README FOR CODE USED IN

Harnessing fluctuations to discover dissipative evolution equations

Xiaoguai Li, Nicolas Dirr, Peter Embacher, Johannes Zimmer, and Celia Reina

April 2019

Contents

1. Overview

This file explains all codes used in

Harnessing fluctuations to discover dissipative evolution equations , by Xiaoguai Li, Nicolas Dirr, Peter Embacher, Johannes Zimmer, and Celia Reina.

to generate Figures 3-6 as well as the embedded movies. Specifically, the generation of the figures required four steps: simulations of the zero range process, data post-processing to convert particle fluctuations to components of the discretized dissipative operator, fit of these components (independently or accounting for mass conservation as a constraint), and macroscopic simulations. The names of the folders and their corresponding purpose are listed here.

- **1_ZeroRangeProcess**: C++ code to carry out multiple realizations of a zero range process (see Section 2 for further details).
- **2_Postprocessing**: Matlab code to compute each component of the discretized dissipative operator from particle data for discrete values of ρ and $\nabla\rho$ (see Section 3 for further details).
- **3a_IndependentFit**: Matlab code to obtain a polynomial fit of each independent nonzero component of the discretized operator in the space $(\rho, \nabla\rho)$ (see Section 4 for further details).
- **3b_FitWithConstraint**: Matlab code to perform a fit to the nonzero components of the discretized operator that satisfies mass conservation — this is imposed as a constraint — (see Section 5 for further details).
- **4_MacroSimulations**: Matlab code to perform coarse graining and compute particle-based solution. (Fig.5, Fig.6(b) and supplementary movies) (see Section 6 for further details).

g++11 compiler is needed for the codes in folder 1_ZeroRangeProcess, and all Matlab codes included in this project were tested with Matlab R2014b.

2. Zero Range Process (folder 1_ZeroRangeProcess)

2.1. Overview of the code: Main file and functions

The folder **1_ZeroRangeProcess** is composed of the following files.

- **Main.cpp**: Main file to be launched where parameters are set and options are selected (see next subsection for details).
- **SavingParameters.h/.cpp**: Function that saves the parameters set in the Main file.
- **InitialProfile.h/.cpp**: Function that generates the initial density profile according to the parameters and options selected.
- **ZRP_KMC_fixedBoundary.h/cpp**: Functions that advances the density profile using a Lattice Kinetic Monte Carlo algorithm, and considering Dirichlet (fixed) boundary conditions.
- **Postprocessing.h/cpp**: Function that evaluates and saves $\langle\rho_\epsilon(t_0),\ \gamma_1\rangle$, $\langle\rho_\epsilon(t_0+h), \gamma_1\rangle$, $\langle\rho_\epsilon(t_0), \gamma_2\rangle, \langle\rho_\epsilon(t_0+h), \gamma_2\rangle, \ldots$ for each realization.

2.2. Parameters and options of the code

The following variables and options can be adjusted in the Main file. Their equivalent name in the paper and their meaning is provided below.

- **Configuration and process parameters**:

process	this option can only be set to "ZRP".
profile	this option can be set as "flat", "linear" or "triangle" to generate a flat profile, a linear profile or a triangular profile. Note that the profile is randomly perturbed in InitialProfile.cpp. Flat profile uses parameters "Nbin" and "Npart". "Linear" and "triangle" profile uses parameters "Nbin", "slope" and "center".
Nbin	number of bins
Npart	number of particles
slope	this is only useful when profile is set to be "linear" or "triangle". It sets slope of linear profile, or the positive slope for the triangular profile (the magnitudes of positive and negative slopes are identical in the triangular profile).
center	this is only useful when profile is set to be "linear" or "triangle". It is the value at the center of the profile.

- **Macroscopic time parameters**:

t_equilibration	$t_{prep} - t_{ini}$, time of equilibration.
t_randomize	$t_0 - t_{prep}$, time to prepare the system for a new realization.
h	h, simulation time for actual measurements.

- **Sampling parameters**:

R1	R_1
R2	R_2

- **Basis function parameters**:

Ngamma1	Number of gamma functions in the first basis.
Ngamma2	Number of gamma functions in the second basis. Note: The code is set to postprocess the same ZRP data with two sets of basis functions, although only results from one of them is shown in the paper.

- **Output options**:

file_path1	Directory where the user wants the save the output of the first group of basis functions.
file_path2	Directory where the user wants the save the output of the second group of basis functions.

In addition, the type of Zero Range Process can be set inside the function ZRP_KMC_fixedBoundary.cpp, by adjusting the parameter "power" to 1.0 or 2.0, for instance, for g(k)=k and g(k)=k^2, respectively. All simulations in the article used the value 2.0.

2.3. Running the code

The following commands may be typed on the terminal to compile the code

```
>> module load gcc/6.3.0
>> g++ -std=gnu++0x -O3 Main.cpp InitialProfile.cpp ZRP_KMC_new.cpp Postprocessing.cpp
SavingParameters.cpp -o executable_name
```

where the first command is needed to load version C++11 in order to use the random variable generator used in the code.

2.4. Output of the code and postprocessing

After running the executable, the code generates in each filename path, one summary file and R1 data files:

- **Summary.m**: contains all the parameters prescribed in the Main file.
- **Data_0.m**: each row contains $\langle \rho_\epsilon(t_0), \gamma_1\rangle$, $\langle \rho_\epsilon(t_0+h), \gamma_1\rangle$, $\langle \rho_\epsilon(t_0), \gamma_2\rangle, \langle \rho_\epsilon(t_0+h), \gamma_2\rangle, \ldots$ for each realization $r \in$ [1 :R2], i.e. there are R2 rows. The number in this data file name ranges from 0 to R1.

This data compiles the information of all realizations of a single profile, and can be further postprocessed using the code in folder **2_Postprocessing** to deliver the expected values of ρ and $\nabla\rho$, and each nonzero component of the dissipative operator discretized using the γ functions. We remark that each profile must be postprocessed independently.

2.5. Parameters used to generate the figures in the paper

The default parameters for Figures 4-6 are:

Nbin	t_equilibration	t_randomize	h	R1	R2	Ngamma	x_basis
5000	4e-06	4e-09	4e-11	400	2000	40	[0.0:0.025:0.975]

Profile: “flat” was used with Npart = [20000:500:50000]; For the data with conservation constraint, “linear” was used with slope = ±[5, 11] and center = 7, and “triangle” was used with slope = ±[15, 19] and center = 7; For the data without conservation constraint, “linear” was used with slope = [-13:1:13] and center = 7, and “triangle” was used with slope = [14:1:19] and [-19:1:-14], and center = 7.

3. Postprocessing (folder 2_Postprocessing)

3.1. Code and test data

Folder **2_Postprocessing** includes the following files.

- **Postprocessing.m**: the Matlab script used to do postprocessing of the data from the zero range process, and estimate the elements of the operator that are expected to be non-zero.

- **Summary.m**: output file from the previous code, cf. Section 2.4. **Postprocessing.m** reads this file, and writes new data at the end of it after postprocessing. Specifically, it provides the expected value for the following quantities for each pair of neighboring γ functions:

  ρ_b, $\nabla\rho_b$, $\langle\mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\ldots)}\gamma_b,\gamma_b\rangle$, using variable names **rho_b_b**, **drho_b_b** and **diagonal_b_b**, respectively.

  ρ_b, $\nabla\rho_b$, $\langle\mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\ldots)}\gamma_b,\gamma_{b+1}\rangle$, using variable names **rho_b_bPlus1**, **drho_b_bPlus1** and **off_b_bPlus1**

  ρ_b, $\nabla\rho_b$, $\langle\mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\ldots)}\gamma_b,\gamma_{b-1}\rangle$, using variable names **rho_b_bMinus1**, **drho_b_bMinus1** and **off_b_bMinus1**

  We remark that for a given pair of neighboring γ functions, these are denoted as γ_b and γ_{b+1} (from left to right), for the purposes of computing $\langle\mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\ldots)}\gamma_b,\gamma_b\rangle$ and $\langle\mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\ldots)}\gamma_b,\gamma_{b+1}\rangle$, and the same functions are labeled as γ_{b-1} and γ_b (from left to right), for the purpose of computing $\langle\mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\ldots)}\gamma_b,\gamma_{b-1}\rangle$. For this reason, the vectors **rho_b_b**, **rho_b_bPlus1** and **rho_b_bMinus1** do not coincide.

- **Data_0.m, ..., Data_4.m**: Output files from the previous code, cf. Section 2.4.

The files **Summary.m**, **Data_0.m, ..., Data_4.m** provided in the folder correspond to an example with $R_1 = 5$ and $R_2 = 200$, which may be used to test the code **Postprocessing.m**. The real dataset is too large to be attached here.

3.2. Output of the code and further steps

The output is added to **Summary.m**, as previously explained.

4. Independent fit of the components (folder 3a_IndependentFit)

4.1. Real dataset from the zero range process after postprocessing

The previous code delivers a **Summary.m** file for a specific density profile, that contains the three nonzero components of the discretized operators as a function of discrete values ρ and $\nabla\rho$, probed with such profile. The results from the various profiles must then be compiled into single vectors **rho_b_b**, **drho_b_b**, **diagonal_b_b**, **rho_b_bPlus1**, **drho_b_bPlus1**, **off_b_bPlus1** **rho_b_bMinus1**, **drho_b_bMinus1**, **off_b_bMinus1** and saved in a **Data.mat** Matlab file. The file **Data.mat** included in folder **3a_IndependentFit** corresponds to the actual data used in the paper.

4.2. Code overview

There are three scripts used to fit three nonzero components in discretized operator independently. Their names and purposes are listed below.

- **fit_dia_b_b.m**: polynomial fit of $\langle\mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\ldots)}\gamma_b,\gamma_b\rangle$ as a function of ρ_b and $\nabla\rho_b$.

- **fit_off_b_bPlus1.m**: polynomial fit of $\langle \mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\dots)}\gamma_b, \gamma_{b+1}\rangle$ as a function of ρ_b and $\nabla\rho_b$.
- **fit_off_b_bMinus1.m**: polynomial fit of $\langle \mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\dots)}\gamma_b, \gamma_{b-1}\rangle$ as a function of ρ_b and $\nabla\rho_b$.

4.3. Options of the code

There are three fit options in these three scripts, which can be set by parameter 'fitID', and determines the order of the polynomial in the variables ρ_b and $\nabla\rho_b$.

fitID	order in ρ_b	order in $\nabla\rho_b$
22	2	2
21	2	1
20	2	0

In Figure 4, all nonzero components are fit with fitID=22.

4.4. Output of the code

These three scripts give the following outputs.

fit parameters	**dia_fit_22.mat**	Fit parameters of $\langle \mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\dots)}\gamma_b, \gamma_b\rangle$ with fitID=22.
	off_b_bMinus1_fit_22.mat	Fit parameters of $\langle \mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\dots)}\gamma_b, \gamma_{b-1}\rangle$ with fitID=22.
	off_b_bPlus1_fit_22.mat	Fit parameters of $\langle \mathcal{K}_{(\rho_b+\nabla\rho|_b(x-x_b)+\dots)}\gamma_b, \gamma_{b+1}\rangle$ with fitID=22.
figures	**dia_b_b.fig** **off_b_bMinus1.fig** **off_b_bPlus1.fig**	Plots of the raw data and fitted surface.
errorPlot	**error_dia_22.fig** **error_off_b_bMinus1_22.fig** **error_off_b_bPlus1_22.fig**	Plots of the relative error between the fitted results and analytic values, known for the zero range process considered. In practice, analytic values of the nonzero components are not necessarily known.

Files **dia_fit_22.mat**, **off_b_bMinus1_fit_22.mat** and **off_b_bPlus1_fit_22.mat** will be used to carry out the macroscopic simulations.

5. Fit of components with mass conservation constraint (folder 3b_FitWithConstraint)

5.1. Real dataset from zero range process after postprocessing

The file **Data.mat** included in folder **3b_FitWithConstraint** contains the actual data used in Fig. 6 of the paper using the following scripts, and corresponds to a subset of the data included in **3a_IndependentFit/Data.mat**. The variable names are identical to the ones of the previous Section, and these are explained in Subsections 3.1 and 4.1.

5.2. Code and parameters

There is a single script in this folder, **optimization_threeComponents.m**, which is the Matlab script used to fit the three nonzero components together so as to guarantee mass conservation. Specifically, this conservation constraint implies that sum of these three components is zero, which is directly imposed by reducing the number of unknown polynomial coefficients (similarly to before, a quadratic polynomial in both ρ_b and $\nabla \rho_b$ is considered). An objective error function is defined as the L_2 norm of the difference between the raw data and the quadratic surface, and this is minimized using a quasi-Newton method to identify the optimal polynomial coefficients.

5.3. Output of the code

These three scripts give the following outputs.

fitParameter.mat	Fit parameters for the quadratic polynomial target functions.
error_dia_22.fig **error_off_b_bMinus1_22.fig** **error_off_b_bPlus1_22.fig**	Plots of the relative error between the fitted results and analytic values, known for the zero range process considered. In practice, analytic values of the nonzero components are not necessarily known.

The file **fitParameter.mat** will serve as input data for the macroscopic simulations.

6. Macroscopic simulations (folder 4_MacroSimulations)

6.1. Code overview

In folder **4_MacroSimulations** there are two Matlab scripts used to perform coarse graining. Their names and purposes are listed below.

- **MacroEvolution.m**: main script that delivers the macroscopic evolution using the particle-based operator and the analytical operator.
- **prepare_analytic_m.m**: function used in main script, which prepares the analytic mobility m used in the computation of the analytic solution.
- **grad.m**: function used in main script, which computes gradient for an array, with consideration of periodic boundary condition.

6.2. Options of the code

There are two main options in the code that enable the user to choose between the constrained or unconstrained fit of the discrete operator components, and to specify the type of boundary conditions to be used in the macroscopic simulations. In all cases, a macroscopic domain $[0, 1]$ is considered in Figs. 5 and 6 of the article.

constraint	0	Reads fit result from independent fit.
	1	Reads fit result from fit with conservation constraint.
boundaryCondition	'periodic'	Periodic boundary conditions.
	'Dirichlet'	Dirichlet boundary conditions.

6.3. Parameters of the code

- **Discretization parameters**:

Ngamma	number of basis functions. This value needs to be in agreement with the value of Ngamma used to postprocess the particle data.
dx	mesh size (dx=1/Ngamma)
dt	discretization in time
T	total simulation time
timesteps	total simulation steps (timesteps=T/dt)

- **Initial profile**:

density_0	initial profile (t=0) as a function of x

- **Video setting**:

dump	number of frames
v	video name and format
v.FrameRate	frame speed

- **Output snapshots**:

snapshots	specify time to output snapshots; real simulation time is set to be snapshots/1000

6.4. Output of the code

Output of the code includes two parts:

- **movies**: this code automatically makes a directory named 'movies', and saves in it a movie of the analytical evolution and the particle-based evolution.
- **snapshots**: this code automatically makes a directory named 'snapshots', and saves in it the data of the analytic profile and the particle-based profile at the inquired times. An independent data file is created for each inquired time, and their names coincide with the values entered in the vector snapshots (see parameters of the code).

6.5. Options and parameters used to generate the figures in the paper

Options :

Figures; Movies	constraint	boundaryCondition	snapshots	density_0
Fig. 5(a); Movie1	0	'periodic'	[1 3 6 10]	-3*cos(2*pi*x/xfinal)+7
Fig. 5(b); Movie2	0	'Dirichlet'	[1 3 8]	-5*cos(3*pi*x/xfinal)+7
Fig. 6(b); Movie3	1	'periodic'	[1 3 6 10]	-3*cos(2*pi*x/xfinal)+7

CommonParameters :

Ngamma	dx	dt	T	dump	v.FrameRate
40	0.025	6.2500e-07	0.01	100	5