Parallel stochastic simulation using graphics processing units for the Systems Biology Toolbox for MATLAB

Software usage guide

Guido Klingbeil, Radek Erban, Mike Giles and Philip K. Maini

This document gives a brief guideline on how to use the parallel stochastic simulation of chemical reaction systems on graphics processing units (GPUs) plugin for MATLAB (STOCHSIMGPU plugin). The plugin is tightly integrated into the System Biology Toolbox 2 for MATLAB (SBTOOLBOX2) [7]. The STOCHSIMGPU plugin is a direct replacement for the stochastic simulation algorithm provided by the SBTOOLBOX2 but provides additional features. These are the implementation of three exact stochastic simulation algorithms, Gillespie's stochastic simulation algorithm (SSA) [3], Li and Petzold's logarithmic direct method (LDM) [4] and the next reaction method (NRM) by Gibson and Bruck [2] and the histogram computation across all realisations of a stationary process, the steady state or at any user defined point in time. These additional features are available through additional optional parameters given to the SBstochsim script.
The STOCHSIMGPU plugin is a direct substitute and no change to the user's code is required. The usage of the stochastic simulation within the SBTOOLBOX2 is given in documentation at `http://www.sbtoolbox2.org`. The basic background in parallel computing on a GPU and the implemented exact stochastic simulation algorithms is given in the supplemental material.

Requirements

Hardware requirements. The STOCHSIMGPU plugin requires an Intel x86 32-bit or 64-bit CPU and an NVIDIA GPU with at least compute capability 1.2 (NVIDIA GeForce GT 240 or later).
The host's main memory available should be twice the amount of the graphics board

memory. The GPU stores the simulated realisations in 32-bit single precision floating point numbers, but MATLAB stores them in 64-bit double precision floating point numbers doubling the memory requirement.

Software requirements. The STOCHSIMGPU plugin requires MATLAB and the SBTOOLBOX2. The code has been developed and tested under Linux only, but it should work using Windows or Mac OS X. STOCHSIMGPU was developed using OpenSUSE 11.1 [6]. To build the plugin, MATLAB and a C/C++ compiler supported by your MATLAB version are required. The code has been tested using MATLAB 2008b or later and was developed using MATLAB 2010a and gcc 4.2.3 [1]. The CUDA toolkit 2.2 or later is required. The plugin has been developed using version 3.0 [5].

1 Installation

The installation of the STOCHSIM GPU plugin is handled by the $\mathrm{install.m}$ installation script. Change to the directory of the stochastic simulation plugin and call the $\mathrm{install.m}$ script. The installation script will check for a compatible CUDA device and will only proceed if one is found. It will compile and copy the MATLAB stochastic simulation plugin into the correct location. In order to do so, it will ask for the SBTOOLBOX2 installation path. Please enter the absolute SBTOOLBOX2 installation path.
You will be notified and asked for confirmation before copying any files. The original files of the SBTOOLBOX2 are archived. As soon as the installation script finishes, the MATLAB plugin is ready for use.

2 Usage of the MATLAB plugin

From a user's perspective, the STOCHSIM GPU plugin behaves in the same way as the stochastic simulation MATLAB code provided by the SBTOOLBOX2 by the $\mathrm{SBstochsim}$ function. Please refer to the SBTOOLBOX2 documentation for further information on how the $\mathrm{SBstochsim}$ function is integrated into the SBTOOLBOX2. The STOCHSIMGPU plugin supports supports additional feature which will be discussed below.
The parameters of the SBTOOLBOX2 simulation script $\mathrm{SBstochsim}$ are given in Table 2 below.

2.1 Additional MATLAB options

The STOCHSIM GPU plugin supports additional options. They are handled as variable input parameters and may be given after the parameters supported by the SBTOOLBOX2. These additional parameters are optional and not required. The additional parameters are listed in Table 2.

Parameter	Description	Default
model	The SBTOOLBOX2 SBmodel to be simulated.	None.
time	End time for simulation.	none
V	If no volume of the system is given (V is set to []), the species are given in number of molecules. If a volume is given, it is interpreted as the volume of the reaction space in litres.	none
units	This value is only used in the case that the species are defined in concentration units. Per default nM (units= 1×10^{-9}) are assumed for all species. The units parameter is only used if a reaction volume V is given.	1×10^{-9}
runs	Number of realizations (simulation runs) to be computed.	1000
Nsample	The Nsample argument is handled differently by the MATLAB GPU plugin depending on the selected sampling grid. Details are given below.	1000

Table 1 – Required parameters of the SBTOOLBOX2 SBstochsim function.

The algorithm option selects one of the three available exact stochastic simulation algorithms. Details of the algorithms themselves are given in the Background Supplemental information. String SSA selects Gillespie's stochastic simulation algorithm [3], LDM the logarithmic direct method by Li and Petzold [4], and NRM the next reaction method by Gibson and Bruck [2].
The grid option selects the sampling method to be used. The fixed grid uses a fixed time interval between sampling points. The dynamic grid samples every n-th reaction event. The dynamic grid is the default behaviour of the SBTOOLBOX2, but fixed grid is the default of the STOCHSIM GPU plugin. Using a dynamic grid, it is not certain which time interval the simulation will cover. The memory of the graphics card is limited, so to ensure the simulation runs until the given final time T, a fixed time grid is chosen by default. The meaning of the frequency parameter is different.
The SBTOOLBOX2 the Nsample parameter gives the number of reaction events occurring between the sampling points. This is the same for the MATLAB plugin using a dynamic grid. When using a fixed grid, the Nsample parameter gives the sampling frequency in sec^{-1}. The histogram option controls the histogram computation. To be consistent with the SBTOOLBOX2, the default is to compute no histogram. The histogram is computed across all realisations. The STOCHSIM GPU plugin is capable of computing the histogram of the stationary process or distribution, the steady state and of any user defined points in time throughout the simulation.

2.2 Output data structure

The output data structure of the SBTOOLBOX2 has been augmented by a histogram data member:

Parameter	Description	Default
algorithm	Accepted values are SSA, LDM, NRM. Selects the exact stochastic simulation algorithm to compute the simulation.	SSA
grid	Accepted values are fixed or dynamic. Selects the sampling grid. Details are given below.	fixed
histogram	Controls the histogram computation. If no value is given (histogram set to []), no histogram is computed. If histogram is set to sp, the histogram of the stationary process or distribution is computed. If a vector of time points $[t_1, \ldots, t_n]$ is given, the histograms at the given time points are computed. To compute the histogram of the steady state give the final time of the simulation.	[]
bins	Number of histogram bins.	100

Table 2 – Additional options supported by the STOCHSIM GPU plugin, their excepted values, and the default values.

```
output
 output.time
 output.speciesdata
 output.runs
 output.timemean
 output.speciesdatamean
 output.species
 output.histogram
```

The number of bins given by the user are an upper bound. If there are fewer states (different number of molecules) per molecular species, the number of bins is reduced to the number of states of each molecular species. The potentially differing number of bins for each molecular species needs to be reflected by the used data structure.
The histogram is returned using a $1 \times M$ MATLAB cell array where M is the number of molecular species. Each cell contains the histogram of a single molecular species in a $B \times 2$ array where B is the number of histogram bins. The first column contains the bin boundaries and the second the frequency of occurrence.

3 Examples

This section illustrates the general usage as well as the additional features of the STOCHSIM GPU plugin using two simple example systems.

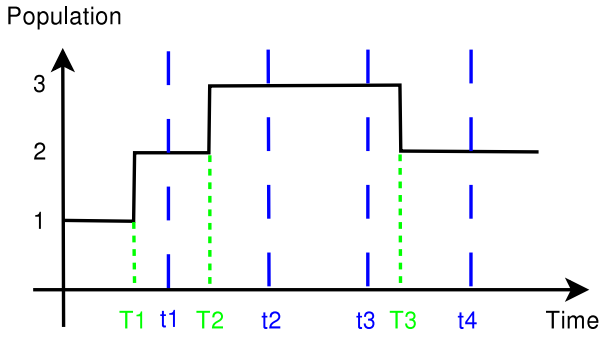

Figure 1 – Sampling the molecular population at fixed time increments or whenever a reaction event occurs. Blue is a fixed sampling grid The molecular populations at equidistant time points are stored. Green is a dynamic sampling grid. The sampling time points are not equidistant, but every n-reaction event is stored.

3.1 Michaelis Menten enzyme kinetics

The Michaelis Menten kinetics describes a simple kinetics converting a substrate S into a product P by the enzyme E. ES is an intermediate enzyme-substrate complex. The enzyme reaction is assumed to be non-reversible. The product is not a substrate of the enzyme:

$$E + S \underset{k_{-1}}{\overset{k_1}{\longleftrightarrow}} ES \xrightarrow{k_2} E + P$$

where k_1, k_{-1} and k_2 are the reaction rates constants. The dynamics of the Michaelis Menten kinetics is shown in Figure 2.
This system has one stable steady state and is well suited to demonstrate the histogram calculation of the STOCHSIM GPU plugin as well its integration into the SBTOOLBOX2. The SBTOOLBOX2 model of the Michaelis Menten kinetics is given in the file michaelis-menten.txtbc and the MATLAB script simulating this system is given in the file michaelis-menten.m. To load the model using the SBmodel command:

```
mm = SBmodel('michaelis-menten.txtbc');
```

The model contains the definition of the reactions, species and their initial conditions as well as the stochastic reaction rates. The initial conditions are given in number of molecules, so we do not have to give a system volume or units of the reaction rates:

```
volume = [];
units = [];
```

We want to simulate for $T = 65sec$ and 10000 realisations:

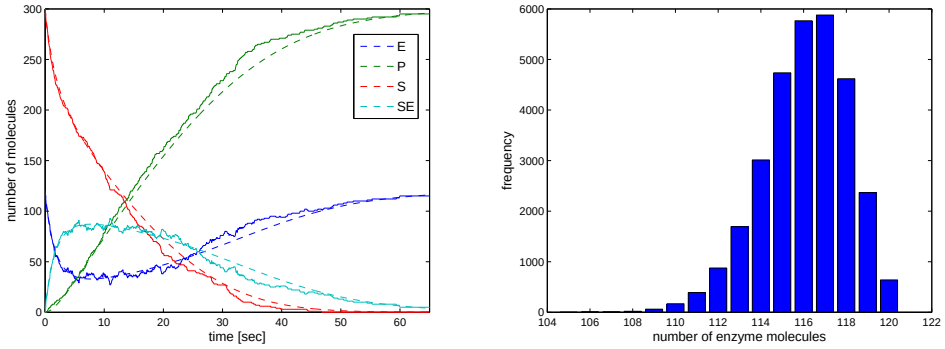

Figure 2 – *Stochastic simulation of the Michaelis Menten enzyme kinetics reaction.* E *is the enzyme,* S *the substrate, and* P *the product.* ES *is the enzyme-substrate complex. The initial conditions at* $t = 0\,\mathrm{h}$ *are* $E = 120$, $S = 300$, $P = 0$, $ES = 0$. *The reaction rates are* $k_1 = 1.66 \times 10^{-3}$, $k_{-1} = 1.0 \times 10^{-4}$, *and* $k_2 = 0.1$. *On the left time evolution of one realisation as solid lines and the average of 10000 realisations as dashed lines are plotted. The histogram of the number of enzyme molecules* E *at* $t = 65\,\mathrm{sec}$ *is shown on the right.*

```
runs = 10000;
time = 65.0;
```

and we want to compute the histogram across all realisations at the steady state. We decide that the histogram shall consist of 200 bins:

```
bins = 200;
```

In contrast to the SBTOOLBOX2, the default of the MATLAB plugin is to use a fixed sampling grid. Simulate the Michaelis Menten system using the LDM algorithm by Li and Petzold and compute the histogram at the steady state:

```
samples = 1000;
mm = SBstochsim(mm, time, volume, units, runs, nsamples,
  'algorithm', 'LDM', 'histogram', time, bins);
```

Plot the time evolution of one realisation and the mean of all realisations:

```
figure;
plot(mm.timemean, mm.speciesdatamean, '--');
hold on;
plot(mm.time{1}, mm.speciesdata{1})
xlim([0 65]);
xlabel('time [sec]', 'fontsize', 12);
ylabel('number of molecules', 'fontsize', 12);
h_legend = legend('E', 'P', 'S', 'SE');
set(h_legend, 'Fontsize', 12);
```

Plot the histogram of the steady state of the enzyme using the MATLAB bar graph function. The steady state of the enzyme is at 120. We expect a distribution slightly below the steady state since we only simulated for $T = 65\,\mathrm{sec}$:

```
figure;
bar(mm.histogram{1}(:,1), mm.histogram{1}(:,2), 'b');
xlabel('number of enzyme molecules', 'fontsize', 18);
ylabel('frequency', 'fontsize', 18);
```

We requested 200 bins for the histogram, but as shown in Figure 2, there are only 16 different numbers of enzyme molecules at the steady state. If more bins than different number of molecules are requested, the number of bins is automatically reduced to the number of different states.

3.2 Schlögl system

The Schlögl system is a simple reaction system with only a single molecular species exhibiting two stable steady states and switching between them:

$$2A \underset{k_2}{\overset{k_1}{\longleftrightarrow}} 3A$$
$$A \underset{k_4}{\overset{k_3}{\longleftrightarrow}} \emptyset$$

The stochastic reaction rates are chosen to be $k_1/V^2 = 0.00015\,\mathrm{min}^{-1}$, $k_2/V = 0.36\,\mathrm{min}^{-1}$, $k_3 = 37.5\,\mathrm{min}^{-1}$, $k_4 V = 2200\,\mathrm{min}^{-1}$. The ODE for this system is given by:

$$\frac{dA}{dt} = -\frac{k_1}{V^2}A^3 + \frac{k_2}{V}A^2 - k_3 A + k_4 V \tag{1}$$

with the reaction rates $k_1/V^2 = 0.000025\,\mathrm{min}^{-1}$, $k_2/V = 0.18\,\mathrm{min}^{-1}$, $k_3 = 37.5\,\mathrm{min}^{-1}$, $k_4 V = 2200\,\mathrm{min}^{-1}$. The ODE model has two stable steady states at $A_{S1} = 100$ and $A_{S2} = 400$ and one unstable steady state at $A_u = 220$. The steady state to which the ODE model evolves depends on the initial conditions. In contrast to the ODE model, a stochastic simulation switches between the two stable steady states as shown in Figure 3. The MATLAB script simulating this system is given in the file $\mathrm{schloegl.m}$ and the SBTOOLBOX2 model $\mathrm{schloegl.txtbc}$. Load the model using the SBmodel command:

```
schloeglbc = SBmodel('schloegl.txtbc');
```

We want to simulate 10000 realisations for $T = 3600\,\mathrm{sec}$ at 1 sample per second. The reaction rates are stochastic reaction rates, so no volume and units are required:

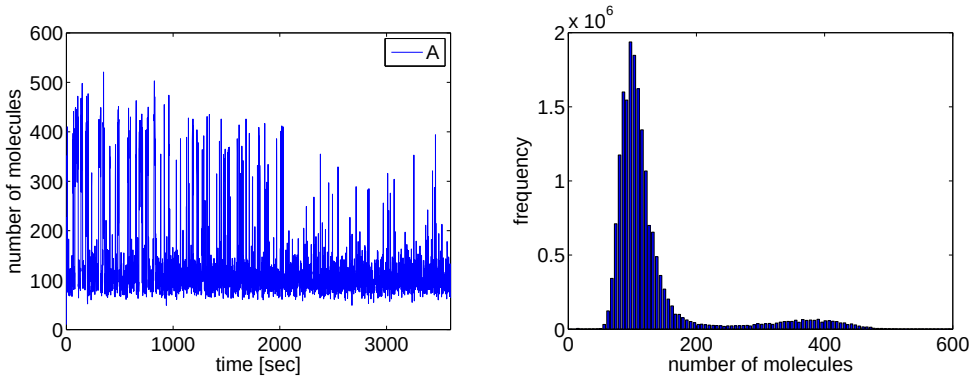

Figure 3 – *Stochastic simulation of the Schlögl system. On the left hand side the time evolution of one realisation is shown. On the right handside the histogram of the stationary distribution. The stochastic reaction rates are chosen to be* $k_1/V^2 = 0.00015\,\mathrm{min}^{-1}$, $k_2/V = 0.36\,\mathrm{min}^{-1}$, $k_3 = 37.5\,\mathrm{min}^{-1}$, $k_4 V = 2200\,\mathrm{min}^{-1}$ *and the initial number of molecules* $A = 10$.

```
time = 3600;
volume = [];
units = [];
runs = 10000;
nsamples = 3600;
```

Call the MATLAB GPU simulation plugin using the SBTOOLBOX2 command SBstochsim:

```
s = SBstochsim(schloeglbc, time, volume, units, runs,
    nsamples, 'algorithm', 'SSA', 'histogram', 'sp');
```

Plot the time evolution of one realisation:

```
figure;
plot(s.time{100}, s.speciesdata{100});
xlabel('time [sec]', 'fontsize', 18);
ylabel('number of molecules', 'fontsize', 18);
legend('A');
set(gca,'FontSize',18);
xlim([0 3600]);
```

The Schlögl system switches between two steady states, so we plot the histogram of the stationary distribution:

```
figure;
bar(s.histogram{1}(:,1), s.histogram{1}(:,2), 'b');
xlabel('number of molecules', 'fontsize', 18);
```

```
ylabel('frequency', 'fontsize', 18);
set(gca,'FontSize',18);
```

References

[1] GCC (2010). The GNU Compiler Collection. http://www.gcc.org.

[2] Gibson, M. and Bruck, J. (2000). Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, **104**, 1876–1889.

[3] Gillespie, D. (1977). Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, **81**(25), 2340–2361.

[4] Li, H. and Petzold, L. (2006). Logarithmic direct method for discrete stochastic simulation of chemically reacting systems. Technical report, Department of Computer Science, University of California Santa Barbara.

[5] NVIDIA (2010). NVIDIA CUDA Toolkit 3.0. http://developer.nvidia.com/object/cuda_3_0_downloads.html.

[6] OpenSUSE (2008). OpenSUSE Linux 11.1. http://software.opensuse.org/111/en.

[7] Schmidt, H. and Jirstrand, M. (2006). Systems Biology Toolbox for MATLAB: a computational platform for research in Systems Biology. *Bioinformatics*, **22**(4), 514–515. The toolbox is available at: http://www.sbtoolbox2.org.