MCSC 6030G : High Performance Computing
Assignment 3: Langevin Dynamics of Interacting Particles

Parikshit Bajpai
100693928

1 Introduction

In 1908, Paul Langevin successfully applied Newtonian dynamics to a Brownian particle to come up with what is now called as the 'Langevin equation', the stochastic physics equivalent to the second law of motion ($F = ma$). This stochastic differential equation is applicable to continous Markov processes and shows that the root mean square dispalcement of a Brownian particle is proportional to the square root of time. The Langevin equation contains both viscous and random forces with the two forces related by the fluctuation-dissipation theorem. In the modern notation, Langevin equation takes the following form:

$$m\,dv = -\gamma\,v\,dt + \sqrt{dt}\,c\,\eta(t) \tag{1}$$

where the velocity of the particle, $v \in \mathbb{R}^2$ in a two-dimensional space, γ denotes the viscosity contribution of the force, and the random force variable η is drawn from a standard normal distribution. The random force coefficient equals, from the fluctuation-dissipation theorem, $c = \sqrt{2\gamma k_B T}$, where k_B is the Boltzmann constant and T is the temperature.

In Langevin dynamics, at short time scales, i.e. $t \ll \tau_p$, where $\tau_p = m/\gamma$ is the momentum relaxation time of particles, the dynamics of a Brownian particle is dominated by its inertia, and, the motion is this region is termed as *ballistic Brownian motion.* At larger time scales, i.e. $t \gg \tau_p$, *diffusive Brownian motion* is observed. When the particles are constrained in a box, we observe a third region beyond the diffusive region. In this region the RMS displacement saturates and we observe a plateau and a constant RMS displacement.

2 Methodology

2.1 Objective

In the present study, Langevin dynamics studied in the previous assignment has been further analysed by including interaction forces between the particles and implementing domain decomposition and parallelisation. The objective of this study is to analyze the Brownian motion of interacting particles with domain decomposition and multi-threading, and, to study the impact of model parameters on the efficiency of parallelization.

2.2 Machine Configuration

The initial computations for the present study were performed on the local machine followed by detailed study on HPC systems accessible through *Compute Canada.*

2.2.1 Local Machine

Manufacturer & Model: Lenovo ThinkPad Yoga 370
Processor: Intel Core i5 -7200U (2 physical cores, 2 hyperthreads)
Clock Rate: 2.50 GHz
RAM: 16 GB
Operating System: Ubuntu 18.04

2.2.2 GRAHAM Cluster

Manufacturer: Huawei
Processor: Intel E5-2683 v4 (Broadwell) (Total 36160 cores, maximum 32 cores used in the study)
Clock Rate: 2.10 GHz
RAM: Node specific
Operating System: CentOS 7

2.3 Implementation

The Langevin equation was discretised using the Velocity Verlet algorithm, which is similar to the Leapfrog method, except that the velocity and position are calculated at the same value of the time variable. The mathematical formulation of the algorithm is shown below.

$$\begin{aligned}\mathbf{x}(t+\Delta t) &= \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 \\ \mathbf{v}(t+\Delta t) &= \mathbf{v}(t) + \frac{\mathbf{a}(t)+\mathbf{a}(t+\Delta t)}{2}\Delta t\end{aligned} \tag{2}$$

While an implementation of the above algorithm essentially involves computing forces (which manifest themselves in the code in form of accelerations) and updating velocities and positions at each time step, additional book-keeping tasks must be performed in order to include domain decomposition.

The principle behind domain decomposition is reducing the computation time by avoiding operations whose results are a-priori known. It has been well established that the interaction forces between particles at a distance larger than a critical distance, r_c, are effectively zero. By selecting a suitable sector size, *Domain Width* $\geq R_c$, the number of interactions to be computed can be minimized to interactions between particles in the same and surrounding sectors, therby resulting in a decrease in wallclock times. Furthermore, each sector can then, at least in theory, be alloted to a separated thread in order to implement multi-threading. The overheads in domain decomposition arise from the requirement to maintain neighbour lists and to account for the particles and updating their sectors at each time step.

In the present work, reflecting boundary conditions have been implemented along with the decomposition of domain into $D \times D$ subdomains/sectors. The following pseudo-codes exhibit the algorithms required to develop a Langevin dynamics code for interacting particles with reflecting boundary conditions and domain decomposition.

Algorithm 1: Main

```
Data: Number of particles ($N$), Number of rows and columns ($d$)
Result: Compute RMS displacement and trajectories
Allocate and initialize all variables
Create neighbour list
Initialize particles and perform ordering
while $t < t_{max}$ do
    Update half-step velocities $\vec{v}_h(t + \Delta t/2) \leftarrow \vec{v}(t) + 0.5\vec{a}\Delta t$
    Update particle positions $\vec{x}(t + dt) \leftarrow \vec{x}(t) + \vec{v}_h(t + \Delta t/2)\Delta t$
    Impose reflecting boundary conditions
    Reorder particles according to sector
    Compute random forces acting on particle and update particle acceleration $\vec{a}$
    for Sector $S1 = 0, 1, \cdots, d^2 - 1$ do
        for Each particle $i$ in sector $S1$ do
            for All neighbours $S2$ of sector $S1$ do
                for Each particle $j$ in sector $S2$ do
                    Compute distance between particles $i$ and $j$
                    if Distance between $i$ and $j$ < Critical distance $r_C$ then
                        Compute interaction force $\vec{F}_i$
                        Update particle acceleration $\vec{a}_i \leftarrow \vec{a}_i + \vec{F}_i/m$
    Update velocities $\vec{v}(t + dt) \leftarrow \vec{v}_h(t + \Delta t/2) + 0.5\vec{a}\Delta t$
    Update time $t = t + \Delta t$
    Write particle position $\vec{x}$ to 'Trajectories'
    Write RMS displacement $\sqrt{\frac{\sum_{i=1}^{N}(\vec{x}-\vec{x}_0)^2}{N}}$ to 'Means'
```

Algorithm 2: Initialization

```
Data: Particle position ($\vec{x} = \langle x, y \rangle$), Particle velocity ($\vec{u} = \langle u_x, u_y \rangle$), Particle acceleration
      ($\vec{a} = \langle a_x, a_y \rangle$)
Result: Set physical parameters and random initial position and velocity of particles
Initialize physical parameters
for Particle $i = 1, 2, \cdots, N$ do
    $x_i \leftarrow$ Uniform random number $\in (-L/2, L/2)$
    $y_i \leftarrow$ Uniform random number $\in (-L/2, L/2)$
    $u_{x_i} \leftarrow$ Normal random number $\in \mathcal{N}(0, \frac{K_B T}{m})$
    $u_{y_i} \leftarrow$ Normal random number $\in \mathcal{N}(0, \frac{K_B T}{m})$
    $a_{x_i} \leftarrow 0$
    $a_{y_i} \leftarrow 0$
```

Algorithm 3: Reflecting Boundary Conditions

```
Data: Particle position ($\vec{x} = \langle x, y \rangle$), Particle velocity ($\vec{u} = \langle u_x, u_y \rangle$), Box length ($L$)
Result: Updates particle position and velocity if outside the box
for $i = 1, 2, \cdots, N$ do
    if $x_i < -L/2$ then
        $x_i \leftarrow -L - x_i$;
        $u_{x_i} \leftarrow -u_{x_i}$;
    if $x_i > L/2$ then
        $x_i \leftarrow L - x_i$;
        $u_{x_i} \leftarrow -u_{x_i}$;
    if $y_i < -L/2$ then
        $y_i \leftarrow -L - y_i$;
        $u_{y_i} \leftarrow -u_{y_i}$;
    if $y_i > L/2$ then
        $y_i \leftarrow L - y_i$;
        $u_{y_i} \leftarrow -u_{y_i}$;
```

Algorithm 4: Assign particle sectors

```
Data: Particle position ($\vec{x} = \langle x, y \rangle$)
Result: Sector indices and number of particles in each sector
for Particle $i = 1, 2, \cdots, N$ do
    if Particle within domain ($-L/2 \leq x \leq L/2$ and $-L/2 \leq y \leq L/2$) then
        Assign sector to the particle;
        Number of particles in sector $\leftarrow$ Number of particles in sector + 1;
    else
        Put the particle in garbage bin;
        Number of particles in garbage $\leftarrow$ Number of particles in garbage + 1;
```

Algorithm 5: Ordered list of particles

```
Data: Particle position ($\vec{x} = \langle x, y \rangle$)), Particle velocity ($\vec{u} = \langle u_x, u_y \rangle$)
Result: Particles ordered according to the sector they are in.
for Particle $i = 1, 2, \cdots, N$ do
    Sort the particles according to their sector.
```

Algorithm 6: Neighbour list

```
Data: Neighbour list
Result: List of neighbours of each sector
for Sector $k = 1, 2, \cdots, d^2 - 1$ do
    for All surrounding sectors do
        if Valid sector then
            Add to neighbour list
```

2.4 Computational Experiments

The simulation of Brownian motion depends on a number of factors, both physical and computational. These parameters not only affect the observed behaviour but also the computational times as discussed in the following paragraphs.

The number of particles, N, is the most important factor affecting the computations and wallclock times. As seen above, the wallclock times for the case without domain decomposition are proportional to N^2 and therefore if we increase the number of particles in the system by 2, we will observe an increase in the wall times by a factor of 4. In order to study the impact of the other parameters, we can then fix the number of particles in the system. However, while doing this, we must ensure that the number of particles is not too large or too small to result in a too dense or too sparse system.

The particle density in the box affects the Brownian motion itself and the observed regions of the RMS displacements. As already seen in the previous assignment, if the particle density is too large, we'll lose the diffusive Brownian motion, and, owing to the large number of interactions, the diffusive region might not be computationally resolved and a direct transition from ballistic to saturation region will be observed. The particle density in the box is itself a function of the number of particle in the domain, N, and the length of the box, L. For a fixed number of particles in the domain, we want to keep the size of the box as large as possible. However, it must also be ensured that we do not make the box too large, and hence the particle density too sparse, therby imposing an upper limit on the box size.

The wallclock times for the Langevin dynamics problem depend on the maximum time of simulation, t_{max}, and the time-steps, Δt, used in the simulations. While the end time for the simulation can be chosen rather arbitrarily, in order to observe all the expected displacement scales, we must make sure that t_{max} is sufficiently large while also making sure that we don't keep it much larger than the saturation time as there won't be any gains in terms of additional physical information that might be obtained but would lead so an increase in wallclock times. With a fixed t_{max}, smaller the time-steps, larger the number of operations and hence longer the wall-clock times will be. However, the time stepping also plays an important role in preventing the loss of particles from the domain. If the time-steps are too large, fast moving particles, i.e ones with velocities much larger than $L/\Delta t$ have a probability of travelling too far outside the box to be moved back in by the reflecting boundary conditions.

Two important physical parameters releveant to the interaction forces are the radius of the particles, σ, and the distance between the particles, R, and the interaction force between particles i and j is given as

$$F_{i,j} = \frac{4\epsilon}{R}\left(-12\left(\frac{\sigma}{R}\right)^{12} + 6\left(\frac{\sigma}{R}\right)^{6}\right)$$

where ϵ denotes the depth of the potential well. The critical radius R_C denotes the minimum distance between the particles at which the interaction forces can be approximated to zero and is normally taken as 2.5σ. In order to minimize the particles being lost due to extremely large interaction forces and resulting velocities, we must ensure that σ is not large and an upper limit of possible values of particle radius can be obtained. A similar argument imposes a lower limit on the possible values of R. The argument can, in other words, be translated in terms of particle density in the box. For a box of fixed size, the number of particles in the box can only be increased such that the particle density is not too high to lead to very high interaction forces and thus loss of particles.

Domain decomposition and multi-threading provide two additional parameters to optimise in order to reduce wallclock times, namely the number of rows/columns, D, and the number of threads,

$N_{Threads}$. As already seen, domain decomposition into $D \times D$ sectors helps reduce the order of floating point operations at each time step by a factor of D^4, i.e., with domain decomposition the wallclock times are expected to be proportional to $\left(\frac{N}{D^2}\right)^2$. Although book-keeping overheads consume a bit of the reduced time, their computations are of lower order compared to the computations using the velocity verlet and we can still observe an appreciable reduction in the wallclock times. However, domain decomposition follows the law of diminishing returns and we observe that beyond a number of domains, the addition of more rows/columns does not result in any significant time gain and might even make the times worse. From the point of view of implementation, to ensure that the interactions are limited to particles within a sector and in the neighbouring sectors, we must ensure that $Domain\ Width \geq R_c$, and an upper limit on the maximum number of sectors is obtained.

With regards to the number of threads, $N_{Threads}$, we expect a linear decrease in the awall times as the number of processors are increased. This behaviour should be observed till the number of threads equals the number of sectors after which any additional thread would sit ideal. However, since in some cases all the threads might not share the same number of sectors, we expect a departure from the linearly decreasing trend.

To obtain an initial benchmark for the number of particles in the system, a number of computations were performed and the number of particles in the system was fixed to 1.3×10^6 which corresponds to a wallclock time of approximately 1 h. The length of the box and the simulation end time were then fixed to 1000 and 10 to ensure that all the desired motions, i.e. ballistic, diffusive and saturation, were observed. An upper limit on the maximum number of rows/columns was obtained as 890 based on the considerations above. The time step size was selected based on the number of particles lost at the end of the simulations and a short discussion about the step sizes is in the results and discussions section. The step size was varied and the number of particles lost at the first and last time steps was used to fix the step size to 0.01 for further studies.

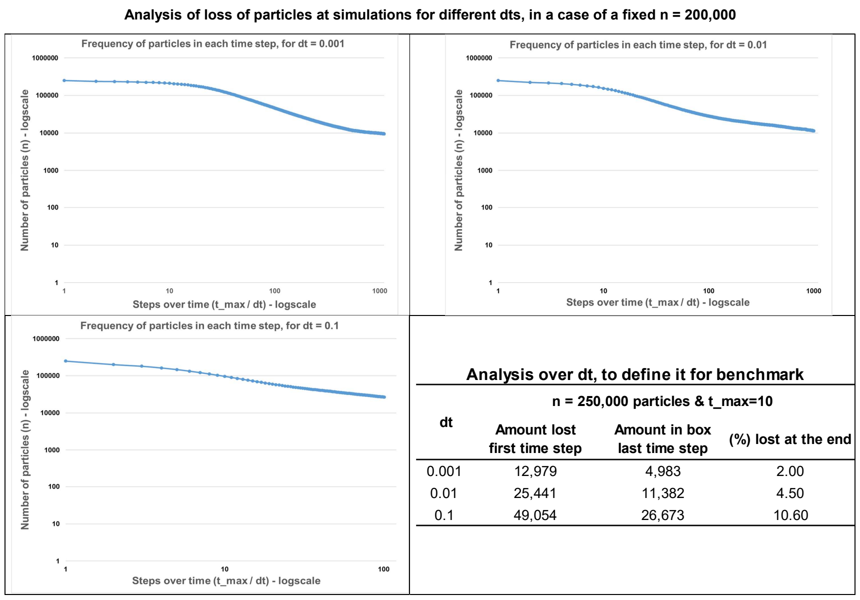
3 Results and Discussion

The number of particles at the first time step and by the end of the computation have been shown in figure 1 and shows that while the number of particles lost at the first time step are lower for smaller time steps, the number of particles lost by the end of the computation actually increase with decreasing time steps. However, since a very large time step would lead to a loss of a significant part of the particles at the first step itself, the time step δt was chosen through a compromise between the two. Subsequently, the times steps were fixed to 0.01 for all the computations.

For a fixed number of sectors, the wall times are expected to increase proportionally with respect to N^2. This behaviour was studied by fixing the number of sectors to 64 and varying the number of particles in the box. The results have been been presented in figure 2.

For a fixed number of particles, the wall times are expected to decrease proportionally with respect to D^4. This behaviour was studied by fixing the number of particles and varying the maximum number of rows/columns and has been presented in figure 3. While the behaviour is as expected for the initial part of the plot, we observe that after a certain number of sectors, as the number of sectors is further increased the wall times decrease at a reduced rate. This can be explained by increase in the overheads as the number of sectors is increased. The book-keeping costs, however, become significant only for a significantly large number of sectors as can be seen from the given plots.

For a fixed value of number of particles N and number of sectors N^2, the linear decrease in wallclock times with an increase in the number of threads has been presented in figure 4. While



Analysis over dt, to define it for benchmark

n = 250,000 particles & t_max=10

dt	Amount lost first time step	Amount in box last time step	(%) lost at the end
0.001	12,979	4,983	2.00
0.01	25,441	11,382	4.50
0.1	49,054	26,673	10.60

Figure 1: Comparison of number of particles lost for different time step sizes.

a reduction in wall times with increase in number of threads was observed for this case, the trend was not exactly linear. The speed-up and efficiency for the case have been presented in table 1

Table 1: Speed-up and efficiency of parallelisation for $N = 1.3 \times 10^6 \& D = 8$.

Number of Threads	Wallclock Time [s]	Parameter	
		Speed-up	Efficiency
01	1717.20		
02	931.04	1.84	0.92
04	561.92	3.06	0.76
08	345.92	4.96	0.62
16	231.27	7.42	0.46
32	180.85	9.50	0.30

4 Conclusion

In conclusion, we can say that a number of physical and computational parameters affect the wall-clock times of the Langevin dynamics simulation. While the number of particles significantly impacts the wall-clock times, significant speed-ups can be obtained by domain discretisation and multi-threading. For a fixed number of particles, we observed that a minimum of 400 sectors with 2 threads gives an efficiency of at least 80% and while we observe a continuous speed-up as the number of threads is increased, there is a loss of efficiency. With 32 threads, we found out that a

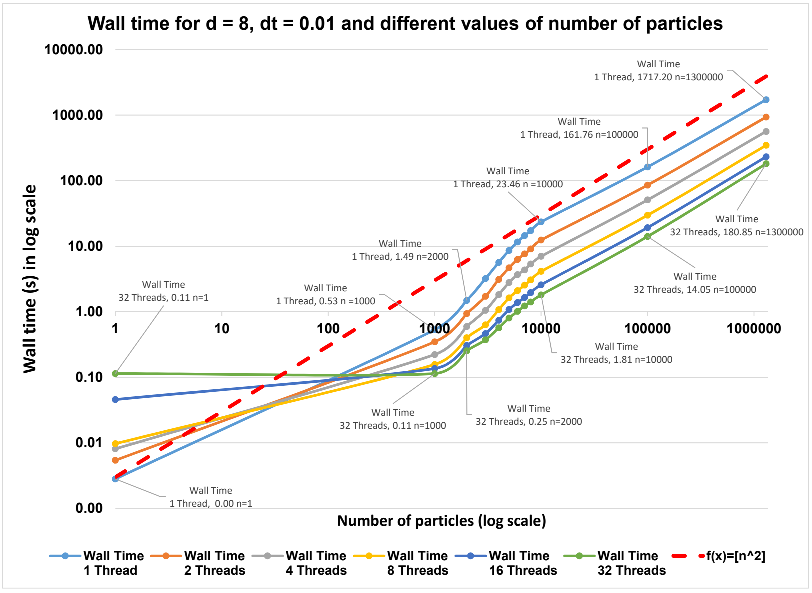


Figure 2: Change in wallclock times for different number of particles.

minimum of 4000 particles were required to have an efficiency of 30%.

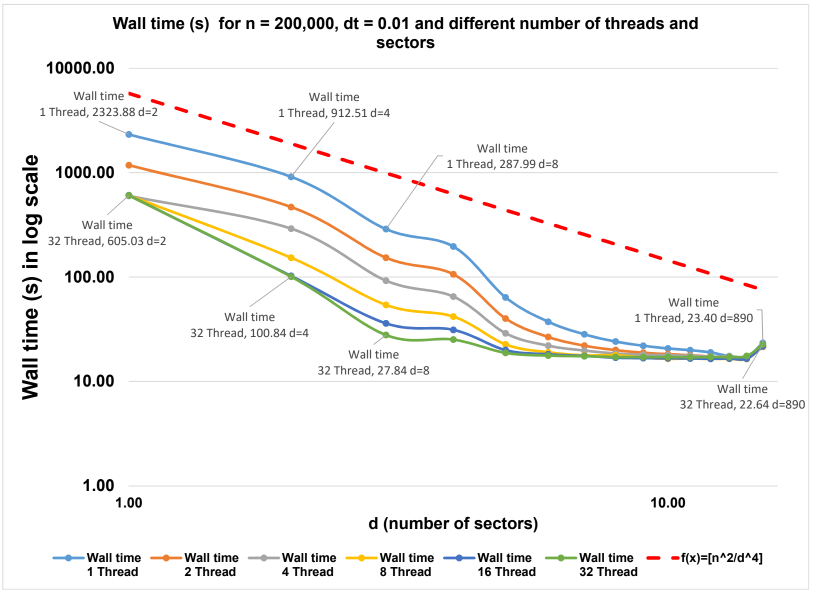


Figure 3: Change in wallclock times for different number of sectors.

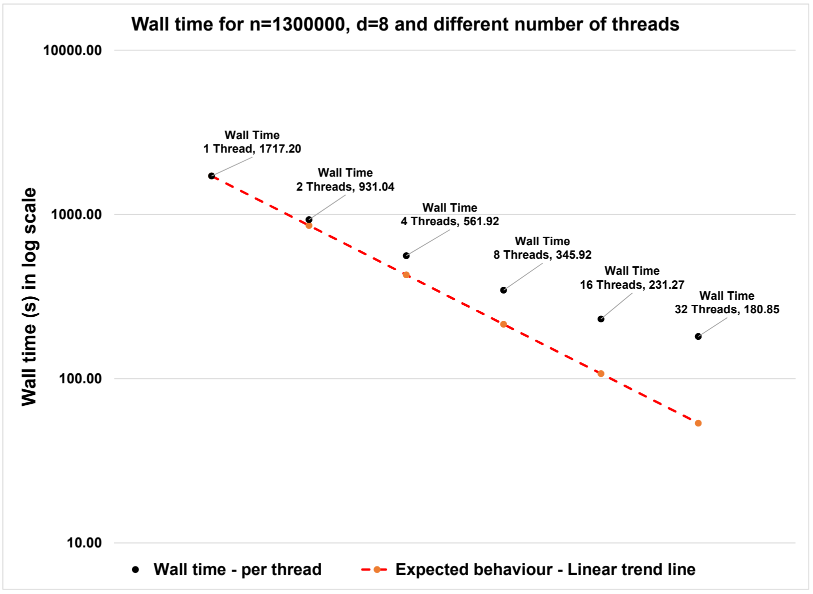


Figure 4: Change in wallclock times for different number of threads.