve. Default value: `false`

4.5.2 Appearance of the plot

Options for appearance of the plot area.

- `--plot-background-color[|=]<Color>` The background color of the plot Default value:
`java.awt.Color[r=255,g=255,b=255]` (white)
- `--plot-grid-color[|=]<Color>` The color of the plot's grid Default value:
`java.awt.Color[r=64,g=64,b=64]` (anthracite)

4.6 CSV options

4.6.1 CSV file selection

Select the character-separated file to be opened.

- `--csv-file[|=]<File>` A CSV file to be opened. Default value: `none`

4.6.2 CSV file characters

Define the special characters in CSV files that separate values or that quote strings of characters.

- `--csv-files-separator-char[|=]<Character>` The separator character that is written between the entries of a character separated value file. Not that actually any Unicode Transformation Format (UTF)-8 character can be used as a separator, not only commas. All possible values for type `<Character>` are: `→`, `↵`, `,`, `/`, `;`, and `|`. Default value: `,`
- `--csv-files-quote-char[|=]<Character>` The character that is used to quote strings inside of CSV files in order to prevent a miss-interpretation of white spaces with separator characters. Default value: `"`

4.7 Garuda options

Settings for the communication with a Garuda Core and hence access to other Garuda gadgets.

`--connect-to-garuda[|=]<Boolean>` Decides whether or not the current application should attempt to connect to the Garuda Core (Ghosh *et al.*, 2011). Default value: true

5 License

SBMLsimulator is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.



This program is distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of **MERCHANTABILITY** or **FITNESS FOR A PARTICULAR PURPOSE**. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program. If not, see <http://www.gnu.org/licenses/lgpl-3.0-standalone.html>.

5.1 Included third-party libraries and packages

SBMLsimulator includes and redistributes the following software packages:

- JSBML (Dräger *et al.*, 2011), which is freely available under the terms of the GNU Lesser General Public License (LGPL) version 2.1, see <http://sbml.org/Software/JSBML>.
- the Systems Biology Simulation Core Library (SBSCCL) (Keller *et al.*, 2013), which is licensed under the LGPL version 3.1, see <http://simulation-core.sourceforge.net>.
- the nature-inspired heuristic optimization framework EvA2 (LGPL Version 3, Kronfeld *et al.*, 2010), see <http://www.cogsys.cs.uni-tuebingen.de/software/EvA2>.
- Apache Commons-math under the terms of the Apache Software License, Version 2.0, see <http://commons.apache.org/math/>.
- JCommon, licensed under LGPL Version 2.1, see <http://jfree.org/jcommon>.
- JFreeChart, licensed under LGPL Version 3.1, see <http://www.jfree.org/jfreechart>.
- Icons: www.pixel-mixer.com.
- Mac OS X support: Quaqua Look and Feel, license LGPL Version 2.1, see <http://www.randelshofer.ch/quaqua/>.

6 FAQ / Troubleshooting

Where can I get help for a certain component/ option/ check-box/ etc.?

Most elements in SBMLsimulator have tool-tips. If you do not understand an option, you can get help in the first place by just pointing the mouse cursor over it and wait for the tool-tip to show up (~ 3 seconds).

I'm getting a "java.lang.OutOfMemoryError: Java heap space"

Some operations need a lot of memory. If you simply start SBMLsimulator, without any JVM parameters, only 64 MB of memory are available. Please append the argument `-Xmx1024M` to start the application with 1 GB of main memory. See section 2.2 on page 2 for a more detailed description of how to start the application with additional memory. If possible, you should give the application 2 GB of main memory. A minimum of 1 GB main memory should be available to the application.

Is an internet connection required to run SBMLsimulator?

An internet connection is not required for simulations, but for some other operations, like the on-line-update feature.

Where can I get the latest version?

Go to <http://www.cogsys.cs.uni-tuebingen.de/software/SBMLsimulator/>.

Which Java™ version must be installed on my computer to launch SBMLsimulator?

SBMLsimulator requires at least Java™ 1.6. Please see <http://www.java.com/de/download/> to download the latest Java™ version.

Why does SBMLsimulator not start on my Mac with Mac OS X prior to 10.6 Update 3?

If you try to launch SBMLsimulator, but the application does not start and you receive the following error message on the command-line or Java™ console of your Mac, you need to update your Java™ installation:

```
Exception in thread "AWT-EventQueue-0" java.lang.NoClassDefFoundError:
    com/apple/eawt/AboutHandler
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:703)
    ...
```

The interface `com.apple.eawt.AboutHandler` was introduced to Java™ for Mac OS X 10.6

Update 3. If you have an earlier version of Mac OS X or Java™, please update your operating system or Java™ installation. Also see the Mac OS X documentation about the AboutHandler for more information. On a Mac, you can update your Java™ installation through the Software Update menu item in the main Apple menu.

How can I report bugs or get help?

Just contact the mailing list using the following e-mail address: ✉ sbmlsimulator@googlegroups.com.

Appendix A

Acknowledgments

Many people and organizations contributed to the project or funded our work over the years. We would like to thank all those, whose effort has been put into this endeavor.

Funding This work was funded by the Federal Ministry of Education and Research (BMBF, Germany) in the project Virtual Liver Network (grant number 0315756) and the European Commission as part of a Marie Curie International Outgoing Fellowship within the EU 7th Framework Program for Research and Technological Development (project AMBiCon, 332020).

Alumni Marcel Kronfeld¹, Hannes Planatscher², Adrian Schröder¹, Philip Stevens¹, Dieudonné Motsou Wouamba¹, Max Zwießebe¹

Collaborators Benjamin Kandel³, Marcus Klein³, Ute Hofmann³, Maria Thomas³, Nicolas Le Novère⁴,

Contributors Stephanie Hoffmann¹, Mike Cooling⁵, Akira Funahashi⁶, Nicolas Rodriguez⁷, Akito Tabira⁶, Noriko Hiroi⁶ Michael J. Ziller⁸

Special thanks We like to particularly thank Bernhard Ø. Palsson⁹ and his research group for supporting this work. For friendly help we thank Beky Kotcon¹⁰, Samantha Mesuro¹⁰, Daniel Rozenfeld¹⁰, Anak Yodpinyanee¹⁰, Andres Perez¹⁰, Eric Doi¹⁰, Richard Mehlinger¹⁰, Steven Ehrlich¹⁰, Martin Hunt¹⁰, George Tucker¹⁰, Peter Scherpelz¹⁰, Aaron Becker¹⁰, Eric Harley¹⁰, and Chris Moore¹⁰. Thanks for support also to the former team from the Center for Bioinformatics Tuebingen (ZBIT): Finja Wrzodek¹, Florian Mittag¹, Sebastian Nagel¹, and Clemens Wrzodek¹.

¹University of Tuebingen, Germany

²Naturwissenschaftliches und Medizinisches Institut (NMI) der Universität Tübingen, Reutlingen, Germany

³Dr. Margarethe Fischer-Bosch-Institute for Clinical Pharmacology (IKP), Stuttgart, Germany

⁴The Babraham Institute, Cambridge, UK

⁵Auckland Bioengineering Institute, University of Auckland, New Zealand

⁶Keio University Graduate School of Science and Technology, Yokohama, Japan

⁷European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, United Kingdom

⁸Department of Stem Cell and Regenerative Biology, Harvard University, Cambridge, Massachusetts, United States

⁹Department of Bioengineering, University of California, San Diego, La Jolla, CA, United States

¹⁰Harvey Mudd College, Claremont, California, United States

Appendix B

Release notes

B.1 Version 1.0

This is the first release from March 1st 2012.

B.2 Version 1.1

This version was released at April 25th 2013.

B.2.1 New features

Includes version 1.3 of the Systems Biology Simulation Core Library (SBSCL) (Keller *et al.*, 2013).

B.2.2 Bug fixes

All issues related to the simulation of SBML files that were fixed in the Systems Biology Simulation Core Library (SBSCL).

B.3 Version 1.2

This third release was announced at April 24th 2014.

B.3.1 New features

- ✓ The estimation targets and their ranges can now be saved into a file and loaded again.
- ✓ Spline interpolation can now be used before optimization.
- ✓ The program has been updated and is now based on the new version 1.4 of the Systems Biology Simulation Core Library (SBSCL) and JSBML version 1.0β1.
- ✓ Support for Garuda has been added.

- ✓ Improved support for Mac OS X
- ✓ Program update to Java™ version 1.6

B.3.2 Bug fixes

- ✗ Some errors in the command line mode have been corrected.
- ✗ There was an error in optimization when time point 0 is not given.

B.4 Version 1.2.1

This is a minor bug-fix release from July 18th 2014.

B.4.1 New features

- ✓ Now a new model can be selected to be opened without closing the current model. The active model can still be saved before being closed.

B.4.2 Bug fixes

- ✗ There was an error in the data importer: columns were not correctly skipped, if chosen by the user.

Appendix C

Acronyms

API Application Programming Interface

CSV Character-Separated Value

GUI Graphical User Interface

Id Identifier

JDK Java™ Development Kit

JVM Java™ Virtual Machine

JAR Java™ Archive

LGPL GNU Lesser General Public License

NaN Not a Number

ODE Ordinary Differential Equation

PDF Portable Document Format

RSE Relative Squared Error

SBML Systems Biology Markup Language

SBSCCL Systems Biology Simulation Core Library

UTF Unicode Transformation Format

Bibliography

- Apache Software Foundation (2013). Commons Math: The Apache Commons Mathematics Library. <http://commons.apache.org/proper/commons-math/>. Accessed: 2013-06-24.
- Bucher, J., Riedmaier, S., Schnabel, A., Marcus, K., Vacun, G., Weiss, T. S., Thasler, W. E., Nüssler, A. K., Zanger, U. M., and Reuss, M. (2011). A systems biology approach to dynamic modeling and inter-subject variability of statin pharmacokinetics in human hepatocytes. *BMC Systems Biology*, **5**(1), 66.
- Clerc, M. (2005). *Particle Swarm Optimization*. ISTE Ltd, London, UK.
- Clerc, M. and Kennedy, J. (2002). The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, **6**(1), 58–73.
- Dräger, A., Kronfeld, M., Ziller, M. J., Supper, J., Planatscher, H., Magnus, J. B., Oldiges, M., Kohlbacher, O., and Zell, A. (2009). Modeling metabolic networks in *C. glutamicum*: a comparison of rate laws in combination with various parameter optimization strategies. *BMC Syst Biol*, **3**, 5.
- Dräger, A., Rodriguez, N., Dumousseau, M., Drr, A., Wrzodek, C., Le Novère, N., Zell, A., and Hucka, M. (2011). JSBML: a flexible Java library for working with SBML. *Bioinformatics*, **27**(15), 2167–2168.
- Funahashi, A., Tanimura, N., Morohashi, M., and Kitano, H. (2003). CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BioSilico*, **1**(5), 159–162.
- Ghosh, S., Matsuoka, Y., Asai, Y., Hsin, K.-Y., and Kitano, H. (2011). Software for systems biology: from tools to integrated platforms. *Nature Reviews Genetics*, **12**(12), 821–832.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Cambridge, MA, USA.
- Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J.-H. S., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novère, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff,

- J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J. M., Wang, J., and the rest of the SBML Forum (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**(4), 524–531.
- Keller, R., Dörr, A., Tabira, A., Funahashi, A., Ziller, M. J., Adams, R., Rodriguez, N., Le Novère, N., Hiroi, N., Planatscher, H., Zell, A., and Dräger, A. (2013). The systems biology simulation core algorithm. *BMC Syst Biol*.
- Kronfeld, M., Dräger, A., Aschoff, M., and Zell, A. (2009). On the Benefits of Multimodal Optimization for Metabolic Network Modeling. In I. Grosse, S. Neumann, S. Posch, F. Schreiber, and P. Stadler, editors, *German Conference on Bioinformatics (GCB 2009)*, volume P-157 of *Lecture Notes in Informatics*, pages 191–200, Halle (Saale), Germany. German Informatics society.
- Kronfeld, M., Planatscher, H., and Zell, A. (2010). The EvA2 Optimization Framework. (6073), 247–250.
- Le Novère, N., Bornstein, B. J., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J. L., and Hucka, M. (2006). BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res*, **34**, D689–D691.
- Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical Recipes in Fortran: The Art of Scientific Computing*. Cambridge University Press, 2 edition.
- Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, Stuttgart.
- Schwefel, H.-P. (1975). *Evolutionsstrategie und numerische Optimierung*. Dr.-Ing. Thesis, Technical University of Berlin, Department of Process Engineering.
- Storn, R. (1996). On the Usage of Differential Evolution for Function Optimization. In *1996 Biennial Conference of the North American Fuzzy Information Processing Society*, pages 519–523, Berkeley, CA, USA. IEEE, New York, USA.

Index

Symbols

Application Programming Interface (API), 1

Character-Separated Value (CSV), 1, 4, 11,
13, 15, 16, 20, 25

Graphical User Interface (GUI), 1, 20, 24

Java™ Archive (JAR), 2, 3

Java™ Development Kit (JDK), 2

Java™ Virtual Machine (JVM), 2, 3

Ordinary Differential Equation (ODE), 1, 21

Portable Document Format (PDF), 9

Relative Squared Error (RSE), 24

Systems Biology Simulation Core Library (SBSCL),
1, 27, 31

Mac OS X, 2, 4, 9–11, 27–29, 31

MS Windows, 2, 4, 9–11

S

SBML, 1, 4, 8, 13, 20–22, 31

E

EvA2, 1, 14, 15, 17, 19, 27

Excel, 11

G

Garuda, 1, 26, 31

J

Java™, 1, 2, 28, 29, 31

Java™ Web Start, 1

JSBML, 1, 27, 31

L

Language pack

English, 3

German, 3

O

Operating System, 2–4, 12, 29

Linux, 2, 4, 9–11