User guide to the LJD program

F. Zamponi
Dipartimento di Fisica and INFM, Università di Roma La Sapienza*, P. A. Moro 2, 00185 Roma, Italy*
(Dated: September 25, 2020)

The present paper is a user guide to the *LJD* program, a *C++* program that simulates a system of classical interacting particles. The sources can be downloaded from `http://www.phys.ens.fr/~zamponi`.

I. THE MODEL

The program simulates a system of N classical particles of equal mass m in dimension d; they are described by their position q_i and momenta $p_i = m\dot{q}_i$, $(p_i, q_i) \in R^{2d}$, $i = 1 \dots N$. The particle are confined in a cubic box of side L with periodic boundary conditions. Each particle is subject to a *conservative force*, $f_i(\underline{q}) = -\partial_{q_i} V(\underline{q})$, and to a constant *nonconservative force* E_i. The force E_i is locally conservative but not globally such due to periodic boundary conditions. The equations of motion are:

$$\begin{cases} \dot{q}_i = \frac{p_i}{m} \ , \\ \dot{p}_i = f_i(\underline{q}) + E_i - \alpha(\underline{p},\underline{q})\ p_i + \xi_i \ , \end{cases} \tag{1}$$

A. Ensembles

It is possible to simulate four different ensembles.

1. *isokinetic ensemble*: the total kinetic energy $K(\underline{p}) = \frac{1}{2m}\sum_i p_i^2$ is constant. In this case $\xi_i = 0$ and from the constraint $\frac{dK}{dt} = 0$ one obtains
$$\alpha(\underline{p},\underline{q}) = \frac{\sum_i E_i\ p_i + \sum_i f_i(\underline{q})\ p_i}{\sum_i p_i^2} \ . \tag{2}$$
2. *isoenergetic (microcanonical) ensemble*: the total internal energy $H(\underline{p},\underline{q}) = K(\underline{p}) + V(\underline{q})$ is constant. In this case $\xi_i = 0$ and from the constraint $\frac{dH}{dt} = 0$ one obtains
$$\alpha(\underline{p},\underline{q}) = \frac{\sum_i E_i\ p_i}{\sum_i p_i^2} \ . \tag{3}$$
3. *constant friction ensemble*: each particle is subject to a constant friction. Here $\xi_i = 0$ and $\alpha(\underline{p},\underline{q}) \equiv \nu$, where ν is the *friction constant*.
4. *Langevin dynamics*: each particle is subject to a random force and to constant friction. Here $\alpha(\underline{p},\underline{q}) \equiv \nu$ and ξ_i is a white Gaussian noise, *i.e.* $\xi_i(t)$ is Gaussian and $\langle \xi_i(t) \rangle = 0$, $\langle \xi_i(t)\xi_j(0)\rangle = 2T\delta_{ij}\delta(t)$.

B. Entropy production rate

Defining the *kinetic temperature*, $T(\underline{p}) \equiv 2K(\underline{p})/(dN-1)$, [1], the *entropy production rate* is defined as

$$\sigma(\underline{p},\underline{q}) = \frac{\sum_i E_i\ \dot{q}_i}{T(\underline{p})} \ . \tag{4}$$

The *current* $J(x,E)$ is defined as

$$J(\underline{p},\underline{q}) = \frac{\partial \sigma}{\partial E} = \frac{\sum_i u_i\, \dot{q}_i}{T} \tag{5}$$

where $E_i = E\, u_i$, and u_i is a (constant) unit vector that specifies the direction of the force acting on the i-th particle.

C. Discretization of the equations of motion

The equations of motion are discretized using the *Verlet algorithm* [2]; for Hamiltonian equations of motion (*i.e.*, $E = 0$, $\alpha = 0$, $\xi = 0$)

$$\begin{cases} \dot{q}_i = \frac{p_i}{m} \ , \\ \dot{p}_i = f_i(\underline{q}) \ , \end{cases} \tag{6}$$

the Verlet discretization has the form

$$\begin{cases} q_i(t+dt) = q_i(t) + \frac{p_i(t)}{m}dt + \frac{1}{2}f_i(t)dt^2 \ , \\ p_i(t+dt) = p_i(t) + \frac{1}{2}\big[f_i(t) + f_i(t+dt)\big]dt \ , \end{cases} \tag{7}$$

where dt is the *time step size*. This discretization ensures that the error is $O(dt^4)$ on the trajectory $q(t)$ and that $p(t)$ approximates the true momenta with error $O(dt^2)$. The implementation of this algorithm on a computer is discussed in detail in [2].

However, this method requires the forces $f_i(t)$ to depend only on the positions and not on the velocities: hence, it has to be adapted to Eq.s 1. This is done in the following way. The discretized equations are

$$\begin{cases} q_i(t+dt) & = q_i(t) + \frac{p_i(t)}{m}dt \\ & \quad + \frac{1}{2}\big[f_i(t) + E_i - \alpha(t)p_i(t) + \xi_i(t)\big]dt^2 \ , \\ p_i(t+dt) & = p_i(t) + E_i + \frac{1}{2}\big[f_i(t) + f_i(t+dt) \\ & \quad - \alpha(t)p_i(t) - \alpha(t+dt)p_i(t+dt) \\ & \quad + \xi_i(t) + \xi_i(t+dt)\big]dt \ , \end{cases} \tag{8}$$

with the same error as in the standard Verlet discretization. The program stores in the computer, at time t, the positions $q_i(t)$, the momenta $p_i(t)$, the forces $f_i(t)$, and the Gaussian multiplier $\alpha(t)$. Then, it performs the following operations:

1. it calculates the new positions $q_i(t+dt)$ using the first equation;

2. using the new positions, it calculates the new forces $f_i(t+dt)$ (the conservative forces depend only on the positions);

3. it calculates the quantity $\pi_i = p_i(t) + E_i + \frac{1}{2}\big[f_i(t) + f_i(t+dt) - \alpha(t)p_i(t) + \xi_i(t) + \xi_i(t+dt)\big]dt$ such that (from the second equation)

$$p_i(t+dt) = \frac{\pi_i}{1+\alpha(t+dt)dt/2} \; ; \tag{9}$$

4. substituting Eq. 9 in the definition of $\alpha(t+dt)$, Eq. 2 or Eq. 3, one gets a self-consistency equation for $\alpha(t+dt)$, which solution is

$$\alpha(t+dt) = \frac{\alpha_0}{1-\alpha_0 dt/2} \; ,$$
$$\alpha_0 = \begin{cases} \text{Eq. 2} \; : \; \frac{\sum_i E_i \; \pi_i + \sum_i f_i(t+dt) \; \pi_i}{\sum_i \pi_i^2} \; , \\ \text{Eq. 3} \; : \; \frac{\sum_i E_i \; \pi_i}{\sum_i \pi_i^2} \; ; \end{cases} \tag{10}$$

the program computes $\alpha(t+dt)$ according to the latter expression for the isokinetic and isoenergetic ensembles, for the other ensembles $\alpha = \nu$;

5. substituting Eq. 10 in Eq. 9 the program calculates $p_i(t+dt)$.

The noise $\xi_i(t)$ is a Gaussian random variable such that $\langle \xi_i(t) \rangle = 0$, $\langle \xi_i(t+ndt)\xi_j(t) \rangle = 2T\delta_{ij}\delta_{n0}/dt$, and is extracted using the *gasdev* routine of the *C++ numerical recipes*.

Finally, it is possible (at some given points of the trajectory, see below) to rescale the velocities in order to set the kinetic temperature to the chosen value T (in the isokinetic ensemble) or the internal energy to the chosen value E (in the isoenergetic ensemble). This rescaling is done only if the actual value of the kinetic temperature differs from T by less than $\text{TTOLL} \cdot T$, where TTOLL is a real number (and the same in the isoenergetic case).

This procedure allows to calculate the new positions, momenta, forces, and α, at time $t+dt$ according to Eq.s 8 *without approximations*, defining a map S such that $(\underline{p}(t+dt), \underline{q}(t+dt)) = S(\underline{p}(t), \underline{q}(t))$. The (*discrete*) dynamical system simulated by the program will be represented by the map $S(\underline{p}, \underline{q})$ and will approximate the differential equations of motion, Eq. 1, with error $O(dt^4)$ for the positions and $O(dt^3)$ for the velocities.
The map S verifies the following properties:

1. it is *reversible*, *i.e.* it exists a map $I(\underline{p}, \underline{q})$ (simply defined by $I(\underline{p}, \underline{q}) = (-\underline{p}, \underline{q})$) such that $IS = S^{-1}I$;

2. in the *Hamiltonian* case ($\underline{E} = \underline{0}$ and $\alpha = 0$, Eq.s 6) it is *symplectic*, *i.e.* its linearization ∂S verifies the relation $\partial S \, \Xi \, \partial S^T = \Xi$, where $\Xi = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ is the *symplectic matrix* (and 1 represents the $dN \times dN$ identity matrix).

D. Details of the model

The particles are labeled by and index $i = 0, \cdots, N-1$ and belong to two species, A and B. Having chosen an integer number CB, particles B are such that $i\%\text{CB} = \text{CB} - 1$. Hence, the concentration of particles B is $c_B \sim 1/\text{CB}$. The external force has the form $E_i = E \, u_i$, where the unit vectors u_i are parallel to the x direction but with different orientation: for even particles they are oriented in the positive direction, and for odd particles in the negative direction, *i.e.* $u_i = (-1)^i \widehat{x}$, in order to keep the center of mass fixed.

The interaction potential is a sum of pair interactions, $V(\underline{q}) = \sum_{i<j} v(|q_i - q_j|)$, and the pair interaction is represented by a Lennard-Jones potential of the following form:

$$v_{\alpha\beta}(r) = \begin{cases} 4\epsilon_{\alpha\beta}\left[\left(\frac{\sigma_{\alpha\beta}}{r}\right)^{12} - \text{LJ}\left(\frac{\sigma_{\alpha\beta}}{r}\right)^{6} + \text{S}\right] \; , \\ \qquad\qquad\qquad\qquad r \le \text{CUT}\sigma_{\alpha\beta} \; ; \\ 0 \; , \qquad\qquad\qquad\qquad r > \text{CUT}\sigma_{\alpha\beta} \; ; \end{cases}$$

α and β are indexes that specify the particle species ($\alpha, \beta \in [A, B]$). The integer $\text{LJ} \in \{0, 1\}$ allows to choose a Lennard-Jones potential or a Soft-Sphere potential. The real number CUT cutoff the potential at a given distance from the origin, and should be such that $\text{CUT} \cdot (\max_{\alpha,\beta} \sigma_{\alpha\beta}) < L/2$ to be compatible with the periodic boundary condition (*i.e.*, a given particle should interact only with one image of all the other particles). The constant S is such that $V_{\alpha\beta}(\text{CUT}\sigma_{\alpha\beta}) = 0$ ($\text{S} = \text{LJ} \; \text{CUT}^{-6} - \text{CUT}^{-12}$), so that the potential is continuous (but, in general, not differentiable).

The constants m (mass of the particles), ϵ_{AA} and σ_{AA} are fixed to 1, defining the *LJ units*; the unit of time is then $t_0 = \sqrt{m\sigma_{AA}^2/\epsilon_{AA}}$. The reduced density is given by $\rho = N\sigma_{AA}^3/L^3$.

The integration algorithm makes use of a *Nearest-Neighbors (NN) table*: each particles has a list of its *nearest-neighbors*, *i.e.* the particles which are closer than $\text{TAB}\sigma_{\alpha\beta}$ to it. The program calculates only the forces between nearest-neighbors. The lists are updated when a particle moves more than $(\text{TAB} - \text{CUT}) \cdot (\min_{\alpha\beta} \sigma_{\alpha\beta})/2$ (where CUT is the cutoff of the pair interaction potential, see below).

E. The file *costanti.h*

In the file *costanti.h* all the variables defining the details of the system are specified. They must be fixed *before* compiling the program. The constants to be specified are reported in table I. Note that the parameter ν is not used in the isokinetic and isoenergetic ensembles, while TTOLL is not used in the constant friction and stochastic ensembles. Having fixed the constants that determine the model, the program is compiled using the script *compila* and can be executed on the computer.

N	Number of particles
d	Dimension of the space
LJ $\in \{0, 1\}$	Lennard-Jones or Soft-Spheres
ISO $\in \{0, 1, 2, 3\}$	Specifies the ensemble
ρ	Reduced density
CB $\in \{1, \cdots, N\}$	Concentration of B particles
σ_{AB} , σ_{BB}	Pair interaction potential
ϵ_{AB} , ϵ_{BB}	Pair interaction potential
CUT	Cutoff of the pair interaction
TAB	Cutoff of the *NN table*
dt	Integration step size
ν	Friction
TTOLL	Tolerance of the thermostat

TABLE I: Constants to be specified *before* compiling the executable file.

II. EXECUTING THE PROGRAM

The compiled file should be put in a directory together with two files, the initial configuration and a file that contains the constants of the simulation run and is called *run_const*.

A. Configurations

The configurations have the following format: N lines of d columns with the positions in the interval $[-1, 1]$; N lines of d columns with the velocities; a line with 0 at the end.

B. Simulation run

The program simulates a given number of trajectories (labeled by the integer n). Each trajectory is made by a total number PASSI of integration steps labeled by an integer $i = 0, \cdots,$ PASSI-1; it starts from the initial configuration *conf_start_n* and the final configuration is written on the file *conf_end_n* and on the file *conf_start_(n+1)*. Thus, the trajectory $n + 1$ starts from the final configuration of the trajectory n. The integer n ranges from RIPSTART to RIPEND. The file *conf_start_RIPSTART* should be in the same directory of the executable file and of the file *run_const*.

For the first trajectory the temperature is fixed to TIMP (isokinetic) or the energy to EIMP (isoenergetic) and the external force E is fixed to F0IMP. At the beginning of each subsequent trajectory the temperature (or energy) is changed by DT and the external force by DF0.

During the trajectory, the velocities are rescaled to the current value of kinetic temperature every TTERM_INT time steps. The rescaling is switched off for $i >$ TTERM.

RIPSTART	Number of first trajectory
RIPEND	Number of last trajectory
PASSI	Number of steps for trajectory
TOUT	Output *BMSSn.log* and *.dat*
TAU0	Entropy production integration time
TCONF	Print configuration
TTERM	Switch off velocity rescaling
TTERM_INT	Time steps between velocity rescalings
EIMP	Initial internal energy (isoenergetic)
TIMP	Initial temperature (isokinetic)
DT	Temperature/energy jump
F0IMP	Initial value of E
DF0	E jump
T0	Correlations initial time
Tmax	Correlations final time
NCORR	Number of correlation times
QQ	Modulus of $\vec{k}$
Qi	Direction of $\vec{k}$
YZ	Output on *alphaYZn.dat*

TABLE II: Constants to be specified in the file *run_const* and do not require recompiling the executable.

C. Output

The output of the program (in *LJ units*) is divided in some different files.

1. *costantin.log* contains the constants of the run specified in tables I and II. The *internal units* are the units in which the programs does its calculations; in these units, $m = 1$, $\epsilon_{AA} = 1$ and $L = 2$ (only the units of length and time change with respect to the LJ units).

2. *conf_t.n.nsteps* - every TCONF integration steps, the program prints the configuration on this file.

3. *BMSSn.dat* - every TOUT integration steps, the program prints on this file the kinetic temperature $T(\underline{p})$, the potential energy $V(\underline{q})$ and the total energy $H(\underline{p}, \underline{q}) = K(\underline{p}) + V(\underline{q})$.

4. *alfan.dat* - the program integrates the entropy production rate over TAU0 time steps and prints the result on this file; hence, on this file the quantity

$$\sigma_{\tau_0} = \int_0^{\tau_0} \sigma(\underline{p}(t), \underline{q}(t))dt \tag{11}$$

where $\tau_0 =$ TAU0dt, is recorder on subsequent segments of the trajectory. The first column is the entropy production derived by Eq. 2, the second column is the correct entropy production given by Eq. 4.

5. *correlation functions* - the program computes some correlation functions at times t_c logarithmically

spaced between T0 and TMAX; the total number of times t_c is NCORR. The correlation functions are

$$\begin{aligned} F(\vec{k},t) &= N^{-1}\sum_i \langle e^{i\vec{k}[q_i(t)-q_i(0)]}\rangle \\ D^2(t) &= N^{-1}\sum_i \langle |q_i(t)-q_i(0)|^2\rangle \\ F_J(t) &= \langle J(t)J(0)\rangle \end{aligned} \tag{12}$$

The functions $F(\vec{k},t)$ and $D^2(t)$ are computed for particles A and B. The function $D^2(t)$ is computed for each component of q_i. The vector k has the form $\vec{k} = \frac{2\pi}{L}\mathrm{QQ}\hat{k}$, where $\hat{k}$ is parallel to the Qi-th direction (0=x, 1=y, 2=z, $\cdots$). The format of the files is the following; the first columns is the time, the other columns are:

- *Fselfn.dat* has four columns: real and imaginary part of $F(\vec{k},t)$ for particles A and real and imaginary part of $F(\vec{k},t)$ for particles B.
- *MSDn.dat* has $2d$ columns: the first d are the components of $D^2(t)$ for particles A and the other d are for particles B.
- *Jcorrn.dat* has two columns: the first is the correlation function of $J(t)$ defined in Eq. 5, the second is the correlation function of $J'(t)$ which is defined using the velocities in the direction y orthogonal to the driving force.

The correlations are calculated and printed every $Tmax$ time steps. The number of times the program computed the correlation functions is printed in the file *BMSSn.log*.

6. *BMSSn.log* - every TOUT integration steps, the program prints on this file: the total number of integration steps i completed; the number of time steps completed per second (averaged on the trajectory); the center of mass velocity (d components that should be zero); the number of correlations (refers to the files *Fselfn.dat*, *Jcorrn.dat*, and *MSDn.dat*); the average number of integration steps between two subsequent calculations of the *NN table* (should be of the order of 10).

7. *alfaYZn.dat* - if the integer YZ = 1, the program prints on this file the "entropy production" calculated using the current J' instead of J. This quantity has zero average and is not an entropy production, but can be useful to compare the fluctuations of σ with its fluctuations.

[1] D.J. Evans and G.P. Morris, *Statistical Mechanics of Nonequilibrium Liquids* (Academic Press, London, 1990).
[2] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids* (Oxford Science Publications, 1987).