Soft Particle Molecular Dynamic Simulation

Soniya Kumawat

Department of Chemical Engineering, Indian Institute of Technology, Kanpur, 208016, Uttar Pradesh, India.

Abstract

Purpose of this study to develop the Discrete Element Method algorithm to simulate dense granular flows using soft particle model (time driven approach , where particles advance by a fixed time step and this works significantly well in dense flow regime) in C programming language. Linear spring and dash pot force model was implemented to calculate forces on particles. Particles positions and velocities are updated using velocity verlet integration scheme. To Optimize the number of interactions of a particle with all the other particles, the simulation box is divided in different bins and all the neighbouring cells were maped with each other then a linklist created used to link all the particles in a cell. For any particle only interaction within the current cell and neighbouring cells are considered.

1 Introduction

Granular materials are collection of discrete macroscopic particles which are commonly found in nature, and has wide application in many geophysical situations (avalanches, landslides, lahars (volcanic mud-flow), etc.) as well as in industries such as pharmaceutical, food processing, fertilizer, polymer, ceramic, glass, mining & minerals, steel, agricultural, cement industries, etc. Individual particles are modeled either as non-deformable hard particles or as deformable soft particles. In the hard particle model, Particles are considered infinitely stiff and hence the collision time is zero. As a result, only one collision or "event" is allowed at a given instant. This also implies that only binary collisions are permitted and multi-particle contacts cannot be accounted for. In the soft particle model, particles interact with each other for a finite time and deform during the interaction. The simulation advances by time steps small enough to capture each collision in several fine steps. Soft particle model is more flexible than hard particle model.It is able to model multiple, long lasting contacts as well as binary contacts. Due to small integrating time step , Soft particle model is more time consuming than hard particle model. In this project, I have implemented an algorithm using soft particle approach with linear spring and dash pot force model in C. Code is designed using different functions which are mentioned below:

- Makemap() - Create a map for every cell.
- Neblist() - Find the neighbours of all particles.
- Linklist() - Create links between all the particles within a cell.
- SortNeb() - Sort all the neighbouring particles ID's for each particle
- ForceCal() - Compute Particle - particle interaction forces , particle interaction force with wall (Linear spring and dashpot model).
- Move() - Updating velocity and position using velocity verlet algorithm.
- Boundary() - Periodic boundary condition is applied
- Profile() - Compute the center of mass of species and average properties.
- Heap() - Compute kinetic energy & move the by time step N_{tmp}
- Initialization() - Initialize all parameters (read ch3d.par & pos.dat)
- Writeall() - Write properties in a bin.
- Main() - Simulate the system for given time i.e, t_{final}

2 Soft Particle Model

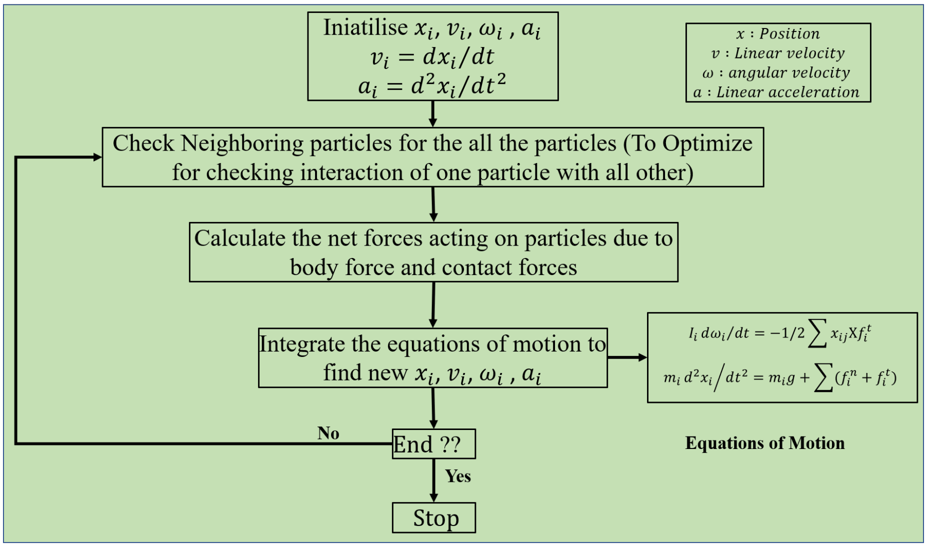

Iniatilise x_i, v_i, ω_i , a_i
$v_i = dx_i/dt$
$a_i = d^2x_i/dt^2$
x : Position
v : Linear velocity
ω : angular velocity
a : Linear acceleration
Check Neighboring particles for the all the particles (To Optimize for checking interaction of one particle with all other)
Calculate the net forces acting on particles due to body force and contact forces
Integrate the equations of motion to find new x_i, v_i, ω_i , a_i
$I_i \, d\omega_i/dt = -1/2 \sum x_{ij} X f_i^t$
$m_i \, d^2x_i \big/ dt^2 = m_i g + \sum (f_i^n + f_i^t)$
Equations of Motion
No
End ??
Yes
Stop

Fig. 1 Soft Particle model Algorithm

3 Periodic Boundary

Periodic boundary conditions are used for approximating a large (infinite) system by using a small system. In Our Code we can apply periodic bounday condition in any direction by setting the $wallflag$ variable to 2.

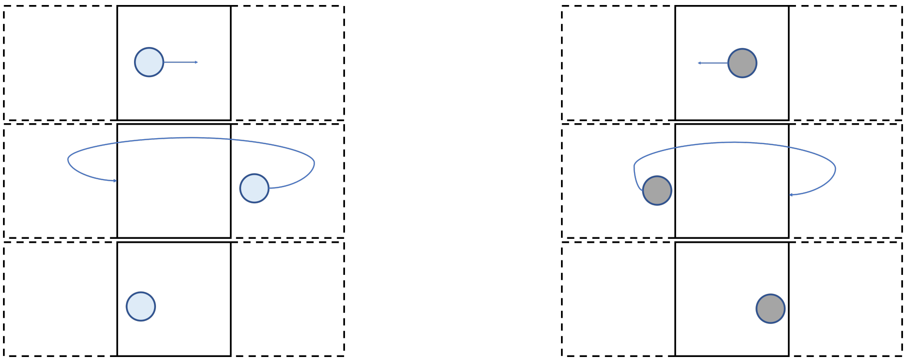


Fig. 2 A particle crossing wall, re-enter it from the opposite side

The condition is implemented by the following algorithm -

Algorithm 1 Periodic Boundary Condition

```
1: let the dimension of the box is $L_i$ and particle center is at $X_i$ in $\hat{i}$ direction.
2: if $X_i < 0$ then
3:     $X_i \Leftarrow X_i - L_i$ re-enter the particle from other side
4: end if
5: if $X_i > L_i$ then
6:     $X_i \Leftarrow X_i + L_i$ re-enter the particle from other side
7: end if
```

4 Velocity-Verlet Algorithm

Velocity verlet scheme is most popular scheme for numerical integration. The Verlet integrator provides good numerical stability at no significant additional computational cost over the simple Euler method. In Velocity verlet algorithm particle's positions are updated at time step δt and velocities are updated at time step $\delta t/2$ or we can say that velocity calculation uses average acceleration at t & $t + \Delta t$.

$$\vec{v}\left(t + \frac{\delta t}{2}\right) = \vec{v}(t) + \frac{\delta t}{2}\frac{\vec{F}}{m} \tag{1}$$

$$\vec{\omega}\left(t + \frac{\delta t}{2}\right) = \vec{\omega}(t) + \frac{\delta t}{2}\frac{\vec{T}}{I} \tag{2}$$

$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}\left(t + \frac{\delta t}{2}\right)\delta t \tag{3}$$

$$\vec{v}(t+\delta t) = \vec{v}\left(t + \frac{\delta t}{2}\right) + \frac{\delta t}{2}\frac{\vec{F}}{m} \tag{4}$$

$$\vec{\omega}(t+\delta t) = \vec{\omega}\left(t + \frac{\delta t}{2}\right) + \frac{\delta t}{2}\frac{\vec{T}}{I} \tag{5}$$

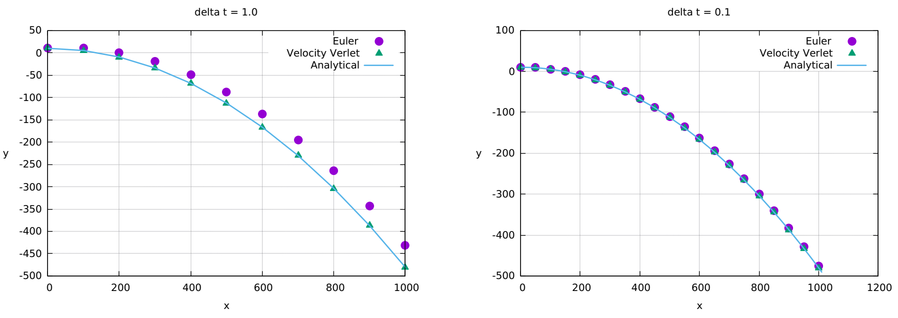


Fig. 3 [From Assignment 3] Comparison of Euler, analytical and velocity-verlet algorithm for a 2-d projectile for different time step

Figure 3 shows the comparison of velocity-verlet, Euler methodIn with analytical solution for the trajectory of a 2-d projectile case. The velocity-verlet numerical scheme results are matches with analytical results for larger δt whereas Euler Scheme differ from analytical solution. Hence, we used the velocity-verlet algorithm for high accuracy.

5 Force Model

Linear Spring and Dashpot model is used both in normal and tangential direction to account for the particles interaction.

Consider two particles i and j having position vector as $\vec{x}_i$ and $\vec{x}_j$. The translational velocities of the two particles are $\vec{v}_i$ and $\vec{v}_j$ respectively and the rotational velocities of the particles are $\vec{\omega}_i$ and $\vec{\omega}_j$.

The relative position vector is defined as

$$\vec{x}_{ij} = \vec{x}_i - \vec{x}_j. \tag{6}$$

Unit normal vector from j to i

$$\hat{n} = \hat{x}_{ij} = \frac{\vec{x}_{ij}}{\|\vec{x}_{ij}\|} \tag{7}$$

The distance between the centers of the particles is given as

$$d = \|\vec{x}_{ij}\| \tag{8}$$

The two particles will be overlapping if $d < (R_i + R_j)$. Overlap between the particles in the normal direction is

$$\delta = R_i + R_j - d \tag{9}$$

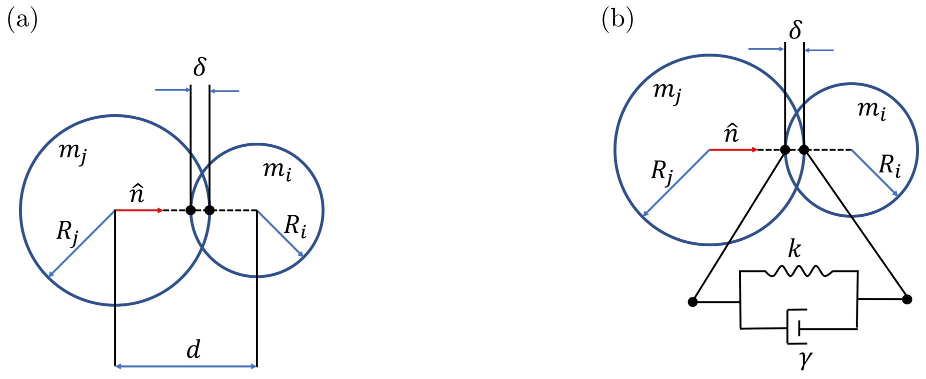


Fig. 4 (a) Two particles interactions (b) Force Model

The translation relative velocity of the two particles is

$$\vec{v}_{ij} = \vec{v}_i - \vec{v}_j \tag{10}$$

Component of the relative velocity vector in the normal direction

$$\vec{v}_n = \big(\vec{v}_{ij} \cdot \hat{n}\big)\hat{n} \tag{11}$$

and in the tangential direction

$$\vec{v}_t = \vec{v}_{ij} - \vec{v}_n \tag{12}$$

Accounting for the angular velocities of the particles

$$\vec{v}_t = \vec{v}_{ij} - \vec{v}_n + \vec{\omega}_i \times R_i(-\hat{n}) - \vec{\omega}_j \times R_j\hat{n} \tag{13}$$

The magnitude of contact force in normal direction is given by

$$\|\vec{F}_n\| = k_n\delta - m_{eff}\gamma_n\|\vec{v}_n\| \tag{14}$$

Contact Force in normal direction is

$$\vec{F}_n = \|\vec{F}_n\|\hat{n} \tag{15}$$

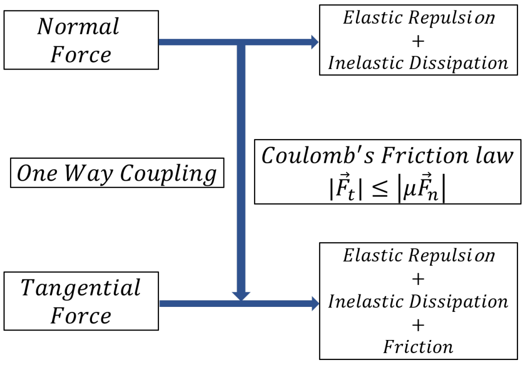
The magnitude of contact force in normal direction is given by

$$\|\vec{F}_t\| = -k_t S_t - m_{eff}\gamma_t\|\vec{v}_t\| \tag{16}$$

Contact Force in tangential direction is

$$\vec{F}_t = \|\vec{F}_t\|\hat{t} \tag{17}$$

There exist a one way coupling between normal and tangential direction due to frictional nature of the particles



Normal Force
Elastic Repulsion + Inelastic Dissipation
One Way Coupling
Coulomb's Friction law $|\vec{F}_t| \leq |\mu\vec{F}_n|$
Tangential Force
Elastic Repulsion + Inelastic Dissipation + Friction

Fig. 5 Contact Force Model : One way coupling

Force on i^{th} and j^{th} particle is given by

$$\begin{aligned}\vec{F}_i &= m_i\vec{g} + \vec{F}_n + \vec{F}_t\\ \vec{F}_j &= m_j\vec{g} - \vec{F}_n - \vec{F}_t\end{aligned} \tag{18}$$

Torque on i^{th} and j^{th} particle is given by

$$\begin{aligned}\vec{T}_i &= R_i(-\hat{n}) \times \vec{F}_t\\ \vec{T}_j &= R_j(\hat{n}) \times (-\vec{F}_t)\end{aligned} \tag{19}$$

6 Neighbouring Cells

The Simulation box is divided in different cells (square cells in $2D$ & cubical cells in $3D$). Each cell can have more than one particles. Any particle situated in a cell can interact with the other particles present in the same cell as well as in the adjacent cells. The adjacent cells in case of $2D$ and $3D$ are shown in figure 6 & 7 respectively. For $2D$ case cell-5 has 8 neighbouring cells which are $1, 2, 3, 4, 6, 7, 8, \&9$. Similarly for $3D$ case, cell-14 has 26 neighbouring cells which are

$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, \&27$. In order to avoid the double counting of the neighbouring particles, we consider only half of the neighbouring cells as neighbours.

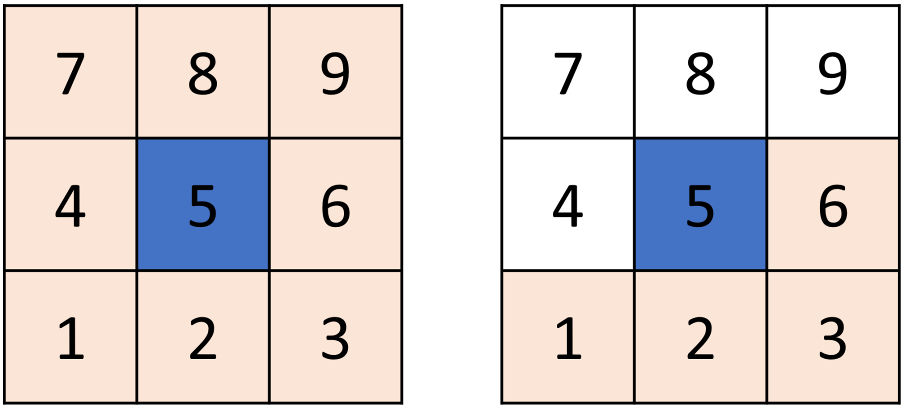


Fig. 6 (2-D) Neighbouring cell of 5th cell, only half of the cells are considered to avoid repetition

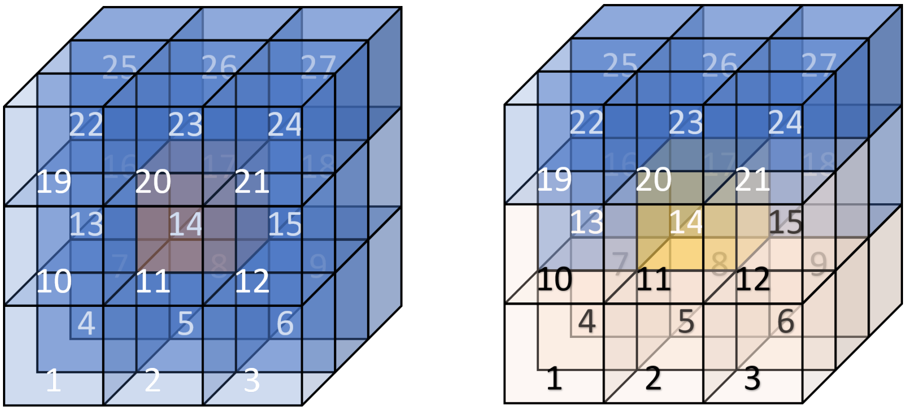


Fig. 7 (3-D) Neighbouring cell of 14th cell, only half of the cells are considered to avoid repetition

7 Make-Map

The simulation box is divided in different cells/bins of size $rng \times R_{largest}$ along all the directions. Each bin has $3^3 - 1 = 26$ neighbouring bins in $3D$ and $3^2 - 1 = 8$ in 2D. To avoid the double counting of the neighbours, Each cell is mapped with only half of bins as the neighbouring cells (4 in case of $2D$ and 13 in case of $3D$). The case of periodic or sidewalls is shown in figure 9 & 10.

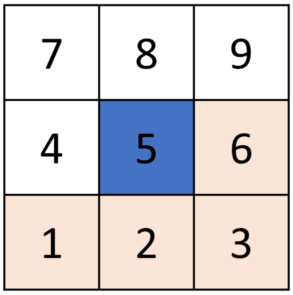


Fig. 8 Neighbouring cell of 5th cell, only half of the cells are considered to avoid repetition

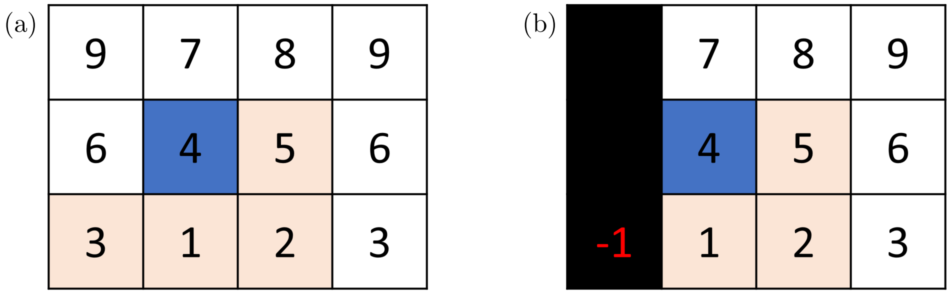


Fig. 9 Neighbouring cell of 4th cell,(a) if periodic boundary condition is applied, (b) if wall is present on left of cell 4th, -1 represent that there is no neighbouring cell at that place. only half of the cells are considered to avoid repetition

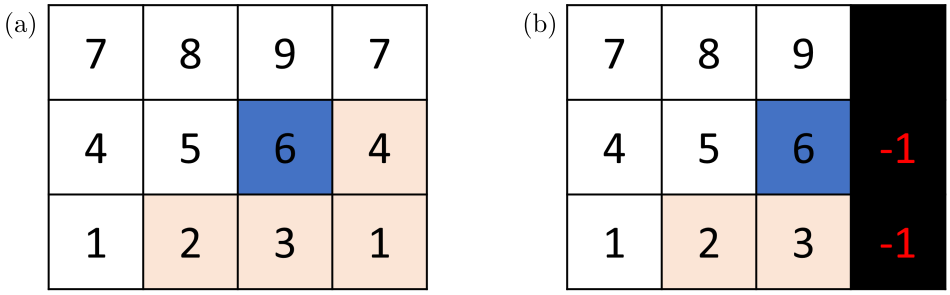
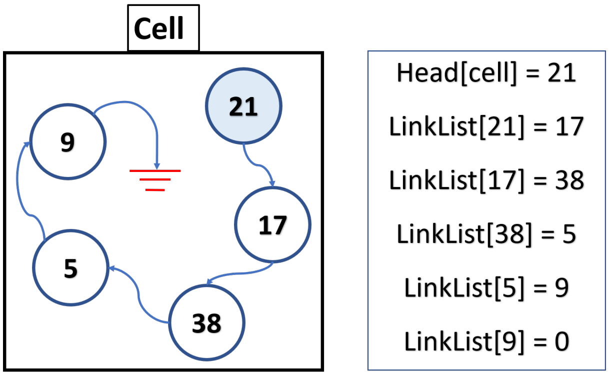


Fig. 10 Neighbouring cell of 6th cell,(a) if periodic boundary condition is applied, (b) if wall is present on right of cell 6th, -1 represent that there are no neighbouring cell at that place. only half of the cells are considered to avoid repetition

8 Link List

The function Link List links all the particles present in a cell. The head variable contains the ID of the first particle in the cell and then the llist array is used to link all the particles in that cell, if the llist of any particle gives zero value, it means that particle is the last particle present in that particular cell. The last particle in the link list is not linked to any other particle in the bin, called tail particle. This can be understand by the following figure



Head[cell] = 21
LinkList[21] = 17
LinkList[17] = 38
LinkList[38] = 5
LinkList[5] = 9
LinkList[9] = 0

Fig. 11 Representation to link all the particles in a bin

9 Neighbour Search

First we take the head particle of of the cell and then check if the remaining particles in this cell are neighbours of this particle or not. Once this cell is completed, next check with all the particles present in the neighbouring cells.

If two particles are within the range of each other, we increase the number of neighbour of any one particle and store the ID of that particle. We update the neighbour for any of one particle to avoid repetition.

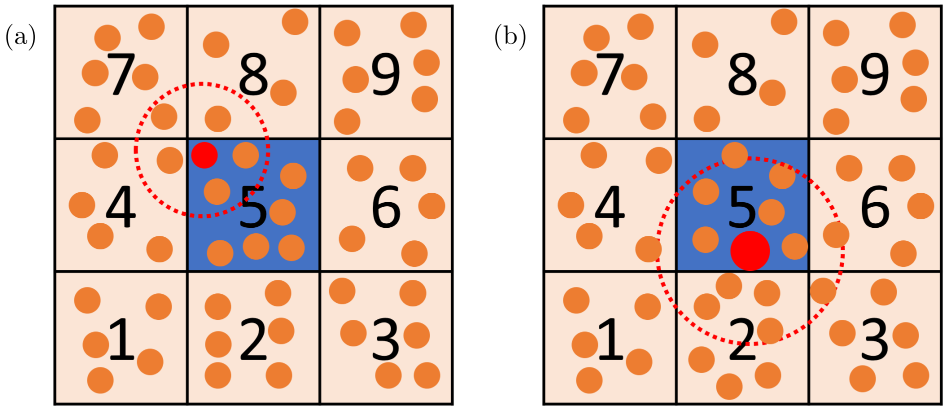


Fig. 12 Neighbouring particles in a minimum range which is specified based on the particle diameter (a) for small particle (b) for large particle

10 Results

https://drive.google.com/file/d/1CoppvghqMdlYsu-
0SjIXopTgP09rsTyh/view?usp=sharing

11 Future Work

- Use extern to declare variables in c language to declare all the variables in one global file, so that any variable is directly accessible to the header files.
- Verification of code for large systems
- The code is checked for 2D case and by taking only 2 particles, further iterations may be needed to before using the code for 3D case.

12 Acknowledgement

I would like to express my special thanks of gratitude to my supervisor **Dr. Anurag Tripathi** who gave me the golden opportunity to do this wonderful project on the topic **Distinct Element Method Soft Particle Molecular Dynamic Simulation** which also helped me in doing a lot of research and I come to know about so many new things. Secondly I would also like a thank to my lab-mates Satyabrata patro and Vishnu Sahu who helped me a lot in finishing this project within the given time. It helped me increase my knowledge and skills.

13 References

https://journals.aps.org/pre/cited-by/10.1103/PhysRevE.81.041307
DEM basic slides AT2019.pdf