Supporting Information for:

ARPS: Adaptively Restrained Particle Simulations

Svetlana Artemova and Stephane Redon

Contents

1 Statistical averages in the NVT ensemble, mathematical proofs 2

2 Temperature in the NVT ensemble 3

3 Threshold parameters 4

4 Partitioned Euler integration method 7

5 Langevin dynamics 7

6 Harmonic oscillator in 1D, phase portraits for different thresholds 8

7 Pseudo-code for the integration algorithm in NVE ensemble 9

8 Pseudo-code for the force update 10

9 Alternative method for the force update 11

10 Collision cascade 12

11 Radial distribution function 13

12 Polymer in solvent, more statistics 14

1 Statistical averages in the NVT ensemble, mathematical proofs

ARPS can recover any statistical property $\langle \mathbf{A} \rangle$ by considering the AR Hamiltonian is a biased version of the original one:

$$
\begin{aligned}
H_{AR} &= \frac{1}{2}\mathbf{p}^T\Phi(\mathbf{q},\mathbf{p})\mathbf{p} + V(\mathbf{q}) \\
H_{AR} &= \frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} + V(\mathbf{q}) + \tfrac{1}{2}\mathbf{p}^T(\Phi(\mathbf{q},\mathbf{p}) - \mathbf{M}^{-1})\mathbf{p} \\
H_{AR} &= H + \frac{1}{2}\mathbf{p}^T(\Phi(\mathbf{q},\mathbf{p}) - \mathbf{M}^{-1})\mathbf{M}^{-1}\mathbf{p} \\
H_{AR} &= H + V_{AR}(\mathbf{q},\mathbf{p})
\end{aligned}
$$

Thus, to compute an average of $\mathbf{A}$:

$$
\langle \mathbf{A} \rangle_H = \frac{\int \mathbf{A}(\mathbf{q},\mathbf{p}) e^{-\frac{H(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}}{\int e^{-\frac{H(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}},
$$

one may split the original Hamiltonian to get:

$$
\begin{aligned}
\langle \mathbf{A} \rangle_H &= \frac{\mathbf{A}(\mathbf{q},\mathbf{p}) e^{-\frac{H_{AR}(\mathbf{q},\mathbf{p}) - V_{AR}(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}}{\int e^{-\frac{H_{AR}(\mathbf{q},\mathbf{p}) - V_{AR}(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}} \\
\langle \mathbf{A} \rangle_H &= \frac{\int \mathbf{A}(\mathbf{q},\mathbf{p}) e^{\frac{V_{AR}(\mathbf{q},\mathbf{p})}{k_B T}} e^{-\frac{H_{AR}(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}}{\int e^{\frac{V_{AR}(\mathbf{q},\mathbf{p})}{k_B T}} e^{-\frac{H_{AR}(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}} \\
\langle \mathbf{A} \rangle_H &= \frac{\langle \mathbf{A} e^{\frac{V_A}{k_B T}} \rangle_{H_{AR}}}{\langle e^{\frac{V_A}{k_B T}} \rangle_{H_{AR}}}
\end{aligned}
$$

When $\mathbf{A}$ only depends on positions, and the AR Hamiltonian is separable, the correct statistics are obtained straight away, because:

$$
\begin{aligned}
&\langle \mathbf{A}(\mathbf{q}) \rangle_{H_{AR}} = \frac{\int \mathbf{A}(\mathbf{q}) e^{\frac{-H(\mathbf{q},\mathbf{p}) - V_{AR}(\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}}{\int e^{\frac{-H(\mathbf{q},\mathbf{p}) - V_{AR}(\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}} = \\
&= \frac{\int \mathbf{A}(\mathbf{q}) e^{-\frac{V(\mathbf{q})}{k_B T}} d\mathbf{q} \int e^{\frac{-T(\mathbf{p}) - V_{AR}(\mathbf{p})}{k_B T}} d\mathbf{p}}{\int e^{-\frac{V(\mathbf{q})}{k_B T}} d\mathbf{q} \int e^{\frac{-T(\mathbf{p}) - V_{AR}(\mathbf{p})}{k_B T}} d\mathbf{p}} = \frac{\int \mathbf{A}(\mathbf{q}) e^{-\frac{V(\mathbf{q})}{k_B T}} d\mathbf{q}}{\int e^{-\frac{V(\mathbf{q})}{k_B T}} d\mathbf{q}} = \\
&= \frac{\int \mathbf{A}(\mathbf{q}) e^{-\frac{V(\mathbf{q})}{k_B T}} d\mathbf{q} \int e^{\frac{-T(\mathbf{p}))}{k_B T}} d\mathbf{p}}{\int e^{-\frac{V(\mathbf{q})}{k_B T}} d\mathbf{q} \int e^{-\frac{T(\mathbf{p})}{k_B T}} d\mathbf{p}} = \frac{\int \mathbf{A}(\mathbf{q}) e^{\frac{-H(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}}{\int e^{\frac{-H(\mathbf{q},\mathbf{p})}{k_B T}} d\mathbf{q}d\mathbf{p}} = \langle A(\mathbf{q}) \rangle_H
\end{aligned}
$$

2 Temperature in the NVT ensemble

If the system is described by a Hamiltonian function, the system's temperature is defined as:

$$T = \frac{1}{Dk_B} \left\langle \sum_{i=1}^{N} \left(p_i \cdot \frac{\partial H_{AR}}{\partial p_i} \right) \right\rangle,$$

where D is the dimensionality of the system (*e.g.* $3N$ in 3D for N particles) and the dot sign stands for the dot product. It is known that for separable Hamiltonians:

$$\left\langle \sum_{i=1}^{N} \left(p_i \cdot \frac{\partial H}{\partial p_i} \right) \right\rangle = \left\langle \sum_{i=1}^{N} \left(q_i \cdot \frac{\partial H}{\partial q_i} \right) \right\rangle = \left\langle \sum_{i=1}^{N} \left(q_i \cdot \frac{\partial V(q)}{\partial q_i} \right) \right\rangle.$$

Thus, the temperature T_{AR} in an AR simulation (with the separable Hamiltonian, which is the case for the inverse inertia matrix described in the main document) may be represented as a quantity that only depends on particles positions. Therefore, this quantity is conserved by AR simulations in the NVT ensemble and, as in the full-dynamics simulation, is equal to the thermostat temperature:

$$T_{AR} = T_t.$$

3 Threshold parameters

Both the stability and efficiency of an AR simulation are influenced by the choice of the threshold parameters.

Although it is too early to provide a method for choosing optimal thresholds (which probably depend on the adaptive simulation type – *i.e.* the choice of the adaptive inverse inertia matrix – as well as on the simulated system), we can still mention some indications.

In the NVE ensemble, velocities $\dot{q}$ of the particles evolve according to equation (3) in the paper, and are influenced by both AR thresholds ε_r and ε_f. Even when both thresholds are large, if the difference between ε_f and ε_r is also large, then a particle will transition very smoothly between restrained dynamics and full dynamics, so that its velocity will not become unrealistic. We illustrate this on a simple example: a harmonic oscillator in 1D of mass 1 and spring stiffness 1. In Fig. 1 we plot trajectories and norms of velocities of the moving particle for the full-dynamics simulation and two AR simulations with different thresholds. On the first AR-simulation trajectory, velocities might indeed become unrealistic since the difference δ between the AR thresholds is small. On the second AR-simulation trajectory, though, the velocities are small because δ is large (while ε_r is unchanged). In this case, actually, the particle never becomes fully released (although the particle still oscillates), and ρ always remain above 0.95 (Fig. 1, bottom).

For the same example, we plot in Fig. 2 the particle's velocity $\dot{q}$ as a function of its momentum p (according to equation (3) in the paper) for the same three simulations. As one can see, for small values of momentum, the velocity $\dot{q}$ is zero as the particle is fully restrained. For large values of p, the velocity is equal to the momentum (divided by the mass, equal to 1 in this example), since the particle is fully released. When δ increases, the velocity $\dot{q}$ oscillates less drastically. Note that the plot in Fig. 2 depends on the specific ρ function used in the paper, and that it might also be possible to choose ρ functions such that the particle velocity always remains below the corresponding full-dynamics velocity (thin black line).

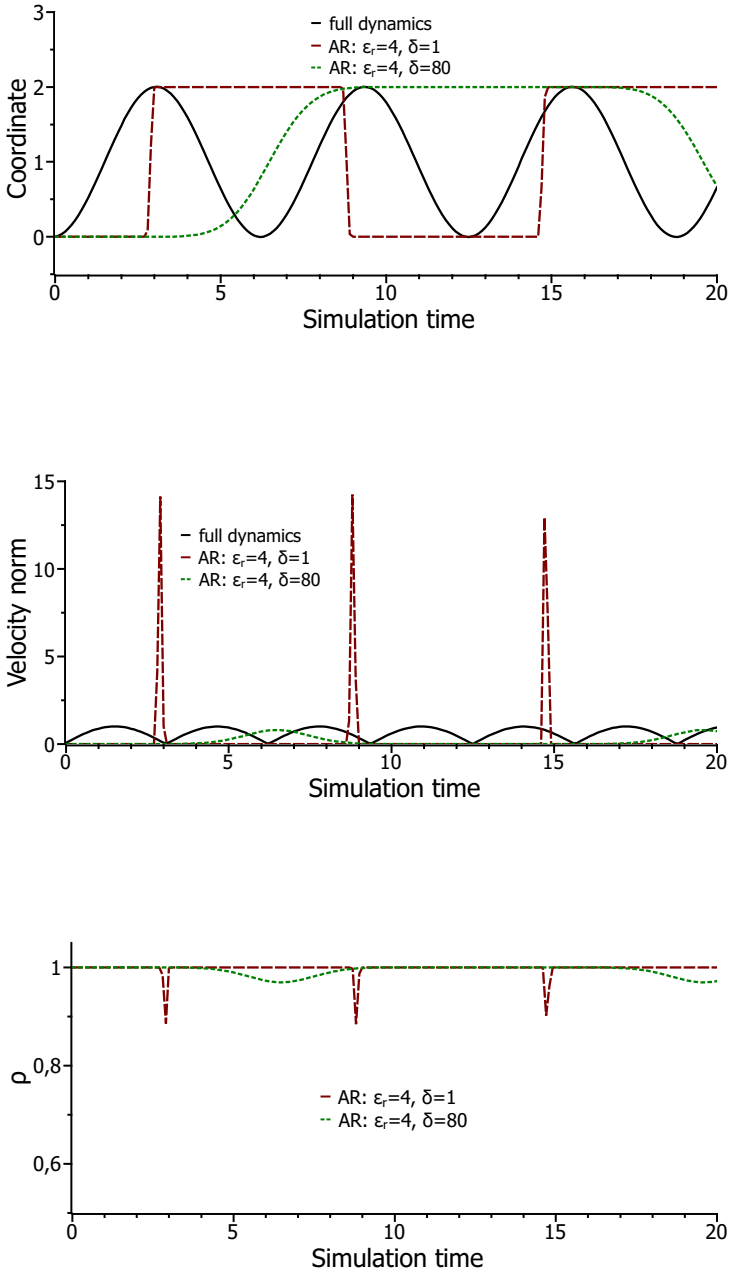

Figure 1: Full dynamics simulation and two AR simulations of a harmonic oscillator in 1D: trajectory of the particle in each simulation (top), norm of the velocity of the particle in each simulation as a function of time (middle), value of the restraining function ρ (bottom).[5]

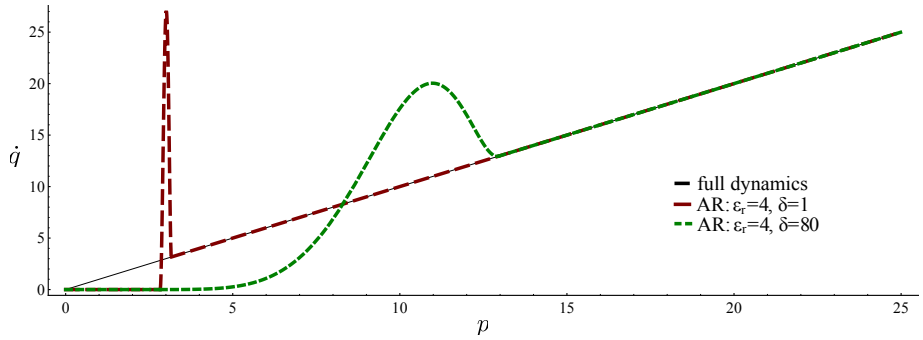

Figure 2: Harmonic oscillator in 1D. Velocity $\dot{q}$ of the particle is plotted as a function of its momenta p for the full-dynamics simulation and two AR simulations.

In NVT simulations, freezing positions too often might be counter-productive in some cases. For example, if the solvent surrounding the polymer is too often static, this might overly limit the speed of conformational sampling for the polymer: convergence of statistical properties might become slower than with full-dynamics simulations.

We hope that we will be able to provide some analytical tools to help users choose AR parameters, but we believe that these will also be determined through experimentation and learning (as other typical simulation parameters, including *e.g.* the equilibration period, the set of temperatures in replica exchange methods, etc.).

4 Partitioned Euler integration method

The *partitioned* Euler method is the integration method which, for partitioned systems:

$$\begin{aligned} \dot{v} &= b(u, v), \\ \dot{u} &= a(u, v), \end{aligned}$$

results in the following set of equations for a numerical integrator:

$$\begin{aligned} v_{n+1} &= v_n + b(u_n, v_{n+1})h, \\ u_{n+1} &= u_n + a(u_n, v_{n+1})h, \end{aligned}$$

where h is a time step. This method is symplectic.

5 Langevin dynamics

To simulate the NVT ensemble we perform an *adaptive Langevin dynamics simulation, i.e.* a Langevin dynamics simulation of the adaptive Hamiltonian.

To do so, we consider the general form of Langevin equations :

$$\begin{aligned} dq_t &= \nabla_p H_{AR}(q_t, p_t)dt, \\ dp_t &= -\nabla_q H_{AR}(q_t, p_t)dt - \gamma \nabla_p H_{AR}(q_t, p_t)dt + \sigma dW_t, \end{aligned} \tag{1}$$

where $t \to dW_t$ is a $3N$-dimensional standard Brownian motion, and σ and γ are $3N \times 3N$ real matrices. The following fluctuation-dissipation relation should also be satisfied: $\sigma\sigma^T = 2\gamma/\beta$ for $\beta = 1/k_B T$.

For an NVT simulation, as an integration algorithm, we use the following splitting scheme: a half-step for a Langevin part of the equations, a full time step for the Hamiltonian part, and again a half-step for a Langevin part. In this case the second equation in (1) always has p_{n+1} in its right part. To solve this implicit equation, we may again use a fixed-point algorithm.

$$\begin{aligned} p_{n+1/2} &= p_n - \left(\frac{\partial H_{AR}(q_n, p_{n+1/2})}{\partial q_n} + \gamma \frac{\partial H_{AR}(q_n, p_{n+1/2})}{\partial p_{n+1/2}} \right) \frac{h}{2} + \sigma G_n \sqrt{\frac{h}{2}}, \\ q_{n+1} &= q_n + \left(\frac{\partial H_{AR}(q_n, p_{n+1/2})}{\partial p_{n+1/2}} \right) h, \\ p_{n+1} &= p_{n+1/2} - \left(\frac{\partial H_{AR}(q_{n+1}, p_{n+1})}{\partial q_{n+1}} + \gamma \frac{\partial H_{AR}(q_{n+1}, p_{n+1})}{\partial p_{n+1}} \right) \frac{h}{2} + \sigma G_{n+1/2} \sqrt{\frac{h}{2}}, \end{aligned}$$

where $\{G_k\}$ is a sequence of i.i.d. Gaussian random vectors with zero mean and identity matrix as covariance.

6 Harmonic oscillator in 1D, phase portraits for different thresholds

Fig. 3 shows phase portraits of the adaptively restrained harmonic oscillator for varying thresholds. For the convenience of the analysis we introduce a variable $\delta = \varepsilon_f - \varepsilon_r$, a width of the transition region.

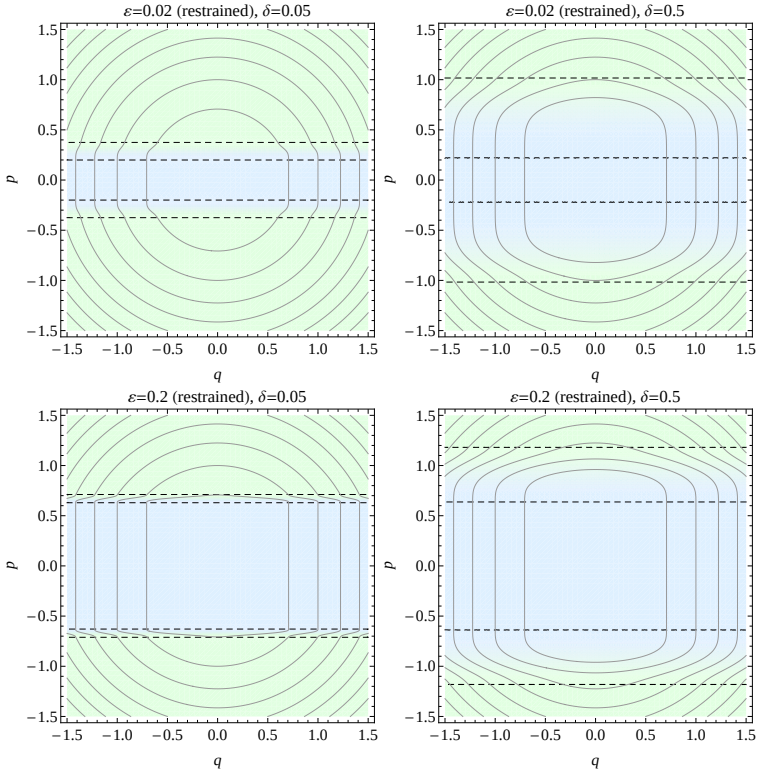

Figure 3: Phase portraits for the AR Hamiltonian for several specific sets of parameters, ε stands for ε^r. Threshold parameters expressed in kcal/mol.

7 Pseudo-code for the integration algorithm in NVE ensemble

The integration scheme in pseudo-code is described in Algorithm 1. The function *updateState()* is called every time step. We also provide the pseudo-code functions called by the *updateState()* function, except for the *updateForces()* one, that can be found below.

Algorithm 1 Integration scheme. Lower indices stand for the time steps, upper indices are particles serial numbers, a denotes a particle, q – particle position, p – its momentum, f – its force. The time step size is denoted by h.

updateState()
    for (each time step)
        updateForces()
        updateMomenta()
        updatePositions()

updateMomenta()
    for (each particle a^i)
        $p^i_{n+1} = p^i_n + f^i_n h$

updatePositions()
    for (each particle a^i)
        compute ρ^i
        $q^i_{n+1} = q^i_n + \left(p^i_n/m^i(1-\rho^i) - 0.5\|p^i_n\|^2/m^i \times \partial\rho^i/\partial p^i_{n+1}\right) h$

8 Pseudo-code for the force update

In Algorithm 2 we summarize in pseudo-code the algorithm for the force update.

Algorithm 2 Force update algorithm. Lower indices stand for time step, upper indices are particles serial numbers, a denotes a particle, q – particle position, f – its force, r – radius vector.

updateForces()

if (step =1)

create grid, distribute all particles to the grid cells
for (all particles)
for (every particle a^k from the current cell and all surrounding cells)
if ($r^{ik} \leqslant$ cutoff **and** $i < k$)
//add interaction
$f^* =$ calculateForce(r^{ik})
$f^i = f^i + f^*$
$f^k = f^k - f^*$

else

for (each active particle a^i)
for (every particle a^k from the current cell and all surrounding cells)
if ($r^{ik} \leqslant$ cutoff **and** ($\rho^k = 1$ **or** $i < k$))
//subtract interaction
$f^* =$ calculateForce(r^{ik})
$f^i = f^i - f^*$
$f^k = f^k + f^*$

for (each active particle a^i)
move the particle (if necessary) to another grid cell

for (each active particle a^i)
for (every particle a^k from the current cell and all surrounding cells)
if ($r^{ik} \leqslant$ cutoff **and** ($\rho^k = 1$ **or** $i < k$))
//add interaction
$f^* =$ calculateForce(r^{ik})
$f^i = f^i + f^*$
$f^k = f^k - f^*$

9 Alternative method for the force update

In this section, we propose an alternative method for the force update.

This method consists in examining the two neighbor lists (*i.e.* the lists of pairs of interacting particles) constructed for the system at the current and previous time steps, to determine which relative positions have changed and, therefore, which forces need to be updated. This method is optimal in the number of updated forces, but might be too expensive when the cost of updating forces is low, since examining the lists has a linear complexity in their size.

In Algorithm 3 we summarize in pseudo-code this method.

Algorithm 3 An alternative method for the force update. Lower indices stand for time steps, upper indices are particles serial numbers, a denotes a particle, q – particle position, f – its force, r – radius vector.

---

updateForces_A()

construct a neighbor list

compare Current and Previous neighbor lists, update forces:

for (each element e_k=(a^i, a^j, r^{ij}_{n-1}, f^{ij}_{n-1}) of the Previous neighbor list)

if ($r^{ij}_{n-1} = q^j_n - q^i_n$)

put e_k to the Transition list

else // subtract interaction

$f^i = f^i - f^{ij}_{n-1}$

$f^j = f^i + f^{ij}_{n-1}$

for (each element e_k=(a^i, a^j, r^{ij}_n, f^{ij}_n) of the Current neighbor list)

if (r^{ij}_n != $q^j_{n-1} - q^i_{n-1}$)

put e_k to the Transition list

// add interaction

$f^* = \text{calculateForce}(r^{ij}_n)$

$f^i = f^i + f^*$

$f^j = f^j - f^*$

Previous neighbor list = Transition list

clear Current neighbor list and Transition list

---

10 Collision cascade

Error analysis

We discuss the error caused by ARPS due to the system simplification. Already in the main document we output the maximal absolute displacement of the particles $\Delta q_{\max}$ and the root-mean-square deviation:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n} \|q_i - q_i^f\|^2}{n}},$$

where q_i is a vector coordinate of particle i at the last time step of the AR simulation (and on the corresponding picture), q_i^f is the same coordinate of this particle in the reference simulation.

However, it might be interesting to analyze the distribution of the error in space. The bottom row of Fig. 4 shows the normalized displacement error A_i for each particle i:

$$A_i = \begin{cases} \dfrac{\|q_i - q_i^f\|}{\sigma}, & \text{if } \|q_i - q_i^f\| < \sigma; \\ \sigma, & \text{else.} \end{cases}.$$

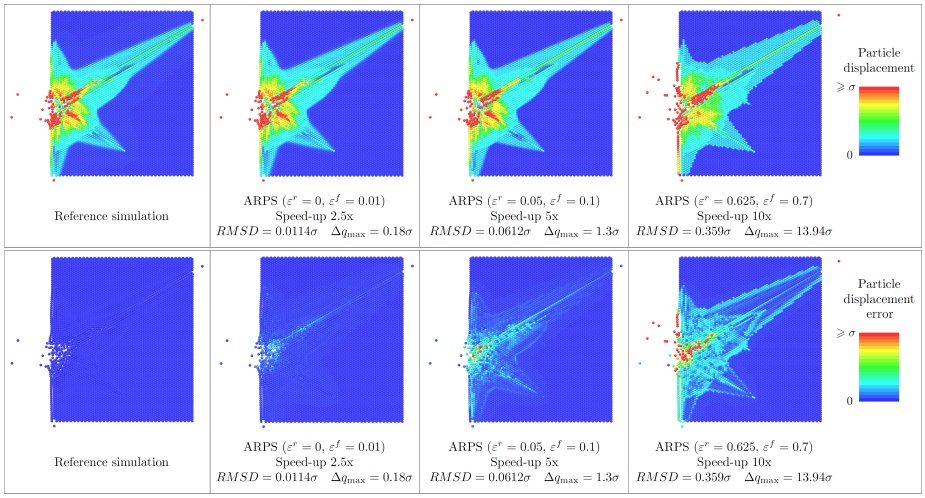

Figure 4: Collision cascade. Top row: the final configurations for varying degrees of precision. Bottom row: particle displacement error.

Video

We provide the video of the collision cascade for these four simulations as a separate supplementary material file as well.

11 Radial distribution function

In this numerical experiment, we show that ARPS preserve the Radial Distribution Function (RDF) of an Argon liquid box. For both simulations, we started collecting statistics when the number of updated forces in the AR simulation became non-zero (t=4.88 ps). The AR simulation modifies the system's dynamics significantly: on average, only 3% of particles are active at each time step. As a result, convergence to the correct radial distribution function $g(r)$ is slower in the AR simulation than in the full-dynamics simulation *for a given number of time steps*. To illustrate this, we plot in Fig. 5 the reference and AR radial distribution functions obtained after 50 000 steps (in the main document, the total number of steps is 150 000, since we wanted to obtain converged results). As can be seen, the RDF obtained using the AR simulation is more noisy than the reference one for this number of time steps – precisely, because particle positions haven't been sufficiently sampled, the plotted AR RDF keeps traces of the original cubical lattice used as initial configuration.

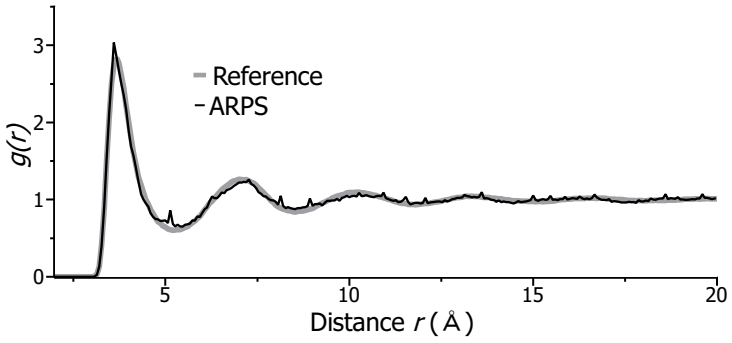

Figure 5: Radial distribution functions for an Argon system (periodic 3D box, 343 particles, NVT ensemble). After 50 000 simulation steps, the RDF obtained with the AR simulation is more noisy than the reference one.

Thanks to the reduced number of moving particles, though, the computational cost of an AR time step is significantly reduced.

12 Polymer in solvent, more statistics

Simulating a solvated polymer in the NVT ensemble, we observed not only the hydrodynamic radius R_H:

$$R_H = \langle R_H^{-1} \rangle^{-1}, \quad \langle \frac{1}{R_H} \rangle = \frac{1}{N^2} \sum_{i \neq j} \langle \frac{1}{r_{ij}} \rangle,$$

where N is the number of monomers in a polymer and r_{ij} is the distance between the i-th and j-th monomers, but also the average maximal distance of a monomer from the polymer's center of mass $\langle \Delta r_{\max} \rangle$, radius of gyration R_G:

$$R_G = \sqrt{\langle R_G^2 \rangle}, \quad \langle R_G^2 \rangle = \frac{1}{N} \sum_i \langle (r_i - R)^2 \rangle,$$

where r_i is the position of the i-th monomer and $R = N^{-1} \sum_i r_i$ is the polymer's center of mass, and end-to-end distance R_E:

$$R_E = \sqrt{\langle R_E^2 \rangle}, \quad \langle R_E^2 \rangle = \langle (r_N - r_1)^2 \rangle.$$

In Table 1 we output the mean values of the variables of interest and also the error: the difference between the full-dynamics and AR values divided by the reference value. As hydrodynamics radius R_H was averaged over more values, it is the most precise. The end-to-end distance R_E was averaged on less information and is the less precise among the presented values.

	Reference	ARPS	Error	Speed-up
$\langle \Delta r_{\max} \rangle$	3.77258	3, 77688	0.1%	4x
R_G	2.77471	2.7776	0.1%	4x
R_E	6.82529	6.83596	0.2%	4x
R_H	4.45913	4.46185	0.06%	4x

Table 1: Average values over reference and AR simulations.