Langevin Dynamics, Part I

HPC Assignment 2 - Feb, 2017

Kamyar Nazeri
Student ID: 100633486

Introduction

In this assignment, we are simulating the Langevin dynamics in its simplest form and perform tests on the code and analyze the output data to build confidence in correctness and accuracy of our simulation.
The following restrictions are imposed in our simulation:

- Particle trajectories are implemented in 2D.
- No interaction exists between the particles.
- Equation is equipped with periodic boundary conditions.

Note: There's no implicit vectorized code in C++, hence the particles initialization is done by creating a data-structure and explicit loop over elements. Today's modern C++ compilers (including gcc and Intel's compilers) automatically generate vectorized code as part of their optimizations[1]

The Langevin Equation:

The Langevin equation describes Brownian motion, the random movement of a particle in a fluid due to collisions with the molecules of the fluid:

$$m\ dv = -\gamma v\ dt + \sqrt{dt}\ c\ \eta(x,t)$$

Where $v \in R^2$ is a velocity of a particle and is implemented in our code using *Velocity Verlet algorithm*[2] and η is the noise term which we implement using *Intel MKL*[3] library to generate random numbers with normal (Gaussian) distribution. As a consequence of the fluctuation-dissipation theorem for this system, we have $c = \sqrt{2\gamma k_B T}$ where k_B is *Boltzmann*'s constant and T is the temperature.

In the code m, γ and $k_B T$ are set to unity, we start nondimensionalizing the above equation to show how setting system parameters to unity makes sense:

$$m\ \frac{dv}{dt} = -\gamma v\ + \frac{c}{\sqrt{dt}}\ \eta(x,t)$$

Parameters: m, γ, c
Variables: x, t
We are finding dimensionless parameters $\hat{x}$ and $\hat{t}$ in terms of m, γ and c (*note that η is a dimensionless random scalar value*)

set: $\hat{x} = \frac{x}{x_s}$, $\hat{t} = \frac{t}{t_s}$, $\hat{v} = \frac{v}{v_s}$ (x_s, t_s *and* v_s *have dimensions*)

$$m\ \frac{d^2x}{dt^2} = -\gamma\frac{dx}{dt}\ + \frac{c}{\sqrt{dt}}\ \eta(x,t) \Rightarrow m\ \frac{d^2(\hat{x}\ x_s)}{d(\hat{t}\ t_s)^2} = -\gamma\frac{d(\hat{x}\ x_s)}{d(\hat{t}\ t_s)}\ + \frac{c}{\sqrt{d(\hat{t}\ t_s)}}\ \eta(x,t)$$

$$\frac{m\ x_s}{t_s^2}\frac{d^2\hat{x}}{d\hat{t}^2} + \frac{\gamma\ x_s}{t_s}\frac{d\hat{x}}{d\hat{t}}\ - \frac{c}{\sqrt{t_s}\sqrt{d\hat{t}}}\ \eta(x,t) = 0$$

$$\frac{d^2\hat{x}}{d\hat{t}^2} + \frac{\gamma\ t_s}{m}\frac{d\hat{x}}{d\hat{t}}\ - \frac{c\ t^{3/2}}{m\ x_s}\ \eta(x,t) = 0$$

The goal is to determine x_s and t_s so that equations become as simple as possible:

$\boxed{\textit{unit of mass} \equiv m}$

$\frac{\gamma\ t_s}{m} = 1 \quad \Rightarrow \quad t_s = \frac{m}{\gamma}$ $\qquad \boxed{\textit{unit of time} \equiv \frac{m}{\gamma}}$

$\frac{c\ t^{3/2}}{m\ x_s} = 1 \quad \Rightarrow \quad x_s = \frac{c\ \sqrt{m}}{\gamma^{3/2}}$ $\qquad \boxed{\textit{unit of length} \equiv \frac{c\ \sqrt{m}}{\gamma^{3/2}}}$

The nondimensionalized form of the original Langevin equation becomes:

$$m\ dv = -\gamma v\ dt + \sqrt{dt}\ c\ \eta(x,t)$$

$$m\ d(v_s\ \hat{v}) + \gamma(v_s\ \hat{v})\ d(\frac{m}{\gamma}\hat{t}) - \sqrt{d(\frac{m}{\gamma}\hat{t})}\ c\ \eta(x,t) = 0 \quad \Rightarrow \quad d\hat{v} + \hat{v}\ d\hat{t} + \frac{c}{\sqrt{m\gamma}v_s}\sqrt{d\hat{t}}\ \eta(x,t) = 0$$

To simplify the differential equation:

$\frac{c}{\sqrt{m\gamma}v_s} = 1 \quad \Rightarrow \quad v_s = \frac{c}{\sqrt{m\gamma}}$ $\qquad \boxed{\textit{unit of velocity} \equiv \frac{c}{\sqrt{m\gamma}} = \frac{\sqrt{2\ K_B T}}{\sqrt{m}}}$

The final dimensionless differential equation:

$$d\hat{v} + \hat{v}\ d\hat{t} + \sqrt{d\hat{t}}\ \eta(x,t) = 0$$

To simplify our simulation, we declare m, γ and $k_B T$ to unity, which means we can totally eliminate the units of m or $k_B T$ (order of 10^{-23}), solve the dimensionless equation and calculate the original variables accordingly:

$$\gamma = m = k_B T = 1 \quad \Rightarrow \quad t_s = 1 \quad x_s = \sqrt{2} \quad v_s = \sqrt{2}$$

When we consider the system on torus, we impose the boundary conditions to the particles. It means the domain size variable L is part of the equation:

$$x > \frac{L}{2} \quad \Rightarrow \quad x = x - L$$
$$x < \frac{-L}{2} \quad \Rightarrow \quad x = x + L$$

The unit of measurement for L is length, hence we can introduce another dimensionless parameter based on unit of length:

$\textit{unit of length} \equiv \frac{c\ \sqrt{m}}{\gamma^{3/2}} \equiv L$ $\qquad \boxed{\textit{nondimensional: } \frac{L\ \gamma^{3/2}}{c\ \sqrt{m}}}$

Test Code/Results:

- **The Periodic Boundary Condition (PBC) Test:**
  We are implementing Periodic Boundary Condition; so that a simulation that consists of only a few hundred particles behave as if it was infinite in size. To implement PBC, at least two steps are needed:

$$x > \frac{L}{2} \quad \Rightarrow \quad x = x - L$$
$$x < \frac{-L}{2} \quad \Rightarrow \quad x = x + L$$

  In our simulations L is the size of the domain, hence boundaries of our simulation box are -$\frac{L}{2}$ and $\frac{L}{2}$. To test our implementation of PBC, we keep track of a moving particle over time and draw a plot; when the particle hit the boundary, it must appear at the opposite side of the boundary. To demonstrate the effect of PBC, the size of the simulation box must be small enough to let particles touch the boundaries. If the size of the domain is very small (*figure 1*) then particles might go out of the boundaries more frequently than a medium size domain (*figure 2*). If the size of the domain is large, then we need to extend the time-duration of our simulation ($tmax$) to let particles touch the boundary (*figure 3*); however there's no guarantee that the particles might touch the boundary when the size of the domain is large (*figure 4*). Note that in all of our simulations, the particle start at the origin (0,0) and then move randomly across the domain.

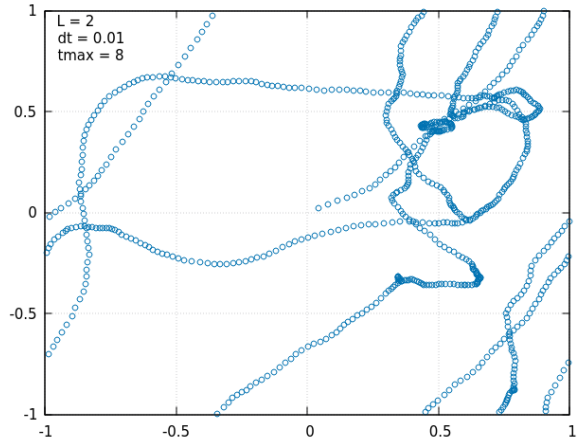

Figure 1: small domain size

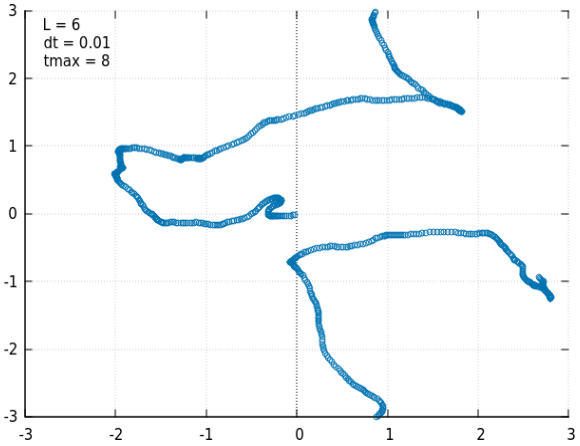

Figure 2: medium domain size

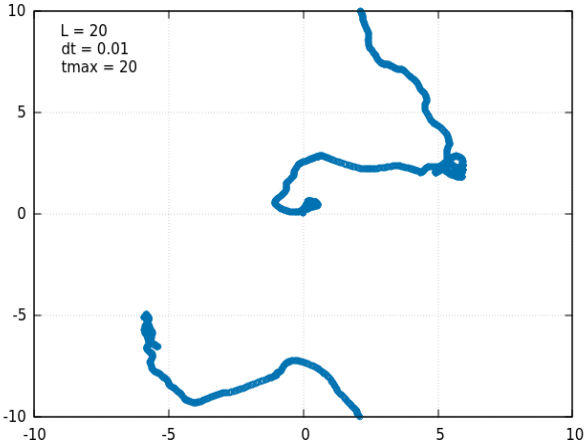

Figure 3: large domain size

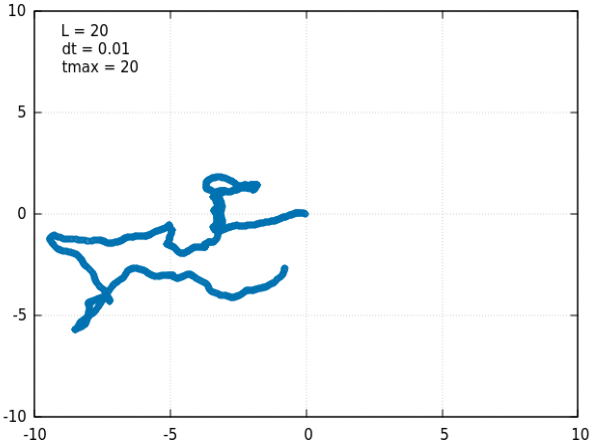

Figure 4: large domain size

- **Pseudo-Random Forcing Implementation Test:**
  We are using *Intel MKL*[3] library to generate random numbers with normal (Gaussian) distribution to represent noise (η) in the original Langevin equation. To show whether all particles are receiving different pseudo-random forcing, we draw the graph of the force received by each particle, we are using 10,000 particle for this simulation and the random force received by each particle is demonstrated for one iteration (figure 5) as well as all iterations of time (figure 6).
  The y-axis shows the frequency of the force received by each particle. As demonstrated in the following plots, the whole distribution is Gaussian and better results are achieved when considering the whole simulation:

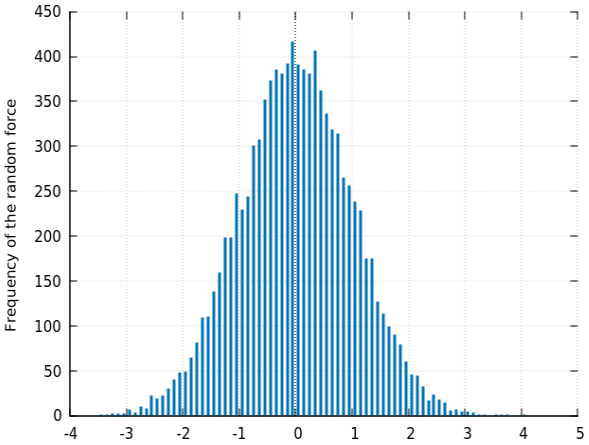

Figure 5: Force distribution - single iteration

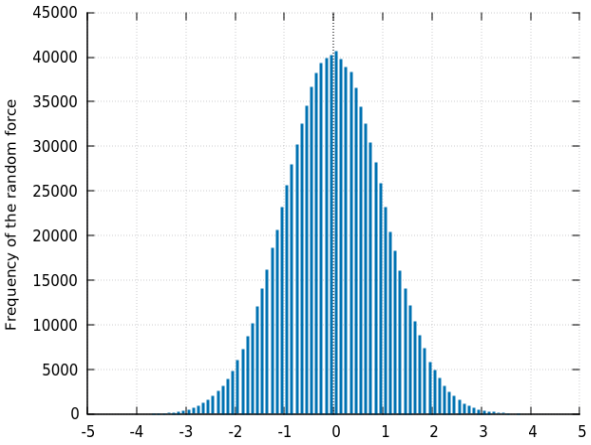

Figure 6: Force distribution - total

- **Particles Distribution Test:**
  If the size of the domain is large enough and the time-duration of our simulation is measured after a relatively long time($tmax$) then we would expect the particles distribution to be gaussian (*figure 7*). If the size of domain is small, lots of particles would escape the boundaries and due to the effect of the periodic boundary condition they appear on the other side of the domain, which after a while makes the distribution of the particles more homogeneous than gaussian (*figure 8*).
  The spike at position 0 is due to the fact that in our simulations, all the particles start from point (0,0).

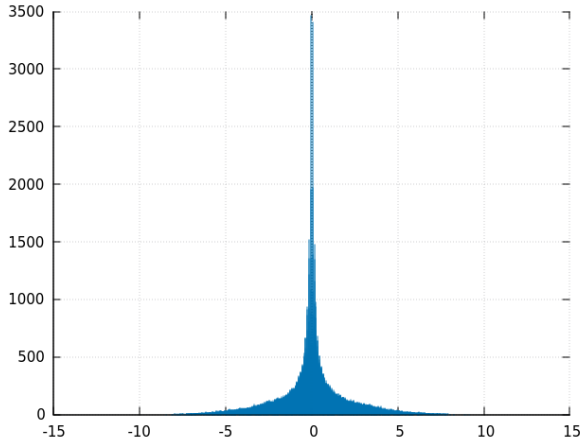

Figure 7: large domain, L = 10

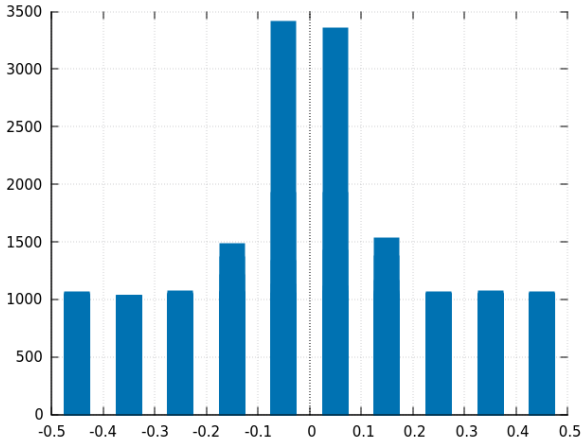

Figure 8: small domain, L = 1

Particles Distribution, n = 10,000

To show the homogeneous distribution of the particles in a relatively small domain, we can calculate the root-mean-square displacement of the particles (*figure 8*). We can see that the root-mean-square displacement of the particle gets saturated over time:

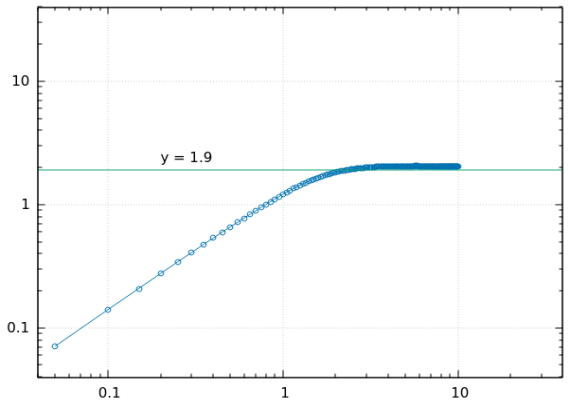

Figure 9: root-mean-square displacement, domain size = 5

If the size of the domain is small enough or the simulation is measured after a sufficiently long time, one is able to calculate the saturation rate at time $tmax$:

$$\int_{\frac{-L}{2}}^{\frac{L}{2}} \int_{\frac{-L}{2}}^{\frac{L}{2}} \sqrt{x^2+y^2}\ P\ dy\ dx$$

Where P is the probability density function (PDF) for a particle in two dimensional space and is equal to $\frac{1}{L^2}$ when the particles displacement becomes saturated. The saturation rate in *figure 9* for $L = 5$ is measured: *(almost equal to the horizontal line drawn in figure 9)*

$$\int_{-2.5}^{2.5} \int_{-2.5}^{2.5} \sqrt{x^2+y^2}\ \frac{1}{25}\ dy\ dx = 1.91$$

Compute Autocorrelation Function:

Velocity Autocorrelation Function (VAF) involves the velocities of the particles at two different time. To compute VAF we compare each particle's velocity with its velocity at the time origin:

$$A(\tau) = \frac{< v(0).v(\tau) >}{< v(0).v(0) >}$$

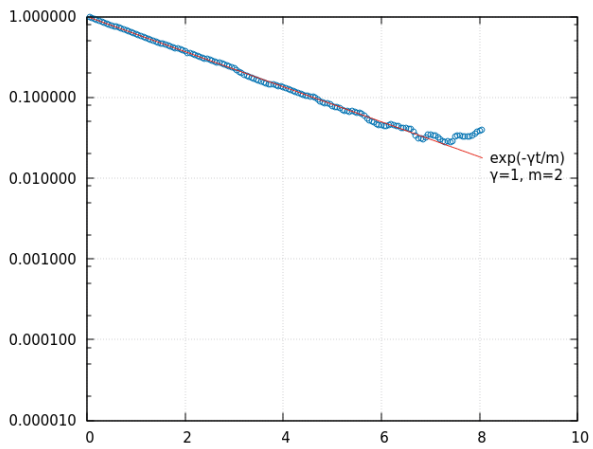

Figure 10: $\frac{\gamma}{m} = 0.5$

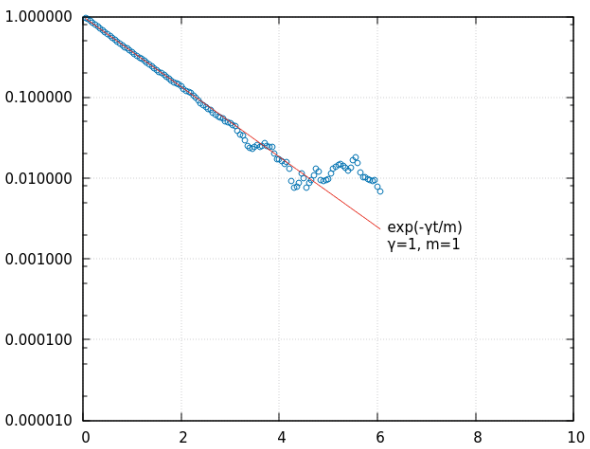

Figure 11: $\frac{\gamma}{m} = 1$

Theory wants it that A decays as $e^{-\gamma t/m}$. The red line in both figures above shows $e^{-\gamma t/m}$ based on the value of γ and m. Meaning that, as time grows we expect to experience less correlation between the velocity of a particle at the time origin and the its velocity at the measured time. Comparing *figure 10* with *figure 11* one can conclude that smaller values for γ/m means higher velocity correlation over time; that means larger values for m (more inertia) and smaller values for γ (less drag force) result in more correlation between the velocity of particles over time.
We know that the mean-square displacement of the particle gets saturated after a sufficiently long time at which the auto-correlation function of the particles decays to zero. At this point, the VAF values no longer match the $e^{-\gamma t/m}$ graph, as shown in both graphs above.

We can demonstrate ballistic-diffusive-saturated regimes by graphing the root-mean-square displacement of the particles over a long time, showing transition from ballistic (*slope = 1*) to diffusive (*slope = 0.5*) scaling at a time scale γ/m:

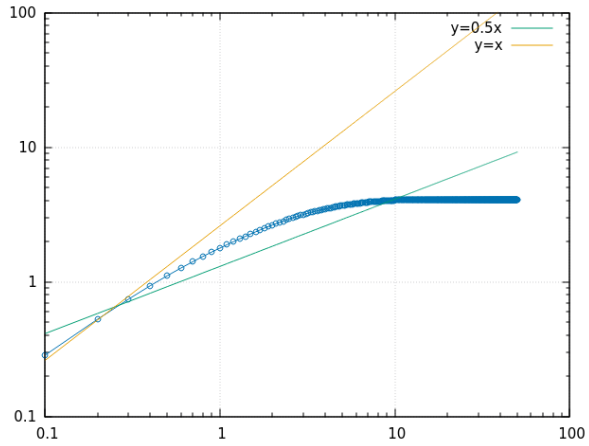

Figure 12: $\frac{\gamma}{m} = 0.5$

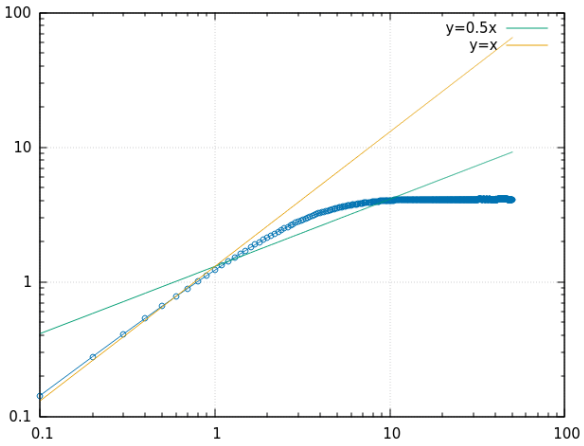

Figure 13: $\frac{\gamma}{m} = 1$

With $\frac{\gamma}{m} = 0.5$ (*figure 12*) the transition from ballistic to diffusive happens at $t = 0.5$, while with $\frac{\gamma}{m} = 1$ (*figure 13*) it happens at $t = 1$.

References

[1] A Guide to Vectorization with Intel® C++ Compilers,
https://software.intel.com/sites/default/files/m/4/8/8/2/a/31848-CompilerAutovectorizationGuide.pdf

[2] The Velocity Verlet algorithm,
https://en.wikipedia.org/wiki/Verlet_integration#Velocity_Verlet

[3] MKL Random Number Generators,
https://software.intel.com/en-us/node/521842