StochKit2.0 Manual

This document provides installation and usage instructions for StochKit version 2.0.*.

Table of Contents

Introduction

StochKit2 is the first major update to the popular StochKit software package. StochKit2 provides command-line executables for running stochastic simulations using variants of Gillespie's Stochastic Simulation Algorithm and Tau-leaping. Enhancements in StochKit2 include:

- Improved solvers including efficient implementations of the SSA Direct Method, Optimized Direct Method [Cao et al. 2004], Logarithmic Direct Method, a Constant-Time Algorithm [Slepoy et al. 2008], and an Adaptive Explicit Tau-leaping method
- Automatic parallelism
- Event-handling for the Direct SSA
- An updated command-line interface that is more user-friendly and provides more options
- Improved support for extending StochKit functionality

StochKit2 is intended for two audiences: 1) practicing scientists who wish to study the behavior of their biochemical models by running stochastic simulations, and 2) software developers who wish to extend the functionality of StochKit or incorporate StochKit into their software.

This document covers basic usage, targeting audience 1. Developers or those wishing to extend StochKit functionality should also read StochKit2_developers_guide.pdf.

Since StochKit2.0 is command-line driven, Linux/Unix and Mac OS X users must be able to access a terminal and should know basic Unix commands (e.g. "cd", "ls") and how to create environment variables for their shell (e.g. bash). There are plenty of good tutorials on the web. Windows users will have to be able to launch the Visual Studio Command Prompt (StochKit Windows full version, recommended) or the command interpreter, cmd.exe, (StochKit Windows lite version) and know basic navigation commands (e.g. "cd","dir").

The StochKit2 Windows "lite" version does not support customized propensity functions or events.

This document uses Unix notation. Note there are slight changes for the Windows version, such as backslashes ("\") in file paths instead of forward slashes. Also, paths with spaces must be surrounded in double quotes in Windows (paths with spaces are not allowed in the Linux/Unix/Mac version of StochKit2).

Citing StochKit2

If you use StochKit2 for numerical experiments used in a publication or presentation, please cite the software using the following reference:

Kevin R. Sanft, Sheng Wu, Min Roh, Jin Fu, Rone Kwei Lim, and Linda R. Petzold. StochKit2: software for discrete stochastic simulation of biochemical systems with events. *Bioinformatics* (2011) 27(17): 2457-2458.

Installation

StochKit2 runs on UNIX/Linux, Mac OS X (StochKit2<version number>.tgz) and Windows. There are two Windows versions: the full version (StochKit2<version number>_WINDOWS.zip) and the "lite" version (StochKit2<version number>_WINDOWS_LITE.zip) that has limited functionality. Below is a list of dependencies for the different versions of StochKit, followed by installation instructions for the different operating systems.

Dependencies:

UNIX/Linux, Mac OS X: StochKit2 uses the xml2 library. The library is available at http://xmlsoft.org/. The locations in which StochKit2 expects to find libxml2 are listed below the main StochKit2 directory in .make/makefile.c. Edit the LIBXML* variables with the appropriate paths if they are different on your system. Mac users must have Xcode installed.
Windows: The full Windows version of StochKit2 needs Visual Studio. Instructions for obtaining the free version of Visual Studio ("Visual C++ 2010 Express") are given in the StochKit2 Windows Installation Instructions. StochKit2 for Windows "lite" has no dependencies.

UNIX/Linux and Mac OS X:

1. Download StochKit2<version number>.tgz from SourceForge: http://sourceforge.net/projects/stochkit/
   where <version number> is the current version of StochKit2 (e.g. StochKit2.0.0.tgz).
2. Unzip the .tgz file to create the StochKit2<version number> directory (this can be done by typing the following at a terminal: tar xvzf StochKit2<version number>.tgz)

3. Navigate to the main StochKit directory
4. Using a terminal, run: ./install.sh
5. If everything compiled correctly, you're ready to run a simulation (see next section).
6. Optionally, you may install the SBML converter (see "SBML converter" in the "Tools" section below).

Windows:

There are two Windows versions: the full version and the "lite" version that has limited functionality.

To install the full version:

1. Download StochKit2<version number>._WINDOWS.zip from SourceForge: http://sourceforge.net/projects/stochkit/
   where <version number> is the current version of StochKit2 (e.g. StochKit2.0.0._WINDOWS.zip).
2. Unzip it to your desired location.
3. Open Installation_instructions.pdf and follow the instructions.

To install the "lite" version:

1. Download StochKit2<version number>._WINDOWS_LITE.zip from SourceForge: http://sourceforge.net/projects/stochkit/
   where <version number> is the current version of StochKit2 (e.g. StochKit2.0.0._WINDOWS_LITE.zip).
2. Unzip it to your desired location.

NOTE: the lite version does not support custom propensity functions or events.

Running a simulation

This document uses UNIX notation. Note there are slight changes for the Windows version, such as backslashes ("\") in file paths instead of forward slashes. Also, paths with spaces must be surrounded in double quotes in Windows (paths with spaces are not allowed in the Linux/Unix/Mac version of StochKit2).

StochKit2 provides two primary simulation drivers: "**ssa**" and "**tau_leaping**".

To run a simulation, open a terminal window (Windows users: open the Visual Studio Command Prompt for the Windows full version or cmd.exe for the Windows lite version) and navigate ("cd") to the StochKit directory and type the following (Linux/Unix/Mac OS X):

```
./<driver name> -m <model file name> -r <number of realizations> -t
<simulation time> <additional optional arguments>
```

For Windows users type:

```
.\<driver name> -m <model file name> -r <number of realizations> -t
<simulation time> <additional optional arguments>
```

Where everything in "< >" brackets is replaced with the appropriate values. To see a list of all required and optional arguments, type:

Linux/Unix/Mac OS X version:

```
./ssa -h
```

Windows versions:

```
.\ssa -h
```

Running your first simulation

From the StochKit directory, Linux/Unix/Mac OS X users enter:

```
./ssa -m models/examples/dimer_decay.xml -t 10 -r 1000
```

Windows users enter (note the backslashes):

```
.\ssa -m models\examples\dimer_decay.xml -t 10 -r 1000
```

This command runs 1000 realizations of the dimer decay model for 10 simulation time units using the "ssa" driver. If error messages are displayed, see the Troubleshooting section below. Otherwise, if no errors are encountered, the following messages should be displayed:

```
StochKit MESSAGE: determining appropriate driver…running
$STOCHKIT_HOME/bin/ssa_odm_small…
StochKit MESSAGE: created output directory "models/examples/dimer_decay_output"…
running simulation…finished (simulation time approx. 5.53562 seconds)
creating statistics output files…
done!
```

The first message indicates that the simulation was run using "ssa_odm_small". The "ssa" driver uses information about the model to try to select the simulation method that will achieve the best performance. In this case, "ssa_odm_small" is a version of the "Optimized Direct Method" that is further optimized for small models.

The second message says that StochKit created a directory for output. The default output directory name is "<model name>_output".

The remaining messages provide simulation status updates and indicate that the simulation started running, finished in approximately 5.5 seconds (simulation time only), created "statistics output files", and finished.

The “statistics output files” mode is the default output option. In this mode, two files are created: means.txt and variances.txt, which contain the means and variances of all model species at the end time. In general, the stats output files will be found in <output directory>/stats/means.txt and <output directory>/stats/variances.txt. In this particular example, these files will be in models/examples/dimer_decay_output/stats/. Opening the means.txt file should reveal a single line such as:

```
10	274.749 365.171 678.337
```

The first number is the time the data was recorded (the simulation end time), followed by the mean population of each species at that time. Of course, your stats file will likely differ slightly due to the randomness of the simulations.

Running your second simulation

The remainder of this document uses the Linux notation, so **Windows users must replace slashes (“/”) with backslashes (“\”) in paths.**

To run a more complicated example, type the following:

```
./ssa -m models/examples/dimer_decay.xml -t 10 -r 10 -i 5 --keep-trajectories -f
```

In this example, we’re running only 10 realizations and we provide some additional arguments:

- “`-i 5`” indicates that data should be kept at 5 evenly-spaced time **intervals**. In this example, data will be stored at time points 0, 2, 4, 6, 8, and 10. NOTE: data is stored at 6 time points when the number of intervals is 5. In the previous example we did not specify the number of intervals so the default value of 0 was chosen; when 0 is chosen, data is kept only at the end time.
- “`--keep-trajectories`” tells the driver to keep trajectory data. This will create an output directory named <output directory>/trajectories that will contain one file for each realization. Use this option with caution since running a large number of realizations and keeping data at a large number of intervals will lead to StochKit producing a large amount of output data.
- “`-f`” short for “`--force`” specifies that we want to overwrite the existing output directory with our new data. If we had omitted this option, we would have received an error message saying

```
StochKit ERROR (StandardDriverUtilities::createOutputDirs): output directory
“models/examples/dimer_decay_output” already exists.
Delete existing directory, use --out-dir to specify a unique directory name, or run with --force
to overwrite.
Simulation terminated.
```

Command-line options

Common options

- "**-m <model name>**" short for --model, specifies the path to the model file. The model file must be in StochKit2 model definition .xml format.
    - Example: ./ssa **-m models/examples/dimer_decay.xml** -t 10 -r 1000
        - specifies the model file "models/examples/dimer_decay.xml"
    - REQUIRED
- "**-t <simulation time>**" short for --time, specifies the length of the simulation in the (arbitrary) time units implied by the propensity functions in the model.
    - Example: ./ssa -m mymodel.xml **-t 10** -r 100
        - specifies running the simulations for 10 time units
    - REQUIRED
- "**-r <number of realizations>**" short for --realizations, specifies the number of realizations to simulate
    - Example: ./ssa -m mymodel.xml -t 10 **-r 100**
        - specifies running 100 realizations
    - REQUIRED
- "**-i <number of intervals>**" short for --intervals, specifies the number of evenly-spaced time intervals in which to keep data.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **-i 5**
        - specifies 5 intervals
        - NOTE: data will be kept at <number of intervals>+1 time points. In the example above, data would be stored at 6 time points: 0, 2, 4, 6, 8, and 10.
        - Default value "0" specifies data will be kept only at the end time.
- "**-f**" short for --force, tells the driver to overwrite the existing output directory.

Output options

- "**--out-dir <output directory name>**" specifies an output directory name instead of the default (<model file name>_output).
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--out-dir mydirectory**
- "**--no-stats**" specifies not to keep statistics data. May be used only with when "--keep-trajectories" or "--keep-histograms" is specified.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--no-stats** --keep-trajectories
- "**--keep-trajectories**" tells the solver to keep data from each individual trajectory. Will create a "trajectories" sub-directory in the output directory and one file for each realization. Use this option only with a small number of realizations and intervals to avoid generating voluminous amounts of output data.
    - Example: ./ssa -m mymodel -t 10 -r 100 **--keep-trajectories**
- "**--keep-histograms**" tells the solver to keep histogram data.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--keep-histograms**
    - See the Tools section below for information on plotting StochKit histogram data in MATLAB.

- "**--bins <number of histogram bins>**" specify the number of bins to include in each histogram. The default is 32. Must be used with --keep-histograms.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 --keep-histograms **--bins 16**
        - specifies 16 bins in each histogram
- "**--species <list of species Id or indices>**" tells the solver to keep data for only a subset of species in the model.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--species S1 S2**
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--species 0 1**
        - both examples keep species "S1" and "S2", assuming that S1 and S2 are listed as the first and second species, respectively, in the SpeciesList in mymodel.xml. NOTE: species indices begin at 0.
- "**--label**" print column labels in stats and trajectories output files.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--label**

Advanced options

- "**--no-recompile**" Models with reactions where Type is "customized" must be compiled into an executable before running a simulation. In the case where the model has previously been run with the same driver, the --no-recompile option tells the solver to use the existing compiled executable. NOTE: this option will lead to errors if used without previously running the same driver on the same (unmodified) model file.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--no-recompile**
- **"--seed <arg>**" specifies a seed for the random number generator. By default the seed is chosen randomly. This option is typically used for debugging purposes. NOTE: using the same seed with different drivers or different options will not produce identical trajectories.
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **--seed 123**
        - seeds the random number generator with "123". If the same command is run again with the same seed, the trajectories generated will be identical.
- "**-p <number of processors>**" short for --processes tells the solver how many processes to use. By default, the driver automatically selects the number of processes to use based on the number of processors on the computer. This option is typically used only on machines where the number of processors cannot be detected automatically (i.e. if you receive the message: "StochKit MESSAGE (ParallelIntervalSimulation()): unable to detect number of processors. Simulation will run on one processor.").
    - Example: ./ssa -m mymodel.xml -t 10 -r 100 **-p 4**
        - specifies to run the simulation using 4 processes
- "**--epsilon <tolerance>**" set the epsilon parameter for the tau_leaping driver. Valid values are between 0 and 1. By default this value is 0.03. See note below.
    - Example: ./tau_leaping -m mymodel.xml -t 10 -r 100 **--epsilon 0.005**
        - specifies a tighter tolerance of 0.005

    - NOTE: The tau_leaping algorithm uses a tau (step size) selection procedure that differs from the one described in Cao et al. 2006. Specifically, the value of "g_i" is taken to be 3 for all i.
- "**--threshold <minimum reactions per leap>**" The tau_leaping driver adaptively switches to the SSA Direct Method when the number of reactions in a single tau-leaping leap step is less than the threshold. By default, this value is 10.
    - Example: ./tau_leaping -m mymodel.xml -t 10 -r 100 --threshold 5
        - specifies switching to SSA when a leap step results in fewer than 5 reactions firing.

Model definition file format

StochKit2 drivers use models in the StochKit2 model definition format. The StochKit2 model definition format uses *tags* to specify the description of a biochemical model in an xml format. NOTE: the StochKit2 model format is similar to, but is not, SBML! StochKit2 does provide an SBML-to-StochKit model converter. See section "SBML converter" below.

The following example is the simplest StochKit2 model definition (from file models/examples/simple1.xml):

```
<Model>
	<NumberOfReactions>1</NumberOfReactions>
	<NumberOfSpecies>1</NumberOfSpecies>
	<ReactionsList>
		<Reaction>
			<Id>R1</Id>
			<Type>mass-action</Type>
			<Rate>1.0</Rate>
			<Reactants>
				<SpeciesReference id="S1" stoichiometry="1"/>
			</Reactants>
		</Reaction>
	</ReactionsList>
	<SpeciesList>
		<Species>
		<Id>S1</Id>
		<InitialPopulation>100</InitialPopulation>
		</Species>
	</SpeciesList>
</Model>
```

The first tag, "<Model>" indicates the beginning of the model definition. The closing tag "</Model>" at the end indicates the end of the model definition. Most xml tags must be matched with a closing tag. The "<NumberOfReactions>" and "<NumberOfSpecies>" tags specify the number of reactions and species, respectively, in the model. The "ReactionsList" tag encloses one or more (in this case only one) "<Reaction>" elements. A Reaction must contain an "<Id>" and a "<Type>". In most cases, a Reaction contains a "<Reactants>" tag and a "<Products>" tag to specify the reactant and product species, respectively. In the above example, however, there are no Products. If the Reaction Type is "mass-action", then the "<Rate>" tag is required. Type "mass-action" refers to an elementary reaction with 0, 1 or 2 Reactants (NOTE: 3rd order reactions are also accepted as valid mass-action reactions but are discouraged since they require the simultaneous collision of 3 molecules. 4th order and higher are not accepted in mass-action reactions. See the section below on Customized propensity function for non-mass action propensities). The Rate is the stochastic reaction rate that is multiplied by the Reactants' populations to calculate the propensity [Gillespie 1977].

The above model definition file describes a model with a single species "S1" and a single reaction "R1". Reaction R1 has one SpeciesReference tag within the Reactants and no Products. (Note the SpeciesReference does not use a closing tag in this example, but instead ends the opening tag with "/>") Since the stoichiometry of the Reactant is 1 and there are no products, Reaction R1 describes species S1 decaying into "nothing".

NOTE: The rate constants in StochKit2 models are interpreted as *stochastic* rate constants. (see Gillespie 1977). For enzymatic reactions where a species is both a reactant and a product, the species should appear in the Reactants and Products to ensure the mass-action propensity is computed correctly.

Creating your own models

The best way to create your own model is to copy and rename an existing model file, then open it in a text editor and modify it to suit your needs. If you have a version of your model in SBML format, the StochKit2 SBML converter may be able to convert all or part of it to the StochKit2 model definition format (see the "SBML converter" section below).

Parameters

The StochKit2 model definition supports (global) parameters. The file models/examples/simple1p.xml demonstrates the use of parameters. Parameters are defined within the ParametersList tag as shown below.

```
…
<ParametersList>
        <Parameter>
                <Id>P1</Id>
                <Expression>1.0</Expression>
        </Parameter>
</ParametersList>
...
```

With parameter P1 defined, it can be used in place of the numerical value, as used in the reaction Rate tag:

```
...
<Rate>P1</Rate
...
```

This change is demonstrated in models/examples/simple1p.xml. The resulting model is effectively identical to that in simple1.xml. NOTE: parameters cannot be functions of the species' populations.

Customized propensity functions

In the simple1.xml example model, the Type tag within Reaction R1 had the value "mass-action". When Type is "mass-action", the propensity function is determined by the Rate and the population(s) of the reactant species [Gillespie 1977]. These "mass-action" reactions are limited to elementary reactions with 0, 1, or 2 Reactants (NOTE: 3rd order reactions are also accepted as valid mass-action reactions but are discouraged since they require the simultaneous collision of 3 molecules.).

StochKit2 allows for non-elementary reactions by specifying a Reaction Type of "customized". When the Reaction Type is "customized", the <PropensityFunction> tag is required. The example below specifies that species "S1" and "S2" react with stochastic rate constant 2.0.

```
...
        <Type>customized</Type>
        <PropensityFunction>2.0*S1*S2</PropensityFunction>
...
```

NOTE: In order for the above PropensityFunction to be valid, species S1 and S2 must be defined and should be listed as Reactants in the Reaction. The expression in the PropensityFunction must be a valid expression composed of arithmetic operators (+,-,*,/), species Id values, Parameters, and C++ functions (e.g. pow, exp). Careless use of customized propensity functions can lead to errors and can cause StochKit2 to crash. "mass-action" is the preferred Reaction Type.

Events

StochKit2 provides support for events. Events are changes in states or parameters that are executed when a specific condition occurs during a simulation. For example, an event may set a species population to zero after a certain time to mimic an experimental condition. Only the ssa driver has support for events (models with events will be simulated using the direct method).

Events are defined in the <Event> tag. All events must be within the <EventsList> tag.

An Event has three required tags: <Id>, <Trigger>, and <Actions>. The Trigger tag must contain a <Type> tag. The valid values of Type are "time-based" and "state-based".

Time-based triggers

```
…
<Trigger>
        <Type>time-based</Type>
        <Value>5.0</Value>
</Trigger>
…
```

The above xml defines a time-based trigger that "fires" when the simulation time reaches 5. The Value of a time-based trigger must be a numerical value.

State-based triggers

```
…
<Trigger>
        <Type>state-based</Type>
        <Value><![CDATA[ S1 == 0 ]]></Value>
</Trigger>
…
```

The above xml defines a state-based trigger that "fires" when the population of species S1 reaches 0. The Value of a state-based trigger is an expression that evaluates to true or false. In this case, the expression is wrapped in the special <![CDATA[…]]> tag. Wrapping the expression in CDATA ensures that the xml parser does not interpret operators such as "<" (less-than) and ">" (greater-than) as part of an xml tag. Valid Value expressions can be made up of species Id, numbers, arithmetic operators ("+", "-", "/", "*") and comparison operators ("<", ">", "<=" (less than or equal), ">=" (greater than or equal), "==" (equal), "!=" (not equal)).

By default, a state-based trigger event will fire at most once during a simulation. However, if the optional Event tag <AllowMultipleFirings> is set to a value of "true", the event will fire every time the trigger function switches from false to true.

Actions

One or more <Action> tags appear within in the <Actions> tag. An Action defines the change that occurs when the Event's Trigger fires. An Action can change a species population or a parameter value.

There are two Action <Type> values for actions that change a species population: SimpleChangeSpeciesPopulation and CustomChangeSpeciesPopulation. The following two actions demonstrate these actions:

```
…
<Action>
        <Type>SimpleChangeSpeciesPopulation</Type>
        <SpeciesReference id=”S1” value=”10000”/>
<Action/>
<Action>
        <Type>CustomChangeSpeciesPopulation</Type>
        <SpeciesReference id=”S1” value=”S1*2”/>
<Action/>
…
```

The first action sets the population of species S1 to 10000, the second action doubles the S1 population. For SimpleChangeSpeciesPopulation, the value in the SpeciesReference tag must be a number. CustomChangeSpeciesPopulation actions can have functions in the value.

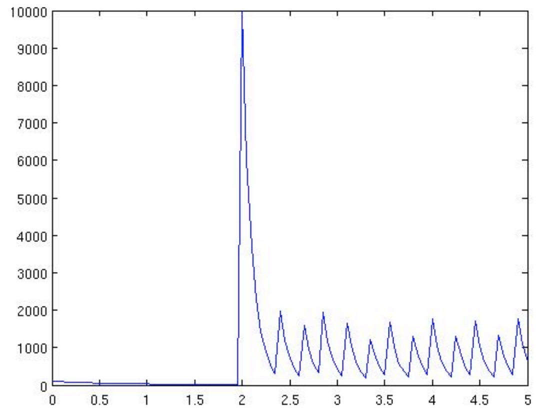
The other <Type> value is ChangeParameter. The following actions demonstrate the two ways to change a parameter:

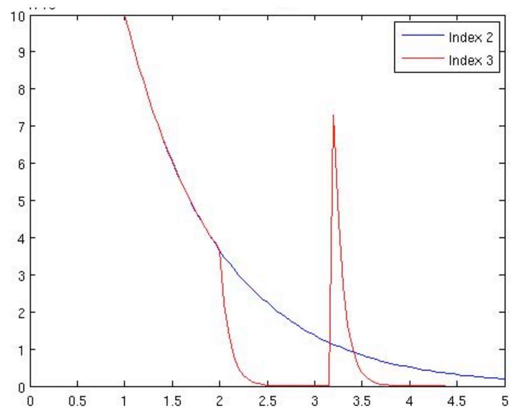
```
…
<Action>
        <Type>ChangeParameter</Type>
        <ParameterReference id=”P2” value=”P1”/>
<Action/>
<Action>
        <Type>ChangeParameter</Type>
        <ParameterReference id=”P3” expression=”P1”/>
<Action/>
…
```

These actions set parameters P2 and P3 to the value of parameter P1. The difference between using the “value” attribute and the “expression” attribute is that when “expression” is used, future changes in a parameter P1 will modify parameter P3. NOTE: an “expression” can not contain the parameter in “id”. Circular references should be avoided.

Events example

The file models/examples/events1.xml provides examples of all the event-related model tags.





The above figures show a sample trajectory for the events1.xml model. The left figure shows a time-based trigger at t=2 that sets the S1 population to 10000, followed by subsequent state-based event when S1==200. Since AllowMultipleFirings is true, the state-based trigger event fires whenever the trigger switches from false to true. The right figure shows that the reaction rates for reactions 2 and 3 are equal after the time-based trigger event at t=1 but they diverge at the event at t=2 because the earlier event that changed "P3" used an "expression" and a parameter in that expression (P1) changed at t=2. The right figure also shows a state-based trigger event where AllowMultipleFirings is false (the default). NOTE: the scale on the right figure is 10*10^4.

Order of event execution and other details

Consider a state-based trigger that fires when S1=100 with an action that sets S1 to 0 and another that fires when S1=0 and sets S1 to 100, with both events having AllowMultipleFirings true. Does this create an infinite loop? This section discusses event-handling details. (By the way, the answer is no.)

The event solver proceeds as follows:

1. The simulation is initialized.
2. It iterates over the state-based trigger events in the order they appear in the model file. If the trigger evaluates to true, the actions fire then it continues to check the next state-based trigger event. It goes through the list exactly once.
3. The next reaction step size is selected, then the time-based trigger events are checked to see if the step size would bring the simulation past a time-based trigger event. If so, the simulation time is set to the next trigger time and the event's action is fired. If there are multiple time-based trigger events that fire at the same time, they are both fired (in random order). If time-based events fire, then the state-based triggers are scanned once as described in 2 and a new step size is then selected.
4. After a step size is selected that does not lead to a time-based trigger event, the next reaction is fired. The list of state-based trigger events is scanned once as

described in 2. The process returns to 3 then 4 and repeats until the end of the simulation.

Additional tags

Comments can be included in a model as follows:

```
<!-- this is a comment -->
```

Most StochKit2 model definition tags allow an optional <Description> tag. For example:

```
<Model>
	<Description>This is a description</Description>
…
```

Tools

SBML converter

The SBML converter is a command-line executable that converts text documents in SBML format into the StochKit2 model definition format. NOTE: not all SBML documents can be converted into the StochKit2 model format. Only a subset of SBML tags are supported by the StochKit2 SBML Converter. Consult the detailed documentation contained in the software distribution for details.

Installing the SBML converter

To install the SBML converter, please refer to the documentation in tools/SBMLconverter/documentation.pdf

Converting an SBML document to the StochKit2 model definition format

To convert an SBML file to the StochKit2 model format, type the following at the command line from the tools/SBMLconverter directory:

```
./bin/sbml2stochkit <sbml file name>.xml
```

Where <sbml file name> is the path to the SBML file with a ".xml" extension. The converter will create a document named <sbml file name>.stochkit.xml. If no errors occurred in the conversion, this *.stochkit.xml file can then be used as the model file input for the StochKit2 drivers.

Optionally, an output file name may be specified as an additional parameter:

```
./bin/sbml2stochkit <sbml file name>.xml <output file name>
```

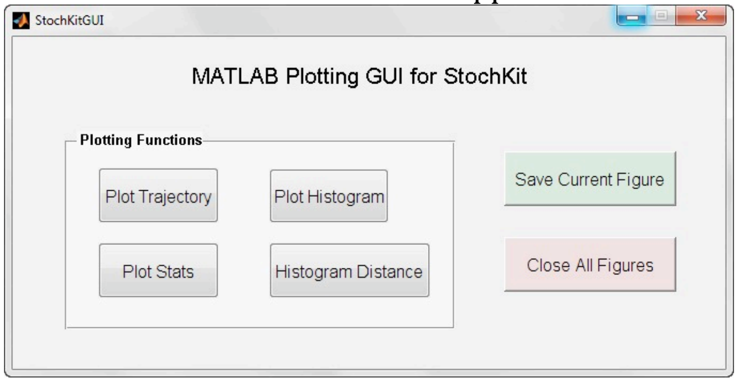
For example, we can convert the SBML file “hsr.xml” in the SBMLconverter directory into a StochKit2-compatible model file named “heat_shock_stochkit.xml” by typing:

```
./bin/sbml2stochkit  hsr.xml  heat_shock_stochkit.xml
```

Plotting tools in MATLAB

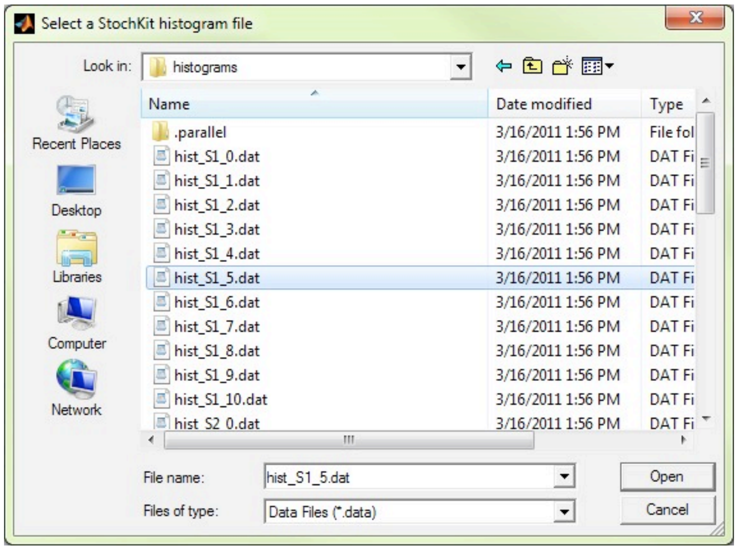
StochKit2 contains four MATLAB-compatible functions for plotting StochKit output files: plotHistogram, histogamDistance, plotStats and plotTrajectories. The first two functions are for visualization and analysis of histograms, and the last two are for plotting statistics and trajectories data. All four functions can be executed in two ways—either by using a MATLAB graphical user interface designed for StochKit or by typing in the MATLAB command-line. The command-line versions are also compatible with Octave.

To open the MATLAB GUI, change directories in MATLAB to tools/MATLAB/. Double click on “StochKitGui.fig” from the Current Directory panel in MATLAB. A separate window titled “StochKitGUI” will appear.



plotHistogram

plotHistogram function can be executed in the MATLAB GUI by clicking on “Plot Histogram” button, which will prompt you for an input data file. Browse and locate the histogram data file to plot.



Name	Date modified	Type
.parallel	3/16/2011 1:56 PM	File fol
hist_S1_0.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_1.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_2.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_3.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_4.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_5.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_6.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_7.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_8.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_9.dat	3/16/2011 1:56 PM	DAT Fi
hist_S1_10.dat	3/16/2011 1:56 PM	DAT Fi
hist_S2_0.dat	3/16/2011 1:56 PM	DAT Fi

File name: hist_S1_5.dat

Files of type: Data Files (*.data)

Once the correct file is chosen, click on the "Open" button to plot a histogram.

To run plotHistogram using the command-line, change directories in MATLAB to tools/MATLAB/. From there, you can plot a histogram data file by typing the following at the MATLAB command-line:

plotHistogram('<path to histogram file>')

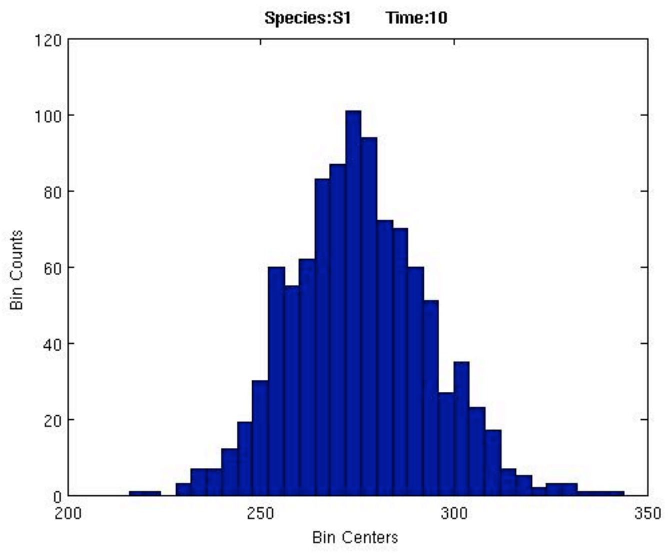
For example, if you previously ran (from the main StochKit directory)

./ssa -m models/examples/dimer_decay.xml -t 10 -r 1000 --keep-histograms -f

You could type (in MATLAB from tools/MATLAB/ directory):

plotHistogram('../../models/examples/dimer_decay_output/histograms/hist_S1_0.dat')

which would generate a figure similar to the one below.



NOTE: the histogram data file format is specified in the StochKit 2 developer guide (StochKit2_developer_guide.pdf).

histogramDistance

histogramDistance function is used to plot two histograms side-by-side (overlapping) and also to calculate both the Euclidean and Manhattan distance between the histograms.

To use the MATLAB GUI for this feature, open "StochKitGUI.fig" and press the "Histogram Distance" button. A pop up window will appear for selection of the two input histogram data files. After selecting two input files, a figure will appear with histograms and a text box containing histogram distance information. NOTE: the histograms will not display correctly if the two histogram data files were generated using different numbers of bins or realizations.

If you want to use the MATLAB command-line, type the following line:
histogramDistance('path to the first histogram file',' path to the second histogram file'),
For example, if you previously ran:

```
./ssa -m models/examples/dimer_decay.xml -t 10 -r 1000 --keep-histograms -f
```

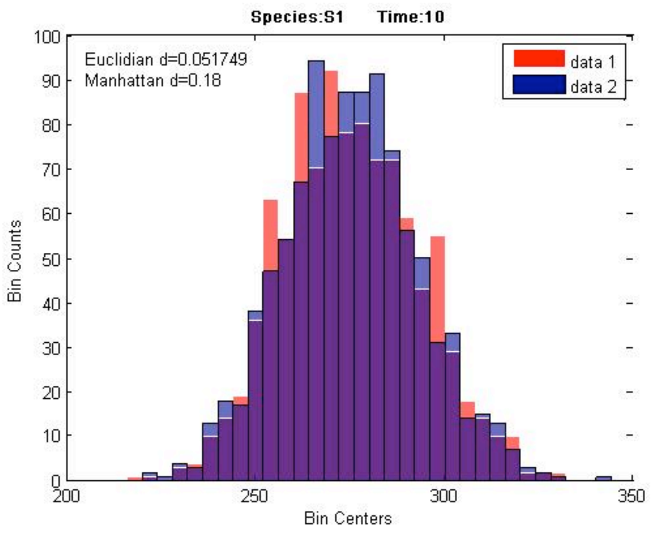
as well as

```
./ssa -m models/examples/dimer_decay.xml -t 10 -r 1000 --keep-histograms -f --out-dir dimer2
```

you could compute the histogram distance by typing:

```
histogramDistance('../../models/examples/dimer_decay_output/histograms/hist_S1_0.dat',
'dimer2/histograms/hist_S1_0.dat')
```

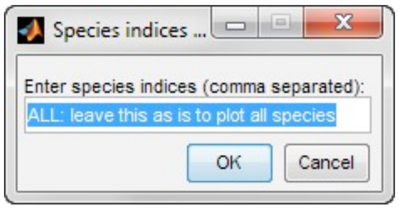
If you entered the paths correctly (and the files exist) the function will generate a figure similar to the one below.



Again, the histograms will only display correctly if the two histogram data files were generated using the same numbers of bins and realizations.

plotStats

To run plotStats using the MATLAB GUI, first open the GUI by clicking on "StochKitGUI.fig" file in tools/MATLAB/ directory. Then click on the "Plot Stats" button. A pop up window will appear for selection of the "stats" directory where "means.txt" and "variances.txt" files are located. After the input directory is chosen, another pop up window will appear asking if the data contains labels (generated using the --label simulation option). Type 'yes' if the data were generated with labels and 'no' otherwise. The final step is selection of species indices to plot. Indices start at 1 and are separated by a comma. For example, typing '1, 5' in the indices field will plot the mean and the variance of the first and fifth species. Duplicate indices will be removed, and the indices input will be rearranged in ascending order before plotting the data. If you want to plot all species, leave the field as is (screen shot below).



Enter species indices (comma separated):

ALL: leave this as is to plot all species

Pressing the "OK" button will generate a plot with the mean population of the selected species as well as a one standard deviation band (for an example, see the next figure).

To use the plotStats function in the MATLAB command-line, type:

```
plotStats ('<path to stats directory>',<column indexes to plot>)
```

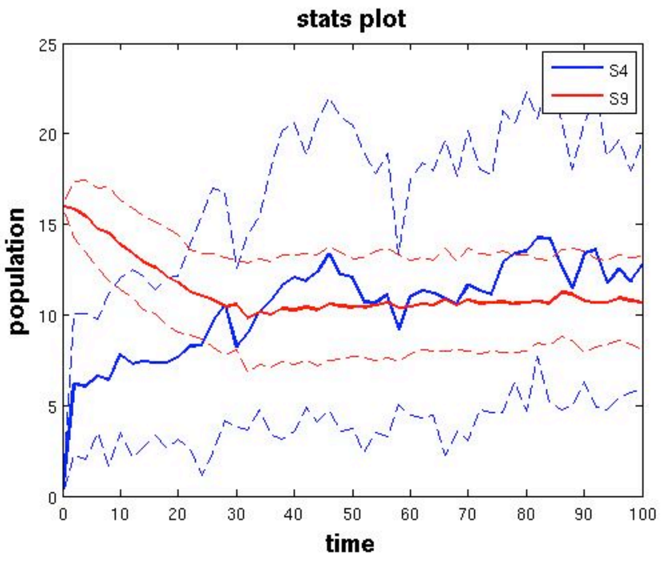
For example, if you previously ran

```
./ssa -m models/examples/heat_shock.xml -t100 -r50 -i50 --keep-trajectories --label
```

You could type:

```
plotStats ('../../models/examples/heat_shock_output/stats',[4 9])
```

which would generate a figure similar to the one below.



The plot shows the means +/- one standard deviation. NOTE: the species indices ([4 9] in this example) begin counting at 1.

plotTrajectories

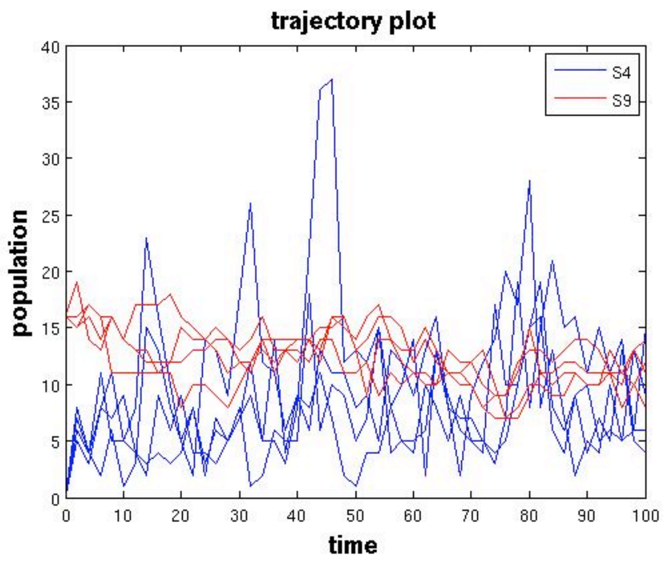
To plot trajectories using the MATLAB GUI, click on "Plot Trajectories" button in the StochKitGUI window. A pop up window will prompt you for input trajectory file(s). For this function, you can select multiple files to be plotted in the same figure as long as the files were generated in the same simulation setting. Browse and locate the trajectory files to plot. For selection of multiple files, use "Shift" or "Control" keys in your keyboard. The rest of the steps, indication of label and selection of species indices, are the same as in plotStats function (see above).

You can also execute plotTrajectories function by typing the following at the MATLAB command-line:

```
plotTrajectories('<path to trajectories directory>',<first trajectory>,<last trajectory>,
[<species indices>])
```

To plot the first 4 trajectories of the same species as the plotStats example, we could type:

```
plotTrajectories('../../models/examples/heat_shock_output/trajectories',0,3,[4 9])
```



NOTE: For plotTrajectories, the 2nd and 3rd arguments are the first and last trajectory number. Trajectory numbers start at 0 and end at (number of trajectories -1).

Creating custom drivers and extending StochKit2

StochKit2 is designed to be easily extended for additional functionality or to include StochKit2 solvers in other software. Please see the StochKit2 Developer Guide included with the StochKit2 distribution.

NOTE: use of StochKit 2 (all versions) is subject to the license agreement. The license was changed in version 2.0.5 (from GPL3 to the BSD 3-clause license).

Troubleshooting

Downloading and Installation

StochKit2 has been installed and tested on a limited selection of Linux, UNIX, Mac OS X, and Windows distributions.

On Linux/UNIX and Mac, if you encounter during compilation an error message about "xml2" or "libxml/*", you need to install the xml2 library. See http://xmlsoft.org/ for details. StochKit2 expects the "include" library to be in /usr/include/libxml2. If you have libxml2 installed in a different location, edit the LIBXML* lines in .make/makefile.c with the appropriate values (note the dot "." in the name of the ".make" directory).

Mac OS 10.7 (Lion) users: Versions of StochKit2 prior to 2.0.4 are not compatible with Mac OS 10.7 (Lion). Those running Lion should upgrade to the latest version of StochKit2. Some users have encountered problems after upgrading OS X to Lion. The StochKit2 installation requires the "g++" C++ compiler, which in some cases might not be in your PATH after upgrading to Lion. Ensure that "g++" points to a valid C++ compiler prior to installing StochKit2. If you encounter other problems trying to install StochKit2 on Lion, please contact the StochKit2 Team.

If you encounter other problems installing StochKit2 on your distribution, please contact the StochKit2 Team at: StochKit@cs.ucsb.edu. Please include the error message you received, the operating system, and the version of StochKit you are using.

Running a simulation

In the Windows full version of StochKit2, if you are using the command interpreter (cmd.exe) instead of the Visual Studio Command Prompt, you have to set the path to "msbuild" in the "path" environment variable as described in the Windows installation instructions (Installation_instructions.pdf).

Most simulation problems are due to errors in the model definition file. Attempts were made to display meaningful error messages but not all errors can be caught by the software. Problems in customized propensity functions and events are particularly difficult to detect.

Some bugs in the model definition file can be detected by running the "single_trajectory_debug" custom driver.

> **Linux/UNIX/Mac OS X**: Navigate to custom_drivers/single_trajectory/. Then run: ./single_trajectory_debug -m <your model file> -t <simulation time>.
> **Windows (full version only)**: In Windows Explorer, navigate to the custom_drivers folder. Double click on custom_drivers.sln. Set to "Release" mode and build the solution. When the compilation completes, navigate on the Visual Studio command line to the StochKit "bin\" directory. Then run: .\single_trajectory_debug -m <your model file> -t <simulation time>.

Hopefully, the program will detect a problem (e.g. a negative population or a propensity that equals "infinity") which you can then use to fix your model file. The single_trajectory_debug driver may not discover all bugs and does not work with models that include events.

See below for a list of common simulation error messages. If you believe you have found a bug in the StochKit2 software, please email the StochKit2 Team at: StochKit@cs.ucsb.edu. Please include the error message you received, your operating system, and the version of StochKit you are using.

Common error messages

- StochKit ERROR (CommandLineInterface::parse): missing required parameter(s). Run with --help for a list of required and optional parameters.
    - At least one of the required command-line parameters is missing. The required form is: ./<driver name> -m <model file name> -t <simulation time> -r <number of realizations> <additional optional arguments>
- StochKit ERROR (StandardDriverUtilities::createOutputDirs): output directory "<output directory name> " already exists. Delete existing directory, use --out-dir to specify a unique directory name, or run with --force to overwrite.
    - StochKit2 drivers create an output directory when they are run. By default, StochKit2 won't let you overwrite an existing directory.
- StochKit ERROR (StandardDriverUtilities::createOutputDirs): error creating output directory.
    - This often happens if the output directory already exists and a file within that directory is open in another program. This creates a filesystem lock file so the directory cannot be deleted. Quit the program that is using the file or specify a different output directory using the "--out-dir" option.
- StochKit ERROR (CommandLineInterface::parse): unable to parse command-line arguments. Run with --help for a list of required and optional parameters.
    - The entered command contains an unrecognized string. Usually occurs due to a typo when entering the command.
- ssa: Command not found.

    - Cause: forgot to put "./" before the command or installation/compilation failed.
- StochKit crashes and the log contains something like: "boost::exponential_distribution<RealType>::exponential_distribution(RealType) [with RealType = double]: Assertion `_lambda > result_type(0)' failed." This is usually caused by a propensity sum that is "not a number", due to a propensity that evaluates to infinity (e.g. a custom propensity function that divides by zero) or similar error. Try running the single_trajectory_debug custom driver as described above.
- StochKit MESSAGE: compiling generated code...this will take a few moments... 'msbuild' is not recognized as an internal or external command, operable program or batch file.
  StochKit ERROR: compile of generated code failed. Simulation terminated. Check log file "<path to generatedCode >\compile-log.txt" for error messages. This occurs in the Windows version and is caused by using the command interpreter (cmd.exe) instead of the Visual Studio Command Prompt and not having the 'msbuild' compiler in your path. Use the Visual Studio Command Prompt or correctly build the windows version by closely following the Windows installation instructions (Installation_instructions.pdf). NOTE: the file compile-log.txt will not exist because the compile command was not executed.

SBML converter

For troubleshooting the SBML converter, see the documentation in tools/SBMLconverter/documentation.pdf.

Plotting tools

The most common errors for plotting are typos in the path or filename or incorrect arguments. The different plotting functions take different arguments—check the examples for the proper form.

Known bugs

A list of known bugs is maintained on sourceforge.net. Visit http://sourceforge.net/projects/stochkit/support and click on "Bugs".

License

Contact information and bug reporting

The StochKit team welcomes your feedback on the software. Please send comments, questions, enhancement ideas, and suspected bugs via email to StochKit@cs.ucsb.edu. To report a bug, please include any error message you receive, the operating system and version of StochKit you are using.

References

1. Cao, Y., Li, H., and Petzold, L. *J. Chem. Phys*. **121**, 4059 (2004).

2. Cao, Y., Gillespie, D.T., and Petzold, L.R. *J. Chem. Phys*. **124**, 044109 (2006).

3. Gillespie, D.T. *J. Phys. Chem*. 81, 25, 2340 (1977).

4. Gillespie, D.T. *J. Chem. Phys*. **115**, 1716 (2001).

5. Slepoy, A., Thompson, A.P., and Plimpton, S.J. J. Chem. Phys. **128**, 205101 (2008).

Glossary of terms

$STOCHKIT_HOME: The directory in which StochKit2 is installed. After installation/compilation, this directory contains the "ssa" and "tau_leaping" executables. The Linux/Unix/Mac version sets an environment named STOCHKIT_HOME to the path to the main StochKit directory when the ssa or tau_leaping driver is run.

Approximate methods: Methods that generate approximate trajectories of the chemical master equation. Tau-leaping is an example of an approximate method. Contrast with "Exact methods".

Command-line: A UNIX or Linux "terminal" window.

Command-line executable: A program that can be run from the command-line. The StochKit2 drivers "ssa" and "tau_leaping" are command-line executables.

Constant-Time method: The composition-rejection algorithm of Slepoy et al. 2008 (see References).

Custom driver: A driver written by a user of StochKit2 or a developer that is not a member of the StochKit2 Team. The default drivers included in StochKit2 may not meet the needs of all users. For that reason, StochKit2 is meant to be easily extended. Those wishing to extend StochKit2 or create custom drivers should consult the StochKit2 Developer's Guide included with the StochKit2 distribution.

Customized propensities: Contrast with mass-action propensities. StochKit2 allows an arbitrary function to be used as a propensity function. For example, if parameters "Vmax" and "Km" are defined, a customized propensity function could define the stochastic Michaelis-Menten propensity: Vmax*S/(Km+S), where S is the name of the substrate species. Most customized propensities, if not describing elementary reaction, are not justified by theory and may lead to simulation results that do not describe any real physical system. Customized propensities can also crash StochKit2 if not used properly. For example, the customized propensity 1.0*S1*S1, which might be used to describe a propensity for a dimerization reaction is incorrect. The correct propensity function is: 1.0*S1*(S1-1)/2 (see Gillespie 1976 or 1977). The previous incorrect propensity function would return 1.0 when the population of S1 is 1. If this dimerization reaction removed two S1 molecules, this would lead to a negative population and likely crash StochKit. Customized propensities are to be used with caution!

Daniel T. Gillespie: Author and co-author of several influential papers in the field of stochastic simulation of biochemical reaction networks. The Stochastic Simulation Algorithm (SSA) was described by Gillespie in his 1976 and 1977 papers (see References) and is often called the "Gillespie algorithm". Gillespie also published the first tau-leaping paper in YEAR.

Driver: A command-line executable that runs a simulation method. StochKit2 has two primary drivers: "ssa" and "tau_leaping". The driver calls an underlying solver.

Elementary reaction: A reaction with 0, 1, or 2 reactants. Valid elementary reactions include synthesis reactions: <nothing> --> products; isomerization reactions: S_i --> products; dimerization reactions: S_i+S_i --> products; and bimolecular reactions: S_i+S_j --> products. See Gillespie 1977 for details. NOTE: reactions with three reactants are not considered elementary reactions.

Ensemble: More than one trajectory (i.e. multiple realizations).

Events: Discrete changes in state or model parameters.

Event-handling: Models with Events can only be run using a driver with event-handling enabled. The StochKit2 "ssa" driver does feature event-handling, though it will only use the direct method (event-handling is not enabled on tau_leaping or any of the other underlying SSA algorithms).

Exact methods: Methods such as the SSA Direct Method that generate exact trajectories of the chemical master equation. See Gillespie 1977. Contrast with approximate methods such as Tau-leaping.

Firing a reaction: The process of modifying the simulation system's current population state. In the solver, this is done by adding the reaction's stoichiometry vector to the current population vector. The solver also updates the propensities and other solver data structures.

Gillespie: See Daniel T. Gillespie

Gillespie algorithm: The Stochastic Simulation Algorithm (SSA) (see Gillespie 1977).

Intervals: The standard StochKit2 drivers ("ssa" and "tau_leaping") take a number of intervals as a parameter. The simulation time is divided by the number of intervals and output is recorded at the interval boundaries. For example, running a standard driver simulation with options "-i 5" and "-t 10" will run simulations for 10 time units, and will store output at time 0, 2, 4, 6, 8, and 10. NOTE: data is stored at (itervals+1) time points. By default, intervals=0 and data is stored only at the simulation end time.

Mass-Action Propensities: Terminology used in this documentation to refer to the values of the propensities of elementary reactions. Valid "mass-action" reactions have reactant stoichiometries of 0, 1, or 2. Reactions with 3 reactants are not considered elementary as they imply simultaneous collision of three molecules and should be modeled as two separate reactions (alternatively, a customized propensity could be used to model such a reaction in StochKit2).

Mixed model: Term used to describe a StochKit2 model with at least one customized propensity function. The customized propensity functions must be converted into C++ code and compiled before a simulation is executed.

Optimized Direct Method: Algorithm from Cao et al. 2004 (see References).

Product: The species that are produced from a given chemical reaction. Products - reactants determines a reaction's stoichiometry. It is possible for a species to be both a reactant and a product in a given reactant.

Propensity: The value of the propensity function.

Propensity function: Function of state that determines the probability of a reaction firing. The propensity functions for elementary reactions are given by the following formulas: synthesis reactions: <rate constant>; unimolecular reactions: <rate constant>*<population of the reactant species>; dimerization reactions: <rate constant>*<population of reactant species>*<population of reactant species -1>/2; bimolecular reactions: <rate constant>*<population of first reactant species>*<population of second reactant species>.

Rate constant: The reaction rate parameter that is multiplied by the populations to calculate the propensity. StochKit2 interprets rate constants as stochastic rate constants. NOTE: the propensity for a dimerization reaction is c*S*(S-1)/2 where c is the (stochastic) rate constant and S is the population of the reactant species.

Reactant: The species that are consumed in a chemical reaction. Products - reactants determines a reaction's stoichiometry. It is possible for a species to be both a reactant and a product in a given reactant.

Realization: A single simulation trajectory. Multiple realizations form an ensemble.

Solver: The underlying C++ implementation of an algorithm that is called by a driver.

Species: Chemical species that are the population state variables in a stochastic simulation.

SSA: see Stochastic Simulation Algorithm.

Statistics data: Species population means and variances data from a simulation. The default output option in StochKit2 is to keep stats data. The MATLAB-compatible function plotStats plots these data, though it plots the standard deviation rather than the variance (see Tools section of this manual).

Stats data: see Statistics data.

Stochastic rate constant: See Rate constant. Related to the deterministic rate constant for the reaction rate equations, except scaled by the system volume (directly or inversely or unscaled depending on whether the reaction is a synthesis, bimolecular, or unimolecular, respectively). See Gillespie 1977.

Stochastic Simulation Algorithm: The monte carlo algorithm for generating exact trajectories of the chemical master equation. See Gillespie 1977. Several variants of the Stochastic Simulation Algorithm exist. The StochKit2 driver "ssa" runs a variant of the Stochastic Simulation Algorithm (the particular variant is determined by the model characteristics).

StochKit2: Term describing all versions of the first major upgrade to the original StochKit software package. Term may be used to describe particular releases: StochKit2.0.0, StochKit2.0.1, etc. and all future StochKit2.* releases.

StochKit2 model definition format: The xml format for models passed to StochKit2 drivers.

StochKit2 Team: Professor Linda Petzold and her research group at the University of California, Santa Barbara. The following people contributed to the development of the original StochKit2.0 release: Kevin Sanft, Sheng Wu, Min Roh, Jin Fu, Rone Lim, Hong Li, and others.

StochKit directory: the main StochKit directory is the location of the unzipped "StochKit2<version>" or "StochKit2<version_WINDOWS" (or _LITE) directory.

StochKitML: Another name for the StochKit2 model definition format.

StochKit website: http://engineering.ucsb.edu/~cse/StochKit/

Stoichiometry: For a single reaction, the stoichiometry is the state change vector. For a model, the stoichiometry matrix is a matrix comprised of the single-reaction stoichiometry vectors.

Synthesis reaction: Reaction with no reactants. The product is modeled as being produced from "nothing".

Tau-leaping: Algorithm for generating approximate trajectories of the chemical master equation. See Gillespie 2001 for the original tau-leaping description. Since Gillespie's 2001 paper, several variants of tau-leaping have been devised. The StochKit2 driver "tau_leaping" implements a tau-leaping variant.

Trajectory: The path of a single stochastic simulation. Also known as a realization.