Stode - Stochastic Simulations of Differential Equations

Carel van Gend
European Media Lab GmbH
Heidelberg, Germany

August 8, 2001

Abstract

This document describes the differential equation parsing and stochastic simulation program, stode, and its graphical front-end, stodeq. Installation instructions are given, and the creation, implementation and simulation of models is discussed.

Contents

1 Why Stochastic Simulations?

The usual approach to simulating a biochemical system is to make a model which approximates that system, and then to solve the set of differential equations which comprise the model. There are many available algorithms which enable one to solve a set of differential equations numerically.

In many situations of biochemical interest, however, one is dealing with situations where there are only a few reactants of some species present. In these cases, the continuity assumption of the differential equation approach breaks down, and a numerical solution to the set of differential equations may lead to misleading results. An alternate approach is to perform a stochastic simulation of the system, whereby each reaction taking place is simulated, using Monte-Carlo methods. Following this approach is valid even when the reactant populations are very low, and indeed, one may see effects which are not apparent using the more traditional approach.

Very often, nevertheless, it is convenient to formulate the model in terms of differential equations, and it these which biochemists are often used to working with. The program described below takes a set of differential equations as input, extracts all necessary information, and performs a stochastic simulation of the system.

2 Program Description

The program is separated logically into two main tasks. The first consists of reading and processing the differential equations, while the second consists of running the simulation. For the first part, stode takes as input a text file specifying the set of differential equations to be parsed, as well as the rate constants, reactants (with their initial concentrations) and other constants needed to determine the system. This file is lexically analysed into tokens and fed to a grammar which instantiates and assigns values to the various variables representing the reactants, constants and differential equations. Following this, stode reconstructs the reactions associated with the differential equations, and if necessary calculates the actual particle numbers from the given reactant concentrations.

The second task is that of performing the simulation. This is done according to Gillespie's algorithm, whereby at each step, the probability of each reaction being the next is calculated. This is proportional to the number of participating reactants and the rate constant for that reaction. Two random numbers are then generated, which in association with these probabilities, allow the next reaction and the time at which it occurs to be chosen. The time and particle numbers are then updated, and the probabilities for the next step calculated.

The simulation proceeds in this way until a specified number of steps have been completed, or a specified total time has been reached. During the course of execution, the particle numbers and time for each step are written to a file.

The program may be run in stand-alone mode, or with a GUI front-end. Generally the latter is somewhat slower, as the GUI has some overhead, but allows the progress of the simulation to be viewed in real time, and allows fine-tweaking of the various parameters. For very long runs, the user will probably prefer to use the stand-alone version.

When the graphical user interface front end is invoked, the user may specify the file to be used as input, or may enter or modify the relevant data in the supplied editor window. The results of the processed input may then be examined before the simulation is begun. As mentioned above, the simulation results are displayed in real time in a plotting window, which in addition may be set to display results over the whole time frame or for a specifiable number of the most recent steps. As was the case for the stand-alone version, the results are written to a file as it is run. For both versions, a gnuplot command file is generated before exiting. This allows easy viewing of the results with gnuplot.

3 Installation

The following assumes that you have the compiled versions of the program. If you prefer, you can also obtain the source and build it yourself.

3.1 UNIX instructions:

The Unix version comes in two forms - one linked dynamically against the Qt libraries, and one linked statically. The latter is of course somewhat bigger, so if you already have a copy of the Qt run-time libraries installed, then you would probably prefer the dynamically linked version.

If you haven't already done so, unzip and untar the package. The command "tar -xvzf stode-x.y.tar.gz" should do the trick (substitute x.y with the version number as appropriate).

The executables need to be copied somewhere where the PATH environment is set. Usually /usr/local/bin is a good choice (make sure it's in your path). The executables should have execute permission (the command "chmod 755 file" will do the trick). The run control files should be copied to your home directory (as .stoderc and .stodeqrc) or to /etc (as stoderc and stodeqrc). A directory in which to store the data files should be created. A directory "stode" under your home directory is a good choice for this, since this is where the executables will look for them if they are not in the directory from which the executables are run and if the full path to the data files is not given.

All of the above can be done manually, or the bash script "install.sh" will do all of that for you. This allows a local, global or hybrid local/global installation, the latter probably being what most users want.

After installation, edit the run control and sample data files to suit your needs.

3.2 Windows instructions:

Create a directory called "stode" under the root directory (C:\). Copy all the files to that directory, except for the .ini files, which should be copied to the root directory. Edit the run control (*.ini) and sample data (*.sdi) to suit your needs.

4 Creating a Model

Far be it for me to tell you, o esteemed user, how to create a model of your system of interest. You probably already have several models you'd like to try out. All that's necessary is that the model be expressed in terms of a set of differential equations, and that you specify the initial conditions. There are nevertheless some points which should be borne in mind:

Firstly, if your model involves terms with more complicated kinetics, which replace several explicit terms, you might consider using the full set instead. For example, enzyme catalyzed reactions can under certain conditions be described by the Michaelis-Menten approximation, greatly simplifying the reaction set description. The conditions for applicability of this, however, are that the substrate population be significantly greater than the enzyme population, and that the intermediate enzyme-substrate complex be in a quasi-steady state for most of the reaction course. If the populations of the reactants in the system are low, then it is possible that these conditions do not hold. Thus, it may be necessary to include explicitly the reactions for the creation and decay of the enzyme-substrate complex.

Secondly, the terms in each differential equation must correspond to reactions in the model. Thus, if you require an oscillatory solution, and you impose this with an explicit oscillatory term, there is probably no easy way to translate this into a stochastic simulation. Remember that the algorithm works by extracting from the differential equations information pertaining to the actual reactions.

5 Creating the Input File

An input file is necessary to specify the system. Normally this is a text file read by the program, although in GUI mode, the input file may be entered directly in the edit window. Several example files are distributed with the program.

This file must contain the differential equations, any other equations specifying relationships between the reactants, the initial reactant concentrations, all reaction rate constants and any other constants necessary. Included among the latter should be the volume occupied by the system, to enable reactant concentrations to be converted to particle numbers. The volume should be given in multiples of litres. The concentration units should be some multiple of molar (mols/litre); the default is $1.0e^{-9}$ (i.e. if the concentration is listed as 40, that means it is 40 nanomolar), but this may be overridden by a command line or initialisation file option.

The parser ignores blank and comment lines. The latter are indicated by beginning the line with a "#"

The parser needs some help in establishing the units of each term. Factors in terms which contain a numerator and denominator should have both appear within parentheses. Generally these are factors resulting from the dynamics of the reaction. The user should thus ensure that the product of the rate constant and all remaining unparenthesized reactants has the correct units, which should be some multiple of molar per second. However, if the input is correct and each factor has the correct units, this should be achieved automatically.

Although the order of individual lines in the file is irrelevant, it is vital to use the correct type of assignment operator (=, :=, –> etc.) on each line. The symbol = is used for differential equations, other equations and constant assignment. The symbol := is used for assigning the cell volume. The symbol –> is used to assign the values of the rate constants.

Important to note is that constants which should have the same units as reactants should be entered as reactants. For example, the constant K_m in Michaelis-Menten kinetics, although derived in terms of the more fundamental reaction rates, should be have its value assigned with the equality sign used to assign reactants. This is easily understood if one notes that it appears in terms such as $[S]/(K_m + [S])$, and thus should scale like the reactant $[S]$. The reason for this (possibly confusing) situation is that although the differential equations are expressed in terms of reactant concentrations, the simulation is performed using actual particle numbers.

A common source of error is input files which are incomplete or wrong. You should make sure that all variables and constants in the differential equations are given values or initialised, once and only once. Make sure also that the values you give make physical sense, e.g. the concentrations should not be so low that no particle number is greater than zero.

6 Running

Invoke the program with its name and any of the optional command line parameters. If an input file is not given on the command line or in the initialisation file, you will be prompted for one during execution. The invocation names are “stode” (the stand-alone version) and “stodeq” (the GUI-enabled version).

On some platforms (well actually just Windows), the program does not take command line parameters. On platforms that do allow this, the command line parameters may be any of the following:

-h	Prints a usage message.
-l	Prints the licence and terms of use.
-d `level`	Print debugging messages at level `level` or higher. Normally, you may omit this option (or, equivalently, choose runlevel 5). Running at level 4 causes some information on the state of the variables during execution to be shown, but at the cost of overall speed.
-f `filename`	Specify the name of the input file.
-o `filename`	Specify the name of the output file.
-F `dirname`	Specify the name of the input directory.
-O `dirname`	Specify the name of the output directory.
-S `maxsteps`	Specify the maximum number of simulation steps.
-T `maxtime`	Specify the maximum (simulation) time for which to run.
-i `interval`	Store results of every `interval`'th step.
-s `conc_scale`	Use concentrations of `conc_scale` molar.
-r `rcfile`	Specify the name of the run-control file.

If the GUI version is being run, the following may additionally be specified:

-b `interval`	Display results of every `interval` th step.
-B `size`	Display no more than `size` points on screen.
-c `interval`	Display results of every `interval` th step in the moving window.
-C `size`	Display no more than `size` points in the moving window.
-t `timeout`	Re-draw every `timeout` milliseconds. Using a low value here makes for a smoother display, but (slightly) slows down the program speed. Your choice.

Each of these parameters may also be specified in the run control file. The command line parameters have a higher priority.

If you're running the GUI enabled version, you can change many of these parameters at run time.

After you hit the enter key, the program begins. Depending on the reactions being simulated and the chosen parameters, this could take a loooooong time. So if the program is being run in stand-alone mode, consider running it in the background (under Unix, place an "&" at the end of the command line).

The GUI version allows several simulations to be run at once. Navigate from the main window File menu to Load predefined reaction sets or create a New window into which a reaction set specification may be entered. Each of these pops up a new simulation window. The keyboard short-cut `CTRL_L` will also popup the load files window, and there are similar short-cuts for most of the menus throughout the program.

In the GUI version, you'll need to start the simulation explicitly. From the Actions menu, you can choose to process the input before starting the simulation, run the simulation, pause or halt it, and reset the simulation. Some options may be grayed out if they are not available from the current state. While running, you can choose an option to see only the most recent results, or the whole buffer.

In each of the simulation windows, there are also menu items to save the edited specification files, choose the output file name, toggle the displayed variables, edit the run-time parameters, brew coffee, and much more. Explore.

A final note of caution: the output files can be *very* large. For a run which lasts several hours or days, make sure you have sufficient space on your hard drive, or choose a higher sampling interval to restrict the output size. Running the model in the GUI for a short while might give you an idea of how big the file might get, how much detail you want to include, etc.

7 Viewing the Output

If the GUI version is being run, the output is displayed as it is produced in the window. As described above, there are various options to control this.

For both GUI and non-GUI versions, an output file is generated, along with a file containing gnuplot commands. The gnuplot program is available separately, and is available from www.gnuplot.org . To view the output with gnuplot, in a terminal window issue the command `gnuplot xxx.gpi`, where `xxx` is the base name of the input and output files. The gnuplot command file is generated before the simulation begins, so that users may begin viewing the data as it is generated. This means however that the run length specified initially is used by gnuplot to scale the time axis, and consequently the initial data may appear to be quite compressed on the left hand side of the plot. This may easily be altered by editing the gnuplot command file (or a copy) to reflect the time range the user wishes to view, and running gnuplot with the edited version. The user will probably also wish to do this if the run ends before the initially specified time is reached.

The format of the output files (.sdo extension) is as follows: The first few lines are preceded by a "#", which comments them out as far as gnuplot is concerned. These lines give the number of reactants, and the names of each reactant. Each subsequent line is a line of data from the simulation, beginning with the initial conditions. The first item in each line gives the time, the second the number of steps so far, and the next items the populations of each of the reactants (in the order in which they are named above).

The output file may also be post-processed with the utility "stodeout-proc". This prepends three (gnuplot commented, as above) lines to the output, reporting the total time taken by the simulation, and the minimum and maximum time differences between pairs of adjacent steps.

A further utility "dilute" takes a completed output file, and produces a second version with fewer data points. The dilution factor is specifiable. This is useful, for example if you have an output file which is very big and oversampled. (Gnuplot doesn't like files which are too big). Bear in mind that thinning a file out by too much can cause important information to be lost. If you have information which may be lost in this way, consider viewing only a small time-slice of the data at a time.

8 Troubleshooting

Q: The simulation windows have lost the "Plot" menu. What's gone wrong?

A: For some bizarre reason I don't understand, the second and subsequent simulation windows in the GUI version don't display the "Plot" menu. It is, however, there - it's just hidden. To display it, click on the area just to the right of the last menu item ("Actions") on the menu bar, and the "Plot" menu will magically appear. At least I hope so.

Q: I had two simulations running in the gui. The first (older one) was highlighted and in the foreground, and had finished running. The newer simulation was in the background and was still running. When I used the keyboard shortcut (Alt-C) to close the foreground simulation window, the background one was closed instead. Are keyboard shortcuts grabbed by the most recently opened simulation, irrespective of which has the focus? Should I use the menus instead in such situations?

A: Yes. Yes.

Q: The simulation won't run. I get messages saying "parse error" or "Error parsing input". What should I do?

A: This is an error processing the input. Make sure your input file has the correct form (follow the example input file). In particular it is important to use exactly the correct relational symbols (=, := and −>) with the appropriate input file lines. Additionally, there should be a newline after the last line in the file.

Q: I'm confused by the above.

A: So will the parser be if you get it wrong.

Q: My input appears to be correct and is correctly parsed. But I nevertheless get run errors.

A: It may be that your input is not valid - for example, you may have chosen concentrations (or the cell volume or concentration scale) such that all particle numbers are zero. One way to spot this is to run the program with a low debug value and to examine the output.

9 To Do

Create a man page, make the package available as an rpm.

10 Bugs

I've tried hard to eliminate all bugs and to make this program as useful and user friendly as possible. But if you do find any glitches, or have improvements to suggest, please mail me at carel@eml.org . If the program crashes or behaves pathologically, it would help me if you provide as much debugging information as possible: i.e. set the debug level to one (command line or run-control file), and save the debugging information to a file. Then send that file, together with the input file that was being used when the crash happened, to me.