README FOR C++ CODE FOR COMPUTING DIFFUSIVITIES FROM PARTICLE MODELS OUT OF EQUILIBRIUM: SEP AND KAWASAKI CODE

Contents

1. General remarks

This readme is to introduce into using the C++ code written to test the numerical method for computing diffusivities for the Simple Exclusion Process and Kawasaki Dynamics from particle models out of equilibrium presented in:

> Embacher, P., Dirr, N., Zimmer, J., Reina, C.: Computing diffusivities from particle models out of equilibrium, Proc. R. Soc. A, DOI 10.1098/rspa.2017.0694 (arXiv:1710.03680).

Specifically, this code was used to create the data for the simple exclusion process (bottom row of fig. 4-7 in the paper), the Kawasaki-type dynamics (fig. 10), the "sequential" method (fig. 9), the performance comparison with mean square displacement (subsection 5e) as well as the parameter dependence in fig. 8. For the exact settings used for these results, see section 4 of this text. The other sections of this readme file are devoted to translating between the notation used in the code compared to the one used in this paper (section 2), explaining the use of the code (section 3) and read its output (section 5). To reproduce the data in the paper, it is enough to read sections 4 and 5.

The code is written in C++ and only requires the standard libraries included in C++98. It was tested with Microsoft Visual C++ 2010 Express and Windows 7, where all the files of "finddiffusivity" and "readdiffusivity" were each included in one project. The resulting two projects do not need to be in the same file directory on the computer. In the following, the filenames always refer to the `.cpp`-file of the respective name, whereas `.h` files are the corresponding headers and do not need be touched.

While there are many more options available in the code, this text only introduces the ones used for the paper. Other settings may not have been tested recently and might not work properly.

2. Correspondence between the nomenclature in code and paper

Code	Paper	Description
`Nbin`	L	number of sites in the particle system
`Npart`	$\approx N$	approximate number of particles in the particle system (up to a randomization and depending on the initial profile $\eta\left(t_{\text{ini}}L^2,.\right)$)
`wait`	$\left(t_0 - t_{\text{ini}}\right)L^2$	total microscopical equilibration time before measurements are started
`fwait`	$\left(t_0 - t_{\text{prep}}\right)L^2$	last part of microscopical equilibration time before experiments are started, that is individual for each measurement
`dT`	hL^2	microscopical time between initial and final state
`sampling`	R_1	number of samples starting from the initial profile $\eta\left(t_{\text{ini}}L^2,.\right)$ (i.e. total number of samples during the time $\left[t_{\text{ini}}, t_{\text{prep}}\right]$)
`fsampling`	R_2	number of samples starting from each state $\eta\left(t_{\text{prep}}L^2,.\right)$
`samplesize`	P	number of samples in case of the sequential method
`x0`	x_0	concentration point of test function γ
`filterparameters`	a_0, a_1, a_2	other parameters characterising the test functions γ

3. How to use the code

3.1. **Settings to be adjusted in `main`.** Almost all settings can be adjusted in the `main` function. Note that many quantities are denoted by ranges (for instance `Nbinrange` instead of `Nbin`), which then allows to insert a list of parameters (for instance `Nbinrange[]={100,1000,10000};`), which will then be computed one after another. If for several quantites lists are given, then the following hierarchy applies (first-mentioned quantities are set first):

$$\texttt{sampling} \longrightarrow \texttt{fsampling} \longrightarrow \texttt{dT} \longrightarrow \texttt{wait} \longrightarrow \texttt{fwait} \longrightarrow \texttt{Nbin} \longrightarrow \texttt{Npart} \longrightarrow \texttt{x0},$$

i.e. for all other settings fixed, the code is run for all entries in the `x0` list. Once this list is done for this setting, the `Npart` is changed to the next setting in its `Npartrange` etc. Note that while the other parameters lead to separate entries in the output textfiles, different x_0-values are written in the same entry. Also note that the algorithm only allows for filterfunctions γ, that vanish at zero and one, as it gives wrong results for such a test function otherwise (other entries in a `x0range` are not affected by this, though). Hence, particularly $x_0 = 0$ or $x_0 = 1$ are not allowed.

Additionally to the ones already given in the table of section 2, the following parameters can be set in `main`:

- `profile`: This sets the initial profile ($\eta\left(t_{\text{ini}}L^2,.\right)$ up to rounding errors). The following options are available ($x \in [0,1]$ is the macroscopic space-coordinate):
  `flat` $\approx \frac{\texttt{Npart}}{\texttt{Nbin}}$
  `sin` $\approx \frac{\texttt{Npart}}{\texttt{Nbin}} \cdot \sin(\pi x)$
  `1-cos` $\approx \frac{\texttt{Npart}}{\texttt{Nbin}} \cdot \frac{1}{2} \cdot (1 - \cos(2\pi x))$
  `sintrans` $\approx \frac{\texttt{Npart}}{\texttt{Nbin}} + 5 \cdot \sin\left(\frac{2\pi\left(x - \frac{1}{2}\right)}{A}\right)$ (with $A = 1$ by default and changeable in `getrho_old`)

- `type`: Determines the underlying stochastic process. `ZRP` denotes the zero range process (and the random walk as a special case of zero range process)(to adjust the jump rates g see section 3.3 item 2), `SEP` the symmetric simple exclusion process and `Kawasaki` the Kawasaki-type dynamic. Additionally, `ZRP_multitag` can be chosen, which then automatically gets evaluated via mean square displacement.
- `mode`: `repeat` stands for the "parallel" method, `continuous` for the "sequential" method desribed in section 5d of the paper.
- `name`: Allows to add a suffix of choice to the names of the output files. Does not affect the calculations. Must not contain space characters.

3.2. **Settings that should not be changed in `main`.** In particular, not to be changed are the following settings in `main`: `micresrange`, `samplingmode`, `initializationtype`, `repeattype`, `storagelength`, `filtertype`, `interpolationtype`, `interpolationref`, `refinement`.

3.3. **Settings that can be adjusted outside of `main`.** The following settings are not in the `main` function, but can also be changed, if the default settings are not convenient:

- The names of output-files can be changed at the beginning of `getfilename` (for the output of the textfile containing initial/final states via `savestatetotext`) and `saveresultstotext` (for the output of the textfile containing the results and most characteristic data).
- The jump-rates per site in the zero range process (without any effect for the other processes): This is done in `g` by commenting out the respective other option.
- Some parameters in the initial profiles $\eta\left(t_{\text{ini}}L^2, .\right)$ are set in `getrho_old`, namely the horizontal stretching A and the overall offset (see `sintrans` at line 36 onwards).

4. Settings used in the paper

Default settings in `main` of "finddiffusivity" are as follows:

- `Nbinrange[] = {5000};`
- `Npartrange[] = {4750};`
- `waitrange[] = {100};`
- `fwaitrange[] = {0.1};`
- `dTrange[] = {0.001};`
- `samplingrange[] = {50};`
- `fsamplingrange[] = {2000};`
- `samplesizerange[] = {1};`
- `x0range[] = {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9};`
- `filterparameters.push_back(1.);`
  `filterparameters.push_back(40.);`
  `filterparameters.push_back(2.);`
- `profile = "sin";`
- `type = "SEP";`
- `mode = "repeat";`

Default settings in `main` of "readdiffusivity" are as follows:

- `x0range[] = {0.025,0.05,0.075,0.1,0.125,0.15,0.175,0.2,0.225,0.25,0.275,0.3,`
  `0.325,0.35,0.375,0.4,0.425,0.45,0.475,0.5,0.525,0.55,0.575,0.6,0.625,0.65,`
  `0.675,0.7,0.725,0.75,0.775,0.8,0.825,0.85,0.875,0.9,0.925,0.95,0.975};`

- `filterparameters.push_back(1.);`
  `filterparameters.push_back(160.);`
  `filterparameters.push_back(2.);`

For `foldername` and `filename` the respective directory and name of the output file for the measured states of "finddiffusivity" (see second item of section 5) have to be inserted (with "\\" replacing the delimiter "\" in the `foldername`).

For the specific settings that were applied for the images in the paper, see the table below. Here all changes are to be applied in "finddiffusivity" and to `main`, unless otherwise stated. Also note that for section 5e only the "finddiffusivity" code was used and no further post processing with "readdiffusivity":

Image	**Setting in code**
fig. 4, bottom	
fig. 5, bottom	`Nbinrange[] = {2000};` with `Npartrange[] = {1900};`, `Nbinrange[] = {5000};` with `Npartrange[] = {4750};`, `Nbinrange[] = {10000};` with `Npartrange[] = {9500};`, `Nbinrange[] = {20000};` with `Npartrange[] = {19000};`, `Nbinrange[] = {50000};` with `Npartrange[] = {47500};`
fig. 6, bottom	`dTrange[] = {0.00001,0.0001,0.001,0.01,0.1,1,10};`
fig. 7, bottom	`fsamplingrange[] = {20,50,100,200,500,1000,2000,5000};`

fig. 8, left	**in "finddiffusivity"**: `Npartrange[] = {125000};`, `type = "ZRP";`,
	`profile = "sintrans";` in `main` and
	`A = 100`, `A = 10`, `A = 1`, `A = 0.25`, `A = 0.1` in `getrho_old`
	in "readdiffusivity" (for each of the outputfiles from above):
	`filterparameters.push_back(1.);`
	`filterparameters.push_back(a1);`
	`filterparameters.push_back(2.);`
	for `a1` = 40, `a1` = 80, `a1` = 200, `a1` = 320, respectively, in `main`

fig. 8, right	**in "finddiffusivity"**: `profile = "sintrans";` in `main` and
	`A = 100`, `A = 10`, `A = 1`, `A = 0.25`, `A = 0.1` in `getrho_old`
	in "readdiffusivity" (for each of the outputfiles from above):
	`filterparameters.push_back(1.);`
	`filterparameters.push_back(a1);`
	`filterparameters.push_back(2.);`
	for `a1` = 40, `a1` = 80, `a1` = 200, `a1` = 320, respectively, in `main`

fig. 9	**for all lines**: `samplingrange[] = {1};`, `fsamplingrange[] = {1};`,
	`samplesizerange[] = {100000};`, `mode = "continuous";`
	for orange line also: `Npartrange[] = {125000};`, `type = "ZRP";` in `main` and
	`g_k = double(k*k);` in `g`
	for red line also: `Npartrange[] = {125000};`, `type = "ZRP";` in `main` and
	`g_k = double(k);` in `g`

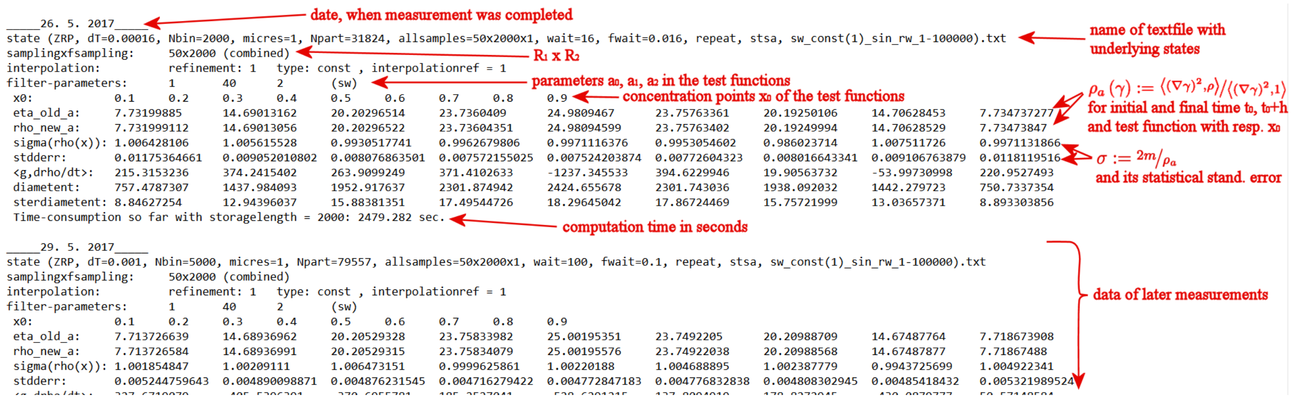
fig. 10	`Npartrange[] = {125000};`, `type = "Kawasaki";`, `profile = "1-cos";`

Image	Setting in code
sec. 5e for MSD	**for all**: `Npartrange[] = {25000};`, `dTrange[] = {0.25};`, `samplingrange[] = {100};`, `fsamplingrange[] = {1};`, `x0range[]={0.5};`, `filterparameters.push_back(1.);` `filterparameters.push_back(160.);` `filterparameters.push_back(2.);`, profile = ”flat”;, `type = "ZRP_multitag";` **for random walk also**: `g_k = double(k);` in g **for ZRP with $g(k) = k^2$ also**: `g_k = double(k*k);` in g
sec. 5e for new method	**for all**: `Npartrange[] = {25000};`, `samplingrange[] = {1};`, `x0range[] = {0.02,0.04,0.06,0.08,0.1,0.12,0.14,0.16,0.18,0.2,0.22,0.24, 0.26,0.28,0.3,0.32,0.34,0.36,0.38,0.4,0.42,0.44,0.46,0.48,0.5,0.52,0.54, 0.56,0.58,0.6,0.62,0.64,0.66,0.68,0.7,0.72,0.74,0.76,0.78,0.8,0.82,0.84, 0.86,0.88,0.9,0.92,0.94,0.96,0.98};`, `filterparameters.push_back(1.);` `filterparameters.push_back(160.);` `filterparameters.push_back(2.);`, profile = ”flat”;, `type = "ZRP";` **for “parallel” also**: `fsamplingrange[] = {40000};` **for “parallel” with less equilibration also**: `fwaitrange[] = {0};`, `fsamplingrange[] = {40000};` **for “sequential” also**: `samplingrange[] = {1};`, `samplesizerange[] = {40000};`, `mode = "continuous";` **for random walk also**: `g_k = double(k);` in g **for ZRP with $g(k) = k^2$ also**: `g_k = double(k*k);` in g
sec. 5e for equilibration time	`Npartrange[] = {25000};`, `samplingrange[] = {100};`, `fsamplingrange[] = {0};`, `dTrange[] = {0.};`, `x0range[]={0.5};`, `type = "ZRP_multitag";` **for random walk also**: `g_k = double(k);` in g **for ZRP with $g(k) = k^2$ also**: `g_k = double(k*k);` in g

5. The output files

By default, the “finddiffusivity” code creates two external text files for each set of parameters (see first paragraph in section 3.1) in the same file directory as its code is located:

- One is called “C++ results.txt” by default and contains the main results (with ten digits precision). If the code is run several times, this file is appended with the last entry being the youngest. As for the formatting, see fig. 5.1.
- One is named according to the chosen parameters of the system by default and contains all the pairs of measured states $\eta\left(t_0 L^2, .\right)$ and $\eta\left(\left(t_0 + h\right) L^2, .\right)$ as well as some data about the system, that is required to correctly post-process the data. This file is used by the algorithm to access already measured states and may not be renamed or relocated on the hard-drive, while the code is running. Note that this file can become rather large (several GB), depending on the size of the stochastic process (`Nbin`) and the number of samples (`sampling` · `fsampling`). If the code is run several times with identical parameters (i.e. the default file name is also the same), this file gets appended with the newest data at the end, thus no data is lost. However, the algorithm for the post-processing is not suited for having different datasets in the same file and will therefore crash. Hence, if the same measurement is meant to be made several times, it is advised to use a different name as suffix (see last item of section 3.1). Particularly,



```
_____26. 5. 2017_____
state (ZRP, dT=0.00016, Nbin=2000, micres=1, Npart=31824, allsamples=50x2000x1, wait=16, fwait=0.016, repeat, stsa, sw_const(1)_sin_rw_1-100000).txt
samplingxfsampling:     50x2000 (combined)
interpolation:          refinement: 1   type: const , interpolationref = 1
filter-parameters:      1       40      2       (sw)
 x0:            0.1     0.2     0.3     0.4     0.5     0.6     0.7     0.8     0.9
 eta_old_a:     7.73199885      14.69013162     20.20296514     23.7360409      24.9809467      23.75763361     20.19250106     14.70628453     7.734737277
 rho_new_a:     7.731999112     14.69013056     20.20296522     23.73604351     24.98094599     23.75763402     20.19249994     14.70628529     7.73473847
 sigma(rho(x)): 1.006428106     1.005615528     0.9930517741    0.9962679806    0.9971116376    0.9953054602    0.986023714     1.007511726     0.9971131866
 stdderr:       0.01175364661   0.009052010802  0.008076863501  0.007572155025  0.007524203874  0.00772604323   0.008016643341  0.009106763879  0.0118119516
 <g,drho/dt>:   215.3153236     374.2415402     263.9099249     371.4102633     -1237.345533    394.6229946     19.90563732     -53.99730998    220.9527493
 diametent:     757.4787307     1437.984093     1952.917637     2301.874942     2424.655678     2301.743036     1938.092032     1442.279723     750.7337354
 sterdiametent: 8.84627254      12.94396037     15.88381351     17.49544726     18.29645042     17.86724469     15.75721999     13.03657371     8.893303856
 Time-consumption so far with storagelength = 2000: 2479.282 sec.

_____29. 5. 2017_____
state (ZRP, dT=0.001, Nbin=5000, micres=1, Npart=79557, allsamples=50x2000x1, wait=100, fwait=0.1, repeat, stsa, sw_const(1)_sin_rw_1-100000).txt
samplingxfsampling:     50x2000 (combined)
interpolation:          refinement: 1   type: const , interpolationref = 1
filter-parameters:      1       40      2       (sw)
 x0:            0.1     0.2     0.3     0.4     0.5     0.6     0.7     0.8     0.9
 eta_old_a:     7.713726639     14.68936962     20.20529328     23.75833982     25.00195351     23.7492205      20.20988709     14.67487764     7.718673908
 rho_new_a:     7.713726584     14.68936991     20.20529315     23.75834079     25.00195576     23.74922038     20.20988568     14.67487877     7.71867488
 sigma(rho(x)): 1.001854847     1.00209111      1.006473151     0.9999625861    1.00220188      1.004688895     1.002387779     0.9943725699    1.004922341
 stdderr:       0.005244759643  0.004890098871  0.004876231545  0.004716279422  0.004772847183  0.004776832838  0.004808302945  0.00485418432   0.005321989524
```

date, when measurement was completed
name of textfile with underlying states
R_1 x R_2
parameters a_0, a_1, a_2 in the test functions
concentration points x_0 of the test functions
$\rho_a(\gamma) := \langle(\nabla\gamma)^2,\rho\rangle / \langle(\nabla\gamma)^2,1\rangle$ for initial and final time t_0, t_0+h and test function with resp. x_0
$\sigma := 2m/\rho_a$ and its statistical stand. error
computation time in seconds
data of later measurements

Figure 5.1. Example for the appearence of the results text file from "finddiffusivity" (see first item in section 5), here with additional annotations in red to explain the file formatting.

the default name does not distinguish between ZRPs with different jumprates g, so this would need to be manually indicated by respective suffixes as well.

Also "readdiffusivity" creates one text file, labelled "C++ read results.txt", in the same file directory as its code is located. This is structured similarly to the "C++ results.txt" file "finddiffusivity" generates.