Stochkit 2.1 Manual

The Petzold Group

May 27, 2015

This document provides installation and useage instructions for StochKit 2.1.⋆.

Contents

List of Figures

List of Tables

1 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 (** denotes new or improved features in StochKit2.1):

- Improved solvers including efficient implementations of the SSA Direct Method (DM), Optimized Direct Method (ODM) [1], Next Reaction Method (NRM)** [2], a Constant-Time Algorithm (ConstantTime) [3], and an Adaptive Explicit Tau-leaping method
- Both automatic/manual method selection for SSA, with a calibrator tool for method speed comparison**
- Automatic parallelism
- Event-handling for all SSA methods**
- An updated command-line interface that is more user-friendly and provides more options
- A SBML converter
- Various output selection and MATLAB/Octave visualization tool
- Improved support for extending StochKit functionality**
- MPI support for cluster use**

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 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).

1.1 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.

2 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.

2.1 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.

2.2 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. StochKit2.1.0.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).

2.3 MPI

Follow the steps described in the installation for UNIX/Linux except step 4. In step 4, instead of running ./install.sh, run: ./install.sh MPI

2.4 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. StochKit2.1.0._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. StochKit2.1.0._WINDOWS_LITE.zip).
2. Unzip it to your desired location.

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

3 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
```

3.1 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 MESSAGE: Simulating using ODM solver...
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 the Optimized Direct Method (ODM) solver. The "ssa" driver uses information about the model to try to select the simulation method that will achieve the best performance. Note that this is only an educated guess without simulating the model. For better selection, user can use the Calibrator tool to compare the actual performance of each method and manually select method with "–method" argument.

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.

3.2 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.
  ```

3.3 Command-line options

3.3.1 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". This argument is **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. This argument is **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. This argument is **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.

3.3.2 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.

- `-keep-user-output` tells the solver to keep user defined output data if it is available. Please refer to developer's manual for details about how to define custom output class. Example:

  ```
  ./ssa_with_custom_output -m mymodel.xml -t 10 -r 100
      --keep-user-output
  ```

- `-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
  ```

3.3.3 Advanced options

- `-method <name of method>` specifies the method to simulate. For SSA simulation, user can choose from: DM, ODM, LDM, ConstantTime, NRM. For Tau-leaping simulation, user can choose from AdaptiveExplicit. Example:

  ```
  ./ssa -m mymodel.xml -t 10 -r 100 --method ODM
  ```

  specifies ODM as the simulation method.

- `-calibrate` If this options is specified, StochKit2 will use the Calibrator to compare performance among different methods. Currently only supported in SSA. Example:

  ```
  ./ssa -m mymodel.xml -t 10 -r 1 --calibrate
  ```

  uses the calibrator instead of simulating.

- `-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 [1]. Specifically, the value of "gi" 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.

3.4 Running MPI Simulation

The arguments to the program are the same as in the single machine use. To run a simulation, the program needs to be executed by mpirun or the equivalent. Below are some examples:

```
mpirun --np 8 ./ssa -m models/examples/dimer_decay.xml -t 10 -r 10 -f
mpiexec.hydra --rmk pbs ./ssa -m models/examples/dimer_decay.xml -t 10
    -r 10 -f
mpirun --np $(grep node $PBS_NODEFILE | wc -l) --machinefile $PBS_NODEFILE
    ./ssa -m models/examples/dimer_decay.xml -t 10 -r 10 -f
```

Consult your cluster documentation for more detailed information about running MPI jobs and requesting resources.

4 Model definition file format

4.1 Creating your own models

4.2 Parameters

4.3 Custom propensity functions

4.4 Events

4.4.1 Time-based triggers

4.4.2 State-based triggers

4.4.3 Actions

4.4.4 Events example

4.4.5 Order of event execution and other details

4.5 Additional tags

5 Tools

5.1 SBML Converter

5.1.1 Installing the SBML converter

5.1.2 Converting an SBML document to StochKit2

5.2 Calibrator

5.3 Plotting in MATLAB

5.3.1 plotHistogram

5.3.2 histogramDistance

5.3.3 plotStats

5.3.4 plotTrajectories

6 Creating custom drivers and extending StochKit2

7 Troubleshooting

7.1 Downloading and installation

7.2 Running a simulation

7.2.1 Common error messages

7.3 SBML converter

7.4 Plotting tools

8 Known bugs

9 License

10 Contact information and bug reporting

11 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 [3].
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 [4]). 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 and is often called the Gillespie algorithm. Gillespie also published the first tau-leaping paper in 2001 [5].
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: Si –> products; dimerization reactions: Si+Si –> products; and bimolecular reactions: Si+Sj –> products. See [4] 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 [4]. 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 [1].
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 reaction 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. 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.

References

[1] Y. Cao, H. Li, and L. Petzold, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," *J. Chem. Phys.*, vol. 121, no. 9, pp. 4059–4067, 2004.

[2] M. a. Gibson and J. Bruck, "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels," *J. Phys. Chem. A*, vol. 104, no. 9, pp. 1876–1889, 2000.

[3] A. Slepoy, A. P. Thompson, and S. J. Plimpton, "A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks," *J. Chem. Phys.*, vol. 128, no. 20, 2008.

[4] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *J. Phys. Chem.*, vol. 81, pp. 2340–2361, Dec. 1977.

[5] D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *J. Chem. Phys.*, vol. 115, no. 4, pp. 1716–1733, 2001.