Dissipative Particle Dynamics (DPD)

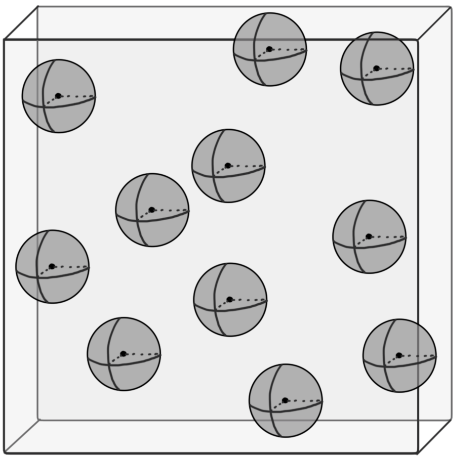
Dissipative particle dynamics (DPD) is a (Brownian) discrete element method technique for running mesoscale particle simulations.

DPD simulations have discrete elements (particles) that exist in a continuous space and move in discrete timesteps. They function similarly to (atomistic) molecular dynamics (MD) simulations, but the system is coarse-grained — particles are used to represent whole molecules, fluid regions, clusters of molecules, etc. (NOT individual atoms).

DPD was first devised and published in 1992 as a coarse-graining method for simulating particle interactions with hydrodynamic effects. It is therefore both well established (and available in most particle simulation platforms) and quite new in terms of research contributions. Even now, in 2022, most students who trained in these methods and further developed them are only just beginning to become established enough faculty members to lead large scale projects with DPD. We are on a cutting edge!

CREATING A DPD SIMULATION

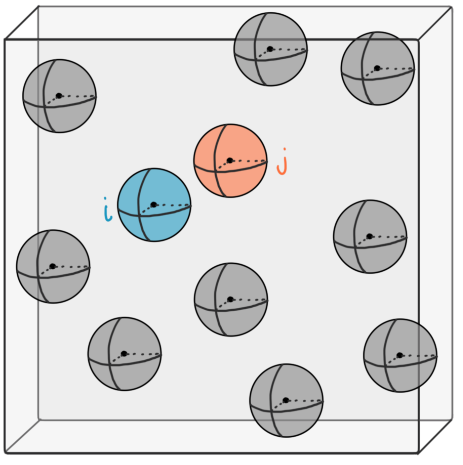
A basic DPD simulation has a large number of one type of identical, usually spherical particles. For example, these spheres could represent small regions of a fluid (e.g. one particle representing a cluster of water molecules).



The physical size of the simulation is defined by a simulation box and the particles are randomly distributed inside it. This is the initial state of the system.

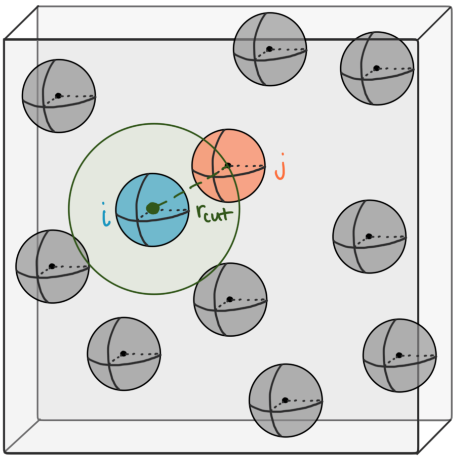
ADDING PARTICLE INTERACTIONS

The motion of a particle in a DPD simulation is determined by it's interactions with the particles around it.



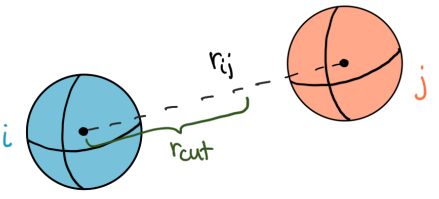
Interactions are calculated as the pairwise interaction forces between any two interacting particles i and j.

To identify which particles form an interacting pair, the simulation constructs a neighboring list defined by a cut-off radius (r_{cut}).



Using the position of the center of each particle, the simulation calculates the interparticle separation distance (r_{ij}) for every possible particle pair. If $r_{ij} \leq r_{cut}$ then the particles are defined as interacting with each other and the pairwise interaction forces are calculated.

If the center-center distance (r_{ij}) between two particles is greater than the cut-off distance, then the particles do not interact.



CALCULATING MOTION

The equation of motion ($F=ma$) for a DPD particle i is the sum of the different pairwise interaction forces between it and any particle j that it is interacting with. This makes DPD extremely versatile because you can easily add additional forces to your simulation (e.g. different types of interparticle attraction or repulsion, the force from an external electric or magnetic field, gravity, activity for active matter, etc.)

Traditional DPD uses the sum of three forces:

- conservative force: a soft repulsive force, which keeps particles from overlapping or sticking together into higher order structures
- dissipative force: the viscous resistance of the fluid (acting against particle motion and trying to reduce the kinetic energy of the system)
- random force: a stochastic force representing random thermal fluctuations

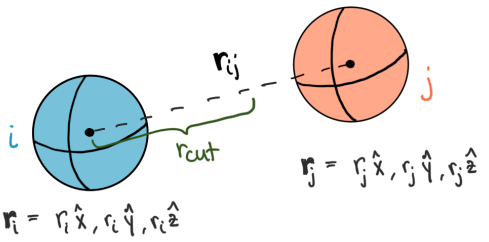
$$\mathbf{F}_i = m_i \frac{d\mathbf{v}_i}{dt} = \sum \mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R$$

where velocity is calculated from the particle's change in position.

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i$$

At each timestep in a simulation the neighboring list is built, the interacting pairs are identified, the forces are calculated, the changes in position and velocity are calculated for each particle, and then the position and velocity of each particle is updated.

CALCULATING FORCES



The force calculations for each ij-pair use:

the interparticle distance vector: $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$

as well as the magnitude and position components

$$r_{ij} = |\mathbf{r}_{ij}| \quad \text{and} \quad \mathbf{e}_{ij} = \hat{\mathbf{r}}_{ij} = \frac{\mathbf{r}_{ij}}{r_{ij}}$$

and a weight function that goes to zero after the cut-off distance (r_{cut})

$$w_{ij}(r_{ij}) = \left(1 - \frac{r_{ij}}{r_{cut}}\right)$$

Conservative Force

a linearly decreasing repulsive soft interaction potential, allowing for larger simulation timesteps (as compared to the hard potential in a Lennard Jones model)

$$\mathbf{F}_{ij}^{C} = a_{ij}\, w_{ij}(r_{ij})\, \mathbf{e}_{ij}$$

where a_{ij} is the repulsion parameter (signifying the max repulsion) defined as:

$$a_{ij} \approx k_B T \, \frac{\kappa^{-1} - 1}{0.2\,\rho}$$

It comes from the chemical identity of the particles being simulated where $\kappa \equiv$ compressibility component and $\rho \equiv$ density. For water, $a_{ij} = 25$.

Dissipative Force

the viscous resistance of the fluid, which acts against the relative motion of the particles and tries to reduce the kinetic energy of the system.

$$\mathbf{F}_{ij}^{D} = -\gamma_{ij} \left[w_{ij}(r_{ij})\right]^2 (\mathbf{v}_{ij} \cdot \mathbf{e}_{ij})\mathbf{e}_{ij}$$

where γ_{ij} is the controlling parameter for the max viscous resistance (related to the background viscosity of the fluid, η_0). Values for γ_{ij} and η_0 vary with the type of simulation. A typical value for a system at equilibrium is $\gamma_{ij} = 4.5$ (and $\eta_0 = 0.3$); while the typical value for a system under flow is $\gamma_{ij} \sim 45\text{-}50$ (and $\eta_0 = 1.1$). Equilibrium simulations will still run at higher γ_{ij} values, but they will take longer to equilibriate (or require larger timesteps)

note also that $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$; and, $\mathbf{v}_{ij} \cdot \mathbf{e}_{ij}$ is the dot product

Random Force

a stochastic (Brownian) force representing the random temperature fluctuations of the system

$$\mathbf{F}_{ij}^{R} = \sigma_{ij}\, w_{ij}(r_{ij})\, \frac{\Theta_{ij}}{\sqrt{\delta t}}\, \mathbf{e}_{ij}$$

Because the dissapative and random forces combine to create the canonical ensemble and control system temperature, it is possible to write their respective controlling parameters (σ_{ij} and γ_{ij}) in terms of one another.

$$\sigma_{ij} = \sqrt{2\,\gamma_{ij}\,k_B T}$$

Therefore, we only need to define one of these parameters in our simulations (γ_{ij}).

The theta parameter (Θ_{ij}) is a white-noise parameter (i.e. random number)

Note that the random force MUST be dependent on the size of the timestep (δt) in order to simulate physical behavior. The mean diffusion over a physical time interval must be finite (and independent of the choice of size of the timestep); however, the spread of the random force increases as that physical time interval is divided into more timesteps. Dependence on timestep accounts for this effect.

MOVING THROUGH TIME: INTEGRATION

At the end of one timestep there are various methods that can be used to update the particle information and move forward in time.

We typically use an NVT method, where the number of particles (N), the volume of the entire system (V), and the system temperature all remain constant. This allows us to observe changes in energy - i.e. the sum of kinetic and potential energy of the system - and pressure (-stress). However, you will notice in the simulation scripts that we access the integration calculations through an NVE module (not NVT). This is only for integration and NOT calculating energy or temperature.

To integrate Newton's equations of motion and calculate the particle trajectories (position and velocity) in the new timestep, we use a velocity-Verlet algorithm. This is a two-step integration method.

A one-step integration method would update position and velocity by one full timestep (δt): $\mathbf{r}_i(t) \rightarrow \mathbf{r}_i(t+\delta t)$ and $\mathbf{v}_i(t) \rightarrow \mathbf{v}_i(t+\delta t)$

The velocity-Verlet algorithm still updates position by one full timestep, but we divide the velocity integration into partial timesteps $\lambda\delta t$ (typically one half timestep, $\lambda = \frac{1}{2}$): $\mathbf{v}_i(t) \rightarrow \mathbf{v}_i(t+\lambda\delta t) \rightarrow \mathbf{v}_i(t+\delta t)$

Because force (and acceleration) are dependent on velocity, breaking the velocity integration step into two parts also adds an intermediate update to the force calculation: $\mathbf{F}_i(t, \mathbf{r}_i(t), \mathbf{v}_i(t)) \rightarrow \mathbf{F}_i(t+\delta t, \mathbf{r}_i(t+\delta t), \mathbf{v}_i(t+\lambda\delta t))$

Through $F=ma$ $\left(a_i(t+\lambda\delta t) = \frac{F_i(t+\delta t, v_i(t+\lambda\delta t))}{m_i}\right)$, this intermediate force calculation is used to finish updating the particle velocities to Δt:

$$v_i\left(t+\lambda\delta t, a_i(t+\lambda\delta t)\right) \rightarrow v_i(t+\delta t)$$

We use the velocity-Verlet algorithm because it helps to smooth the transition between ptimestes and produce more physically accurate behavior despite our simulations' otherwise non-physical use of discretized time.

Velocity-Verlet Algorithm

STEP 1:

use the current force, position, and velocity values for each particle to move the position one full timestep forward in time

$$r_i(t+\Delta t) = r_i(t) + \delta t\, v_i(t) + \tfrac{1}{2}\delta t^2\left(\frac{F_i}{m_i}\right)$$

then update the velocity by a partial timestep $\lambda\delta t$ (typically $\frac{\delta t}{2}$)

$$v_i(t+\lambda\delta t) = v_i(t) + \lambda\delta t\left(\frac{F_i}{m_i}\right)$$

and use this information to update the DPD forces by a partial timestep

$$F_i(t, r_i, v_i) = \sum F_i^{DPD}(t, r_i, v_i)$$

$$\rightarrow F_i(t+\delta t, r_i(t+\delta t), v_i(t+\lambda\delta t))$$

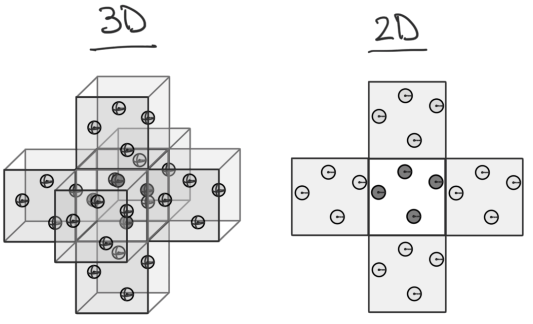
STEP 2:

finish updating velocity using the partially updated forces $(F=ma)$

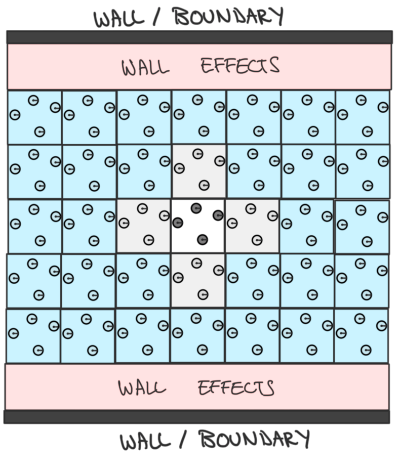
$$v_i(t+\delta t) = v_i(t+\lambda\delta t) + \lambda\delta t\left(\frac{F_i(t+\delta t)}{m_i}\right)$$

BOUNDARY CONDITIONS

A basic DPD simulation does not include any wall effects. Instead we use periodic boundary conditions that allow the simulated system to interact with duplicate images of itself in all directions.

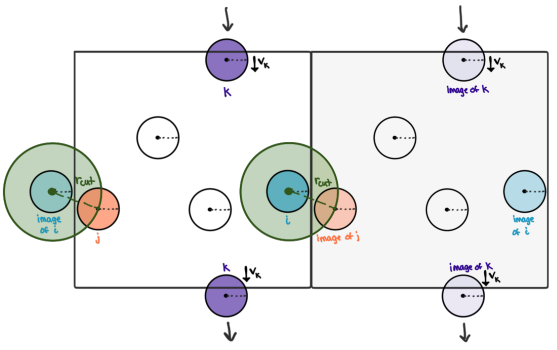


If we are simulating a real-world system (like water) that has a more-or-less uniform structure throughout, then we can use periodic boundary conditions to assume our results represent any location in the system that is far enough away from a real world boundary to avoid wall effects.



This means that a particle i located near a boundary of the simulation box will interact with a particle j along the opposite boundary.

Similarly, a particle K that crosses a boundary will reappear at the opposite side of the simulation box (the particle motion is wrapped around the box to the other side).



DPD UNITS

In a DPD simulation space and time are measured in DPD units, and you must calculate the correct conversion to relate your system back to real world units. Measurements in DPD units in and of themselves are meaningless in the real world, but the relationships between them can be generalized to describe a variety of systems.

Especially since we are interested in discovering generalized descriptions of colloidal physics that apply to many different experimental systems, we often do not convert back to specific real world units, but non-dimensionalize things as much as possible – e.g. scale units against a generalized experimental parameter, such as scaling distance in terms of particle radius, or scaling time in terms of particle diffusion time.

Other parameters typically use standardized references values accepted by the DPD research community. Because DPD has been around for over 30 years, you can find most of the important, generally accepted values for specific types of systems (and the rationale for how they were chosen) already published in scientific papers.

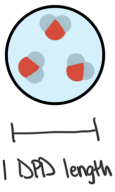
For a basic DPD simulation of water we usually base our system around a cluster of three water molecules. These conversions were published in the paper: Boromand, Jamali, and Maia (2015) "Viscosity measurement techniques in Dissipative Particle Dynamics" in Computer Physics Communications.

For this simulation we set:

- simulation box size (L_x, L_y, L_z)
- particle radius (r_p), volume (V_p), and mass (m_p)
- the cut-off radius (r_{cut})
- the number density (ρ)
- the system temperature (k_BT)
- repulsion parameter (a_{ij})
- viscous dissipation parameter (γ_{ij}) and background viscosity (η_0)
- simulation timestep (δt) AKA integration timestep
- recording timestep (AKA trigger or period)
- length of the simulation (the total number of timesteps)

The simulation box is typically given a minimum side length $L_{min} \geq 30$ DPD units

A single particle (one DPD bead) is defined as a soft sphere the size of 3 H_2O molecules with a diameter of 1 DPD unit of distance (radius $r_s = 0.5$).



$r_s = 0.5$

$V_s = \frac{4}{3}\pi r_s^3$

Each particle is given a mass of 1 DPD mass unit ($m_s = 1.00$), where:

$$m = 1.00 \equiv \frac{\left(\begin{smallmatrix}\text{number of}\\ \text{molecules}\\ =3\end{smallmatrix}\right)\left(\begin{smallmatrix}\text{molar weight}\\ \text{of water}\\ =18\,g/mol\end{smallmatrix}\right)}{\text{Avogadro's number}} = 8.98 \times 10^{-23}\,g$$

We set the cut-off radius for calculating the neighboring list and interparticle forces to 1 DPD unit as well, where:

$$r_{cut} = 1 \equiv \left[\left(\begin{smallmatrix}\text{density}\\ \rho\end{smallmatrix}\right)\left(\begin{smallmatrix}\text{number of molecules}\\ N\end{smallmatrix}\right)\left(\begin{smallmatrix}\text{particle mass}\\ m\end{smallmatrix}\right)\left(\begin{smallmatrix}\text{volume of a}\\ \text{water molecule}\\ V=30\text{Å}\end{smallmatrix}\right)\right]^{1/3} = 6.45\,\text{Å} = 0.645\,nm$$

and the number density of the system is set to $\rho = 3$ DPD units, where:

$$\rho = 3.00 \equiv \frac{(\text{density } \rho)\left(\begin{smallmatrix}\text{number of molecules}\\ =3\end{smallmatrix}\right)\left(\begin{smallmatrix}\text{molar weight}\\ \text{of water} = 18g/mol\end{smallmatrix}\right)}{(\text{Avogadro's number})(\text{cut-off radius})^3} \sim 996.3\ \frac{Kg}{m^3}$$

The system temperature is set as a thermal energy measurement in terms of the Boltzman constant (k_B); typically $k_BT = 1$ for water simulations.

The repulsion parameter for water molecules is $a_{ij} = 25$

The viscous dissipation parameter (γ_{ij}) can vary to speed up simulation run time or more accurately simulate different behaviors. For water, we usually use $\gamma_{ij} = 4.5$

The "background viscosity" of this simulation is the viscosity of water and varies with the viscous dissipation parameter and the system temperature. For $\gamma_{ij} = 4.5$ and $k_BT = 1.0$, the viscosity is calculated as $\eta_0 = 0.85 \pm 0.1$ DPD units and usually set as $\eta_0 = 1.1$ DPD units, where

$$\eta_0 = 0.85 \pm 0.1 \equiv \frac{(\text{viscosity of water } \eta)(\text{time } \tau)(\text{Boltzmann constant } k_B)(\text{Temperature } T = 298K)}{(\text{cut-off radius})^3} = 0.001 \text{ Pa·s}$$

The simulation timestep (δt) can also vary to optimize simulation performance. Its conversion value will vary with the size of the cut-off radius (r_{cut}). For this system a timestep of $\delta t = 0.01$ DPD times is approximately 1 picosecond.

$$\delta t = 0.01 \equiv \frac{(\text{timestep } \delta t)(\text{mass})^{0.5}(\text{cut-off radius})}{[(\text{Boltzmann constant } k_B)(\text{Temperature } T = 298K)]^{0.5}} \sim 1 \text{ ps}$$

The recording timestep (or trigger or period) is the interval at which data is recorded to the output file. It is measured in number of timesteps. Increasing the trigger will decrease the number of timesteps that are recorded as frames in the output file and also decrease simulation run time, but you could miss valuable data. For water try running an equilibrium simulation with trigger = 1 (to record every timestep) and a shearing simulation with trigger = 100 (to record every one-hundredth timestep).

The length of a simulation will also depend on the type of behavior being observed. For water try running an equilibrium simulation for $N_{timesteps} = 1000$ and a shearing simulation for $N_{timesteps} = 100,000$