Manual
Simple $A \rightarrow B$ - Ensemble Average

Table of Contents

1. Introduction and System Description

The purpose of this manual is to give a quick introduction to the system under consideration, the general features of the algorithm used to solve the problem and a fast guide to run the simulations. Further details on the functions, methods and other documentation can be found within the *sqrt0_simpleAB_1D_ens_avg.cpp* file. It is assumed that the user has enough C++ knowledge to run and modify a simple code, as needed.

1.1 The system under consideration

Consider a system consisting of a liquid of infinite volume where molecules of type A are diffusing around the fluid undergoing Brownian motion. A porous material is immersed in the fluid so that the molecules can diffuse in and out of the pores, i.e., finite cylindrical channels, see Figure 1. The pores are cylindrical, transverse the molecule completely and do not intersect with each other; we shall only consider a single pore. When the A molecules reach the **inside** wall of the pore, they can react to turn into a B molecule, that can react and turn back to an A molecule. The molecules do not have any other interaction than a steric interaction; the molecules cannot intersect each other or the pore walls. If the pore is narrow enough, the A and B molecules inside the pore will not be able to pass each other, or the pore could be wide enough so that the A and B molecules will pass each other. If the pore has a length L, the goal is to find the probability of a molecule $Y = \{A, B\}$ being in the range $x = [0, L]$; that can also be considered the concentration of molecules at x.

When many molecules are considered, a continuous Langevin molecular dynamics simulation would not be able to capture the required time scale. Thus, the model is coarse-grained to a one-dimensional discrete lattice-gas model. In the coarse-grained model, the molecules diffuse by hopping to nearest neighbor sites in a linear pore of length L and cross-section 1×1. The molecules can hop to nearest neighbor empty sites with probability $P = 1.0$ at a rate h_Y; where h_Y is the hop rate of the Y molecule specie, $Y = \{A, B\}$ If the site is occupied by a molecule specie, the molecules will exchange with probability $P = P_{ex}$ at a rate $h_{ex} = P_{ex}(h_A + h_B)/2$. Additionally, the A molecules can catalytically convert into B molecules at a rate k_{AB}, and B molecules can turn to A molecules at a rate k_{BA}. In general, there can be both A and B molecules

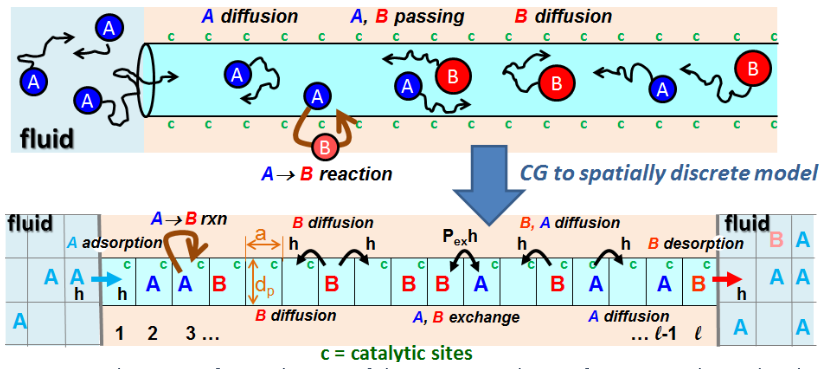


Figure 1: Schematic of a single pore of the system under configuration; the molecules are represented by spheres. The model is coarse-grained to a lattice-gas form.

in the outside fluid, that we shall consider well stirred. To specify the model completely, the parameters that have to be given as input are:

- The concentration χ of molecules A and B in the outside fluid. The concentrations of molecules in the outside fluid must satisfy:

$$\chi_A + \chi_B \leq 1,\ \ \chi_A, \chi_B \geq 0.$$

- The hop rate of A and B molecules h_A and h_B. The hop rates of the molecules have to be positive or zero:

$$h_A, h_B \geq 0.$$

- The reaction rate k_{AB} of $A \rightarrow B$ and the reaction rate k_{BA} of $B \rightarrow A$. The reaction rates have to be positive or zero:

$$k_{AB}, k_{BA} \geq 0.$$

- The passing probability of the molecules P_{ex}. Since P_{ex} is a probability:

$$0 \leq P_{ex} \leq 1.$$

For a detailed list of the parameters to be used as input see Table 1. In reality, the concentrations of the molecules in the outside fluid are not directly used, but are used as the parameters to determine the adsorption and desorption parameters for each type of molecule.

1.2 File structure and location of the input parameters

The file starts with a brief description of what the program does and some basic information:

```
/*

 Title: Kinetic Monte Carlo Algorithm for one to one conversion.

 Author: Andres Garcia.

 Algorithm Type: Non-Rejection.

 Description: Kinetic Monte Carlo Algorithm for one to one diffusion conversion of particles in a one dimensional
 pore. The particles can hop in and out of the pore, that is determined by the concentration of particles outside
 of the fluid. Instead of taking an ergodic average, it takes an ensemble average over many simulations.

 Last Date Modified: October, 6 / 2016.

*/
```

Table 1: Parameters of the model to be used as input of the simulation.

Concentration of A molecules	Concentration of B molecules	Hop rate of A molecules	Hop rate of B molecules	Reaction rate of A molecules	Reaction rate of B molecules	Passing propensity
χ_A	χ_B	h_A	h_B	k_{AB}	k_{BA}	P_{ex}

Following the basic information is the "Modules and Imports" section. In this section, the specific libraries to be imported are defined; these contain the functionality for the needed operations:

```
//-------------------------------------------------------
//   MODULES AND IMPORTS
//-------------------------------------------------------

#include <algorithm>  // Has the functions for Min and Max (see documentation)
#include <fstream>  // To write the info in a file
#include <iostream>  // Basic input/output file
#include <math.h>  // Trigonometric, exponential, logarithmic functions, etc.
#include <random>  // Includes the random number generators (we want the Mersenne Twister, mt19937_64)
#include <sstream>  // Creates a special input/output stream
#include <string>  // To be able to use cout with strings and other basic operations
#include <time.h>  // Contains time functions to seed the generator
#include <vector>  // Use of dynamic arrays to select moves

using namespace std;
```

Next, the pseudo random number generator is defined. In this case we use the Mersenne Twister that is seeded by a pure random number generator, or by an unsigned integer seed. The generator is not the same as the distribution used, thus, the distribution is also defined. The distribution is defined to be a uniform real distribution that generates long double numbers in the range (0.0L, 1.0L); where the suffix "L" indicates that the number should be interpreted as a long double number:

```
//-------------------------------------------------------
//   RANDOM NUMBER GENERATOR(S)
//-------------------------------------------------------

random_device rd;
unsigned int seed = rd();  // Generate a random number to seed the mt19937_64
mt19937_64 generator(seed);  // Get the random number generator
uniform_real_distribution<long double> rand_real3(0.0L, 1.0L);  // The distribution over which to generate numbers
```

The section after is “File Names and Save Options”. This program outputs files with the final statistics, so that they can be read and analyzed using an appropriate graphical interface. Thus, the names of the output files are defined here for ease of manipulation:

```
//-----------------------------------------------------
//   FILE NAMES AND SAVE OPTIONS
//-----------------------------------------------------

const string FL_NM_GAMMA = "simpAB-gamma.csv";
const string FL_NM_SINGL = "simpAB-singl.csv";
const string FL_NM_SINGL_SYM = "simpAB-singl-sym.csv";

const string FL_NM_POREC = "simpAB-poreConfig.csv";
```

The names of the files refer to:

- FL_NM_CUSTM: The name of the file used for customized user output.
- FL_NM_SINGL: The name of the file used for outputting the statistics of the single site statistics $\langle X_n \rangle$.
- FL_NM_SINGL_SYM: The name of the file used for outputting the statistics of the *symmetrized* single site statistics $\langle X_n \rangle$.
- FL_NM_POREC: The name of the file used for outputting the state/configuration of the pore at a specific time.

The section that follows is “Constants”, that is the most important section since it is the section where the parameters have to be input for the program to run properly. It will be discussed in a section of its own, thus, for the moment, the section after this one is the one that is going to be described.

The section following is the “Variables” section. In this section, the storage and statistics variables are defined:

```
//-----------------------------------------------------
//   VARIABLES
//-----------------------------------------------------

// Percentage of simulation complete
long double perc;
```

```
// Current simulation number
int nSims;
int nsSims;

// Array that contains the atom types of the system
int atom_types[PORELEN];

// Array that contains the rates of the system
long double rates[NMOVES];

// Rate related parameters
long double total_system_rate;  // Total rate of the system

// Time variables
long double time_ell;  // Elapsed time
vector <long double> timeTrack;

// Variables to keep statistics
long double accum_stats_singl[MTSIZE][PORELEN][NATOMS];
long double gammat[MTSIZE][NATOMS];

// Arrays that contain the moves for the hops in the pore
int hop_mvs_A[PORELEN];
int hop_mvs_B[PORELEN];
int swp_mvs_AB[PORELEN];
int rxn_mvs_A[PORELEN];
int rxn_mvs_B[PORELEN];

// Pointers to the arrays that contain the moves for the hops in the pore
int hop_ptr_A[PORELEN];
int hop_ptr_B[PORELEN];
int swp_ptr_AB[PORELEN];
int rxn_ptr_A[PORELEN];
int rxn_ptr_B[PORELEN];

// The number of moves available in the pore
int n_mvs_A;
int n_mvs_B;
int n_mvs_rxn_A;
int n_mvs_rxn_B;
int n_mvs_swp_AB;
```

The documentation in the code is self-explanatory.

This program has a ***save and load*** features in case anything goes wrong or the simulation has to be stopped and run from the last save point. The variables in the "Save and Load File Constants and Variables" section accomplishes this:

```
//------------------------------------------------------
//   SAVE AND LOAD FILE CONSTANTS AND VARIABLES
//------------------------------------------------------

//Name of the file where to save the simulation state
const string STATE_FILE = "state.csv";
const string GEN_STATE_FILE = "genState.txt";

//If the simulation should be saved/loaded
const bool LOAD_FILE = false;
const bool SAVE_FILE = true;
```

The variables are defined such that:

- const string STATE_FILE: The name of the file where the state of the simulation is stored. Two files are created to account for file corruption; e.g., if the simulation is stopped in the middle writing one of the files. The extension must be that of a comma separated file (*.csv*). This file does ***not*** save the seed or the state of the random number generator.

- const string GEN_STATE_FILE: The name of the file where the random number generator of the simulation is saved. Two files are created to account for file corruption; e.g., if the simulation is stopped in the middle writing one of the files. The extension must be that of a text filed (*.txt*).

- const bool LOAD_FILE: The boolean variable that denotes if a simulation file should be loaded. True means that a simulation should be loaded from a file. False means that a simulation should ***not*** be loaded from a file.

- const bool SAVE_FILE: The boolean variable that denotes if the simulation should be saved. True means that a simulation should be saved to a file periodically. False means that a simulation should ***not*** be saved to a file.

2. Running the Simulation

2.1 Quick start – Running a simple simulation

To simply run the simulation, under the section "Constants":

```
//----------------------------------------------------
//   CONSTANTS
//----------------------------------------------------
```

The following parameters have to be given as input:

- The number of simulations over which to average; we shall use 1000 for this example:

```
// Length of the pore
const int NSIMS = 1000;
```

- The pore length, that is the length of the lattice to be examined; for this example, we shall use a pore length of 100:

```
// Length of the pore
const int PORELEN = 100;
```

- The input parameters of the outside concentration of the molecules; for this example, we shall use $\chi_A = 0.3$ and $\chi_B = 0.47$:

```
// Equilibrium values
const long double x_equil_A = 0.3L;
const long double x_equil_B = 0.47L;
```

- The exchange probability between A and B molecules is chosen as $P_{ex} = 0.82$; notice that this is a number in the range [0.0L, 1.0L]. This must be specified before the hop rate of the molecules:

```
// Exchange probability
const long double pex = 0.82L;
```

- The hop rates of the molecules A and B, i.e., h_A and h_B. For this example, we choose $h_A = 1.0$ and $h_B = 2.0$:

```
// Hop rates for the molecules in the system
const long double hA_L = 1.0L;
const long double hB_L = 2.0L;
```

- Choose the reaction rate for the process $A \rightarrow B$ as $k_{AB} = 0.1$ and the reaction rate for the process $B \rightarrow A$ as $k_{BA} = k_{AB}/2$:

```
// Reaction rate for the molecules
const long double rxn_A = 0.1L;  // Reaction rate for a molecule A to go to B
const long double rxn_B = 0.5L * rxn_A;  // Reaction rate for a molecule B to go to A
```

- The set of times for which the statistics must be taken; the **5** times to be explored are defined in the array below:

```
// Time constants
const int MTSIZE = 5;
const long double maxTime[MTSIZE] = {0.0L, 5.0L, 25.0L, 125.0L, 625.0L};
```

IMPORTANT: Notice that the vector maxTime needs to have the same number of entries as denoted by MTSIZE.

To run the program, run with the usual code with the usual compiler and with the flag:

-std=c++11 or *–std=c++0x*

or equivalent.

2.2 Further functionality – Setting up more features

It may happen that the model could have different adsorption and desorption parameters than the ones pre-defined in the sample file. To modify the adsorption and desorption parameters, change the settings under the "Constants" section that correspond to the adsorption and desorption parameters:

```
// Adsorption and desorption rates
const long double ads_rate = hA_L * x_equil_A + hB_L * x_equil_B;
const long double des_rate_A = hA_L * (1.0L - x_equil) + hE_AB * x_equil_B;
const long double des_rate_B = hB_L * (1.0L - x_equil) + hE_AB * x_equil_A;
```

where *ads_rate* is the adsorption rate to empty sites and *des_rate_A* and *des_rate_B* are the desorption rates of A and B molecules respectively.

2.2.1 Additional functionality of the program

There is the option of printing the configuration of the pore using the function:

void print_pore(int).

This function receives an integer parameter defined in the “Files Names and Save Options”.

```
//----------------------------------------------------
//   FILE NAMES AND SAVE OPTIONS
//----------------------------------------------------
```

Before using the function to print the state of the pore, the function should be called to open and generate the file where the information will be stored. The way to call the function is:

print_pore(OPEN_FL).

To write the pore configuration, use:

print_pore(WRIT_FL).

3. Output Files

When the program is done running, output files will be generated with the statistics. The files that are generated by default are the files that contain the single, double, triple, quartet and quintet statistics.

The files are generated by the function *write_info_to_file()*, that contains the instructions. Since the computing precision of the spreadsheet programs and other programming languages can be less than the long double used in the program, there is a function to generate customized output:

write_info_to_file_custm().

The statistics for specific quantities can be obtained from the accumulated statistics arrays:

```
long double accum_stats_singl[MTSIZE][PORELEN][NATOMS];
long double gammat[MTSIZE] [NATOMS];
```

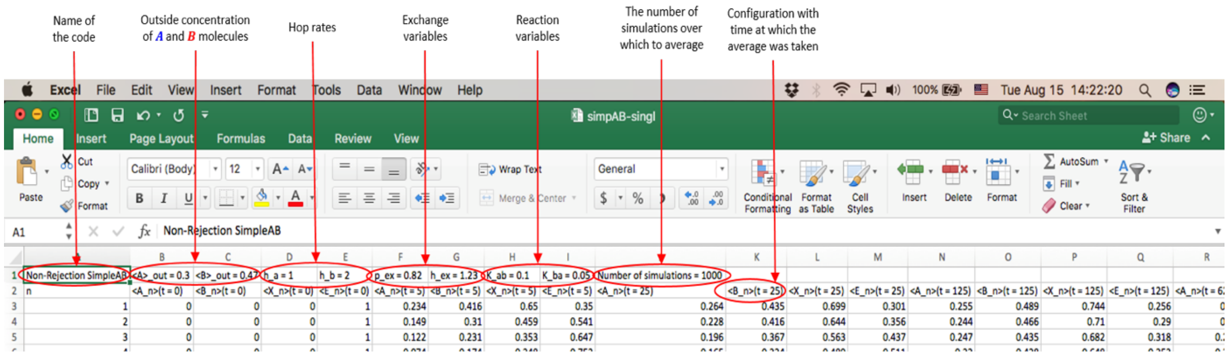
The arrays contain the quantities:

- long double accum_stats_singl: This array contains the number atoms (represented by the index in the box corresponding to NATOMS), at a certain site (represented by the index in the box corresponding to PORELEN), at a specific time (represented by the index in the box corresponding to MTSIZE).

- long double gammat: This array contains the number atoms of a specific type (represented by the index in the box corresponding to NATOMS) **in the pore**, at a specific time (represented by the index in the box corresponding to MTSIZE).

3.1 Output file structure – Example

The output files are comma separated valued files. If the files are opened in MS Excel 2016, the files will display the information of the parameters of the system and the information of the probability to find a certain configuration, see Figure 2. The first column indicates the site n of the correlation.



Name of the code
Outside concentration of A and B molecules
Hop rates
Exchange variables
Reaction variables
The number of simulations over which to average
Configuration with time at which the average was taken

A1 | Non-Rejection SimpleAB

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Non-Rejection SimpleAB	<A>_out = 0.3	<B>_out = 0.47	h_a = 1	h_b = 2	p_ex = 0.82	h_ex = 1.23	K_ab = 0.1	K_ba = 0.05	Number of simulations = 1000								
2	n	<A_n>(t = 0)	<B_n>(t = 0)	<X_n>(t = 0)	<E_n>(t = 0)	<A_n>(t = 5)	<B_n>(t = 5)	<X_n>(t = 5)	<E_n>(t = 5)	<A_n>(t = 25)	<B_n>(t = 25)	<X_n>(t = 25)	<E_n>(t = 25)	<A_n>(t = 125)	<B_n>(t = 125)	<X_n>(t = 125)	<E_n>(t = 125)	<A_n>(t = 6
3	1	0	0	0	1	0.234	0.416	0.65	0.35	0.264	0.435	0.699	0.301	0.255	0.489	0.744	0.256	0
4	2	0	0	0	1	0.149	0.31	0.459	0.541	0.228	0.416	0.644	0.356	0.244	0.466	0.71	0.29	0
5	3	0	0	0	1	0.122	0.231	0.353	0.647	0.196	0.367	0.563	0.437	0.247	0.435	0.682	0.318	0.

Figure 2: Example of the output form of the statistic files. The simulation parameters are shown are the top of the file. The specific correlations that are displayed are in the second line, with the first column labeling the site number.