Nuclear Science and Techniques (2026) 37:103
https://doi.org/10.1007/s41365-026-01952-z

Alternative pairwise nuclear fusion optimization algorithm in arbitrarily weighted particle-in-cell simulations

Qian Dong[1,2] · Chao-Zhi Li[1] · Chen Xie[1] · Zhi-Dong Chen[1,2] · De-Bin Zou[1] · Wen Luo[2] · Tong-Pu Yu[1]

Received: 19 February 2025 / Revised: 26 June 2025 / Accepted: 7 July 2025 / Published online: 25 March 2026

Abstract
An optimization algorithm for pairwise nuclear fusion with arbitrarily weighted macroparticles is proposed and demonstrated using particle-in-cell (PIC) simulations. A faster computation of the Coulomb collision operator is achieved compared with that obtained through the classical nuclear fusion algorithm. Thus, the proposed algorithm can be implemented in the PIC simulation code with marginal effort. The algorithm is benchmarked in scenarios with like-particles, D(d,n)^{3}He, unlike-particles, D(t,n)^{4}He, thermonuclear fusion, and beam-target fusion. Moreover, the CPU time could be reduced significantly without reducing the simulation accuracy.

Keywords Particle-in-cell simulation · Nuclear fusion · Coulomb collision · Binary collision

1 Introduction

Particle-in-cell (PIC) simulations provide insights into the kinetic properties of particles (such as velocity and position) during plasma interactions by tracking charged particles under the influence of both self-consistent and external electromagnetic fields. This technique provides an intuitive understanding of the kinetic behavior of field–particle interactions and is a vital tool for evaluating various plasma physics problems [1–3], such as inertial confinement fusion (ICF) [4, 5], ion acceleration [6, 7], laser–nuclear interactions [8], and positron generation [9–12]. Although PIC simulations of plasma physical processes are widely used, two critical challenges need to be addressed in practical applications: 1) numerical noise, which can be suppressed significantly with various methods [13] (e.g., variable weights [14], adaptive mesh refinement [15]), and 2) the more critical challenge of the high computational cost, which results in long simulation times even on parallel computers [16]. Therefore, optimizing PIC codes is important and necessary to increase the capabilities of the corresponding simulations [17, 18].

In PIC simulations, the kinetic modeling of plasmas is usually dominated by collective effects. Most PIC codes omit short-range Coulomb collisions between particles. At high temperatures ($\gtrsim$ 1 keV) and relatively low plasma densities ($\lesssim 10^{27}$ m^{-3}), the collision effects are usually considered marginal, and a collision-free approximation is valid in PIC simulations [19]. However, in plasmas with lower temperatures or significantly higher densities, the effect of short-range interactions between particles on the evolution of the systems needs to be addressed. Reducing the computational effort required to account for Coulomb collisions in PIC simulations has become a primary concern in the research on topics including ICF and strong-field ionization [20].

Initially, the particle distribution in plasma was determined by directly solving the Fokker–Planck kinetic equation. The key to this method is the calculation of the Rosenbluth potential to assess the contribution of Coulomb

Qian Dong and Chao-Zhi Li have contributed equally to this work.

This work was supported by the National Natural Science Foundation of China (Nos. 12375244, 12135009, and 12175310) and Natural Science Foundation of Hunan Province of China (No. 2025JJ30002).

✉ Wen Luo
wenluo-ok@163.com

✉ Tong-Pu Yu
tongpu@nudt.edu.cn

[1] Department of Physics, National University of Defense Technology, Changsha 410073, China

[2] School of Nuclear Science and Technology, University of South China, Hengyang 421001, China

collisions to the scattering coefficient [21]. However, coupling this method with PIC simulations was initially complex. In addition, this method requires numerous simulated particles to accurately calculate the Rosenbluth potential. In practical PIC simulations, most models are based on the "binary collision" proposed by Takizuka and Abe (TA77) [22]. This model replaces direct collisions by calculating the Coulomb cross-section between particles in the same cell and randomly selecting the appropriate scattering angle. In addition, this methodology enables the simultaneous consideration of multiple elastic and inelastic collision channels in a pairwise process. Lavell et al. [23] applied the corrected binary collision model originally developed for charged-particle collisions to all the scattering channels considered. They validated the accuracy and utility of their model in a wide range of plasma environments. However, in binary collision modeling, the computational efficiency (i.e., computational speed and accuracy) of high-dimensional Monte Carlo (MC) integration methods depends strongly on the sampling method used. A simple and efficient MC method for computing arbitrary ion velocity distributions was proposed by Xie et al. [24]. A method similar to Monte Carlo collision (MCC) to simulate the transition region between collisionless plasma and Coulomb collision plasma is presented in Ref. [25]. A lattice-based "collision field" concept handles the collision process between different types of particles. This has significant advantages for high-dimensional simulations. Manheimer et al. [26] developed a Coulomb collision model for electron–electron and electron–ion collisions. It was simplified to an isotropic scattering model. The N97 [27] method uses the same-order N binary-pairing procedure presented in TA77. However, it uses a different distribution function for the scattering angle. A grid-based binary model for Coulomb collisions was described by Cohen et al. [28]. It defines the unique particle counterpart of a particle on the grid by randomly sampling the velocity distribution. The model omits the process of direct pairing of particles in the simulation cell, significantly reducing the CPU runtime.

However, all these methods assume that the macroparticles have identical weights so that the real particle density is linearly proportional to the number of macroparticles in a cell. In many simulations, variable weights are necessary to render the computational task feasible. Using the appropriate pairing statistics, Miller and Combi [29] applied the TA77 method to differentially weighted particles. The algorithm preserves energy and momentum and produces suitable relaxation time scales compared with theoretical predictions. Subsequently, Nanbu and Yonemura [30] considered a Coulomb collision algorithm for weighted particles based on the cumulative properties of Coulomb collisions in plasma. The effectiveness of the proposed algorithm was verified using several examples. Higginson et al. [31] detailed a method for applying fusion to a PIC simulation with different particle weights and validated the method using analytical benchmarks in thermonuclear and beam-target fusion reactions. Unlike the work of Higginson et al. [31], Wu et al. presented an algorithm for pairwise fusion applicable to macroparticles of arbitrary weights at relativistic energies [32]. It is compatible with the Coulomb scattering algorithm widely used by Nanbu [30] and Sentoku [33] and can be implemented in the PIC simulation code with Coulomb scattering with marginal effort. Recently, Angus et al. [34] described a method for adjusting particle velocities post-scatter to restore the exact conservation of momentum and energy. They also illustrated its efficacy with various test problems.

In the present work, we described an efficient optimization method for the nuclear fusion algorithm and its implementation within the existing PIC code Smilei [35]. The aim was to develop collision-operator schemes with maximum efficiency while maintaining simulation accuracy. Based on the classical nuclear fusion algorithm described in Ref. [22], our algorithm simplifies particle pairing and reduces the computational overhead by eliminating the particle address randomization step. Collision particles are selected randomly. This significantly reduces the CPU time and improves the computational efficiency of the collision processes.

2 Optimization of the nuclear fusion algorithm

2.1 Theoretical background

The two fusion-reactant macroparticles a and b undergo fusion and create two products A and B with an energy gain of Q. This is described by Eq. (1):

$$a + b \rightarrow A + B + Q. \tag{1}$$

In the center-of-momentum (CM) frame, a macrocollision may be considered as being associated with two interacting macroparticles and the local density of their respective species. In general, the fusion probability P of this interaction in the CM frame can be expressed as:

$$P = n_{\max} v_{\mathrm{rel}} \sigma_{ab} \Delta t, \tag{2}$$

where $n_{\max}$ is the maximum density of two particles a and b. v_{rel} is the relative velocity between the particles. The fusion reaction cross-section associated with the kinetic energy of the incident particle (denoted as σ_{ab}) is expressed as follows:

$$\sigma(E_{\mathrm{cm}}) = \frac{S(E_{\mathrm{lab}})}{E_{\mathrm{cm}}} \exp\left(-\sqrt{\frac{E_{\mathrm{G}}^{\mathrm{cm}}}{E_{\mathrm{cm}}}}\right). \tag{3}$$

Here, $E_{\rm cm}$ and $E_{\rm lab}$ denote the kinetic energies of incident particles in the CM and laboratory frames, respectively. $E_{\rm G}^{\rm cm}$ represents the Gamow energy given by $E_{\rm G}^{\rm cm} = (\pi a_{\rm f} Z_1 Z_2)^2 m_{\rm r} c^2$. Here, $a_{\rm f} = e^2/\hbar c$ is the fine-structure constant, Z_i denotes the atomic number, $\hbar$ represents the reduced Planck constant, and c is the speed of light. $m_{\rm r}$ is the reduced mass, as defined by $m_{\rm r} = m_a m_b/(m_a + m_b)$. The exponential in Eq. (3) denotes the Coulomb barrier penetration probability. $S(E_{\rm lab})$ is known as the astrophysical Sfactor. It varies gradually with the kinetic energy for most nuclear reactions. According to the parametric fitting formulae of Chulick et al. [36], the values of $\sqrt{E_{\rm G}^{\rm cm}}$ for the reactions D(d,n)^{3}He and D(t,n)^{4}He are 31.40 and 34.37, respectively. The cross-section and Sfactor are parameterized as functions of the energy of the reactant particle.

Note that the transformation of the coordinate system during the implementation of nuclear fusion in the code is mandatory. When a fusion reaction occurs, the fusion products are generated in the CM frame, transformed into the laboratory frame, and finally transformed back into the simulation frame.

In the laboratory frame, the coordinate origin is fixed at a point in the laboratory similar to the target nucleus in a nuclear experiment. Particle a moves at a relative velocity of $v_{\rm rel}$. Therefore, the velocity in the simulation frame should first be transformed into the relative velocity in the laboratory frame by the transformation $v_{a,\rm lab} = |v_a - v_b|$.

The sum of the kinetic energy of the incident particle a and target nucleus b relative to the center of mass is typically referred to as the kinetic energy of their relative motion. It is denoted as $E_{\rm cm}$. It represents the kinetic energy of the CM frame. As shown in Fig. 1, the distance between the incident particle a and target nucleus b is denoted by x, whereas that between the center of mass and target nucleus is denoted by $x_{\rm c}$. According to the definition of the center of mass,

$$\frac{{\rm d}x_{\rm c}}{{\rm d}t} = \frac{m_a}{m_a + m_b}\frac{{\rm d}x}{{\rm d}t}. \tag{4}$$

Here, $\frac{{\rm d}x}{{\rm d}t}$ represents the velocity of incident particle a (denoted by v_a). Similarly, $\frac{{\rm d}x_{\rm c}}{{\rm d}t}$ represents the velocity of the center of mass (denoted by $v_{\rm cm}$). This equation indicates that particle a moves toward particle b with velocity v_a, whereas the center of mass moves toward particle b with velocity $v_{\rm cm}$.

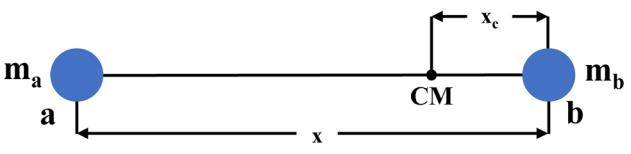

Fig. 1 (Color online) Relationship between center-of-mass velocity and incident particle velocity

Therefore, the velocity of a relative to the center of mass (or equivalently, the velocity of a in the CM frame) is given by:

$$v_{a,\rm cm} = v_{a,\rm lab} - v_{\rm cm} = \frac{m_b}{m_a + m_b} v_{a,\rm lab}. \tag{5}$$

The velocity of b relative to the center of mass (denoted by v_b in the CM frame) is equal in magnitude to the velocity $v_{\rm cm}$ of the center of mass relative to b, but opposite in direction. Hence, the following relationship is obtained:

$$v_{b,\rm cm} = v_{\rm cm} = \frac{m_a}{m_a + m_b} v_{a,\rm lab}. \tag{6}$$

The sum of the kinetic energies of particles a and b in the CM frame can be calculated using Eqs. (5) and (6).

$$E_{\rm cm} = \frac{1}{2} m_a v_{a,\rm cm}^2 + \frac{1}{2} m_b v_{b,\rm cm}^2 = \frac{1}{2}\frac{m_a m_b}{m_a + m_b} v_{a,\rm lab}^2. \tag{7}$$

Subsequently, the fusion products are generated in the CM frame with the conservation of energy and momentum. The energy released in a reaction (generally referred to as the Q value) is given by the difference between the remaining mass energies of the reactants and products. Therefore, the total kinetic energy in the CM frame after the reaction is $Q + E'$. Here, Q is the energy released, and E' represents the total kinetic energy of the product particles. This energy is then distributed among product particles A and B in proportion to their masses. In the CM frame, the total momentum of the particles is zero both before and after the reaction. This implies that $m_A V_{A,\rm cm} = m_B V_{B,\rm cm}$. Therefore, the kinetic energy of the product particles can be expressed as follows (considering particle A as an example):

$$\frac{1}{2} m_A V_{A,\rm cm}^2 = \frac{m_B}{m_A + m_B}\left(Q + \frac{m_a m_b v_{a,\rm lab}^2}{2(m_a + m_b)}\right), \tag{8}$$

where m_A and m_B are the masses of the product particles. The velocity of particle A in the CM frame is:

$$|V_{A,\rm cm}| = \sqrt{\frac{2m_B}{(m_A + m_B)m_A}\left(Q + \frac{m_a m_b v_{a,\rm lab}^2}{2(m_a + m_b)}\right)}. \tag{9}$$

To facilitate the calculation of the fusion product, the momentum space is transformed such that the velocity of the center of mass is in the z-direction. This transformation is expressed as follows:

$$R = \begin{pmatrix} \cos(\theta)\cos(\phi) & \cos(\theta)\sin(\phi) & -\sin(\theta) \\ -\sin(\phi) & \cos(\phi) & 0 \\ \cos(\phi)\sin(\theta) & \sin(\theta)\sin(\phi) & \cos(\theta) \end{pmatrix}. \tag{10}$$

Here, θ represents the polar angle of the outgoing particle in the z-CM frame, and φ represents the azimuthal angle of

the outgoing particle in this frame. In the CM framework, particle emission is assumed to be isotropic. This implies that it occurs uniformly in all directions with respect to polar angle θ. Azimuthal angle φ is selected from a uniform distribution between 0 and 2π. These angles are then used to calculate the velocity of the first product in the z-CM frame, as follows:

$$V_{A,\mathrm{cm-z}} = |V_{A,\mathrm{cm}}|(\sin\theta\cos\varphi x + \sin\theta\sin\varphi y + \cos\varphi z). \quad (11)$$

From the conservation of momentum $m_A V_{A,\mathrm{cm}} = m_B V_{B,\mathrm{cm}}$, the velocity of the fusion product B in the CM frame is

$$V_{B,\mathrm{cm-z}} = -\sqrt{\frac{m_A}{m_B}} V_{A,\mathrm{cm-z}}. \quad (12)$$

The particle velocity vector should then be transformed back into the CM frame using the inverse of the previous transformation.

$$R^{\mathrm{T}} = \begin{pmatrix} \cos(\theta)\cos(\phi) & -\sin(\phi) & \cos(\phi)\sin(\theta) \\ \cos(\theta)\sin(\phi) & \cos(\phi) & \sin(\theta)\sin(\phi) \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}, \quad (13)$$

$$V_{A,\mathrm{cm}} = R^{\mathrm{T}} V_{A,\mathrm{cm-z}}, \quad (14)$$

$$V_{B,\mathrm{cm}} = R^{\mathrm{T}} V_{A,\mathrm{cm-z}}. \quad (15)$$

After the previous coordinate transformations and velocity calculations, the particle velocity vector is rotated back to the simulation frame using an inverse transformation. Finally, the obtained fusion product is propelled and considered as a conventional macroscopic particle in a simulation frame. Although this calculation is performed for all binary pairs in each simulation cell, most of the computational time required by the binary collision operator is spent randomizing and pairing particles rather than computing the collision kinematics.

2.2 Optimizing the algorithm

Most PIC codes (including Smilei [35]) with different collision models [27, 37, 38] were inspired by TA77 [22]. The time steps of these codes follow a similar procedure, as follows:

(a) Particle groups. The particles are stored as a linked list within each simulation cell. This data structure has significant advantages for parallel computations.
(b) Particle addresses index randomization. This ensures the randomness of the subsequent particle pairing.
(c) Pairing similar particles. Pairs of particles of the same species are determined from the top of the addresses.
(d) Pairing of unlike-particles. Pairs of particles from different species are selected from the top of the addresses.

Finally, new particles are generated by the fusion of the paired particles. The original linked list is updated, and the product particle information is stored. The primary steps of the algorithm described above are summarized in Fig. 2.

The relaxation time used in the Coulomb collision calculation is more accurate than the standard approximation. However, it is computationally expensive and requires substantial CPU time. This presents a significant challenge for large-scale PIC simulations. To improve the computational efficiency while maintaining the accuracy of the Coulomb collision model, we implemented targeted optimizations using the classical binary collision algorithm.

First, we eliminated the particle address index randomization. Macroparticles may collide multiple times during the simulation period. However, the collision pair selection is, on average, performed once per execution of the optimization algorithm. This would be a significant time investment if particle address index randomization is considered at each time step. The selection of colliding particle pairs is achieved by random numbers. This ensures

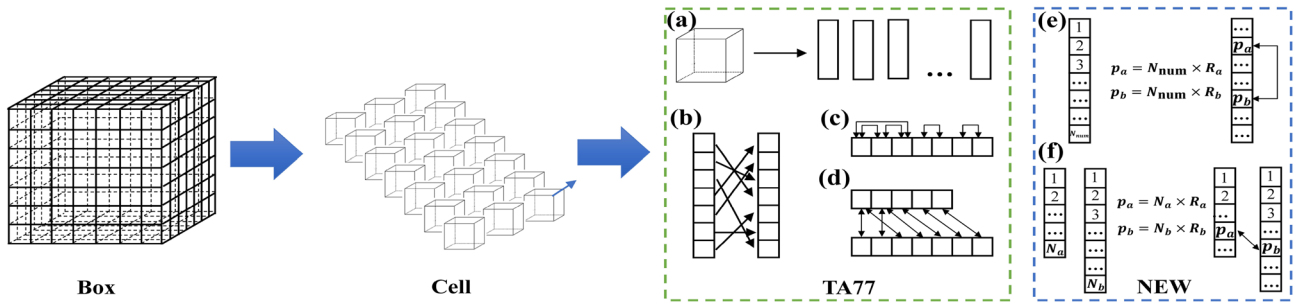

Fig. 2 (Color online) Implementation process of the classical binary collision (TA77) and optimization algorithms (NEW). **a** Grouping of particles in the simulation cell. **b** Randomly altering the index order of particles within the simulation cell. **c** Pairing of like-particles of TA77. **d** Pairing of unlike-particles of TA77. **e** Pairing of like-particles of the optimization algorithm. **f** Pairing of unlike-particles of the optimization algorithm

the randomness of particle pairing and eliminates the need for a particle address randomization operation.

We then optimized the pairing of like-particles. As illustrated in Fig. 2e, when particle pairs belong to the same type, N_{num} denotes the total number of particles in the simulation cell. Two random numbers R_a and R_b are generated subsequently ($R_a, R_b \in (0, 1]$). The indices of the paired particles are determined from $p_a = R_a N_{\text{num}}$ and $p_b = R_b N_{\text{num}}$. This step is performed in a loop until the pairing of $N_{\text{num}}/2$ particles is completed. If $p_a = p_b$ and $p_a \neq N_{\text{num}}$, p_a is incremented by one. Otherwise, it is decremented by one. It is necessary to recursively apply the above steps to the remaining simulation cells until all the cells are paired. The pseudocode is provided in Algorithm 1.

Algorithm 1 Pairing of like-particles

Input: Information regarding all macroparticles in the simulation cell; Number of particles in the cell (N_{num})
Output: Information of $N_{\text{num}}/2$ particle collision pairs

repeat
  Generate two random $R_a, R_b \in (0, 1]$;
  `// Determine the index of particle pairs`
  $p_a = N_{\text{num}} \times R_a$, $p_b = N_{\text{num}} \times R_b$, **if** $p_a = p_b$ and $p_a ! = N_{\text{num}}$ **then**
    `// If particle pairs have an equal index within valid bounds`
    $p_a = p_a + 1$;
    else
      `// If particle pairs have an equal boundary index`
      $p_a = p_a - 1$;
    end
  end
until Select $N_{\text{num}}/2$ pairs of colliding particles;
return $N_{\text{num}}/2$ particle colliding pairs;

Third, we optimized the pairing of unlike-particles. For different types of particle pairings, the total number of particles with a lower density N_a and that of particles with a higher density N_b should be determined first. This is similar to the optimization of the pairing of like-particles, where two random numbers $R_a, R_b \in (0, 1]$ are generated and the indices of the pairs are determined by $p_a = R_a N_a$ and $p_b = R_b N_b$. As shown in Fig. 2f, the above procedure is repeated until N_a particle pairing is completed. This process is then executed recursively across all the simulation cells until all the cells are particle-paired. The pseudocode is given in Algorithm 2.

Algorithm 2 Pairing of unlike-particles

Input: Information regarding all macroparticles in the cell; Number of particles with lower density in the cell (N_a); Number of particles with higher density in the cell (N_b)
Output: Information of N_a particle collision pairs

repeat
  Generate two random $R_a, R_b \in (0, 1]$;
  `// Determine different density particle index`
  $p_a = N_a \times R_a$;
  $p_b = N_b \times R_b$;
until Select N_a pairs of colliding particles;
return N_a particle colliding pairs;

The spatial grid size in this procedure is of the order of the Debye radius. Because it is unnecessary to consider the Coulomb collisions between particles spaced farther apart than the Debye radius, the interactions between the particles in different simulation cells can be omitted. This also enables parallel computation of the simulation procedures. A flowchart comparison between the classic binary collision algorithm (TA77) and optimization algorithm (NEW) is shown in Fig. 3.

3 Benchmarks

To evaluate the efficiency of the optimization algorithm, we present three typical simulations: D(d,n)^{3}He and D(t,n)^{4}He thermonuclear fusion and D(d,n)^{3}He beam-target fusion. The results demonstrate that the optimization algorithm computes the Coulomb collision process with maximum efficiency without compromising physical accuracy. These benchmarks were implemented using the PIC code Smilei [35]. All the tests were performed on a disabled field. Simulations can use cross-sectional data derived from experimental measurements or parametric fitting equations. In this case, the second approach was preferred to ensure a direct comparison of the simulation results with the theoretical predictions.

It is important to emphasize that the optimization algorithm operates on probability-driven random pairing, rather than on the complete traversal of all particles. Although the number of pairings is limited by lower-density species, the optimization algorithm leverages MC methods to achieve a robust sampling of high-density species. In nuclear fusion, the velocity distribution of ions follows the Maxwell–Boltzmann distribution, wherein energetic particles (although sparse) contribute significantly to the fusion reaction [31]. Random pairing cumulatively covers the high-energy region using multiple independent samplings

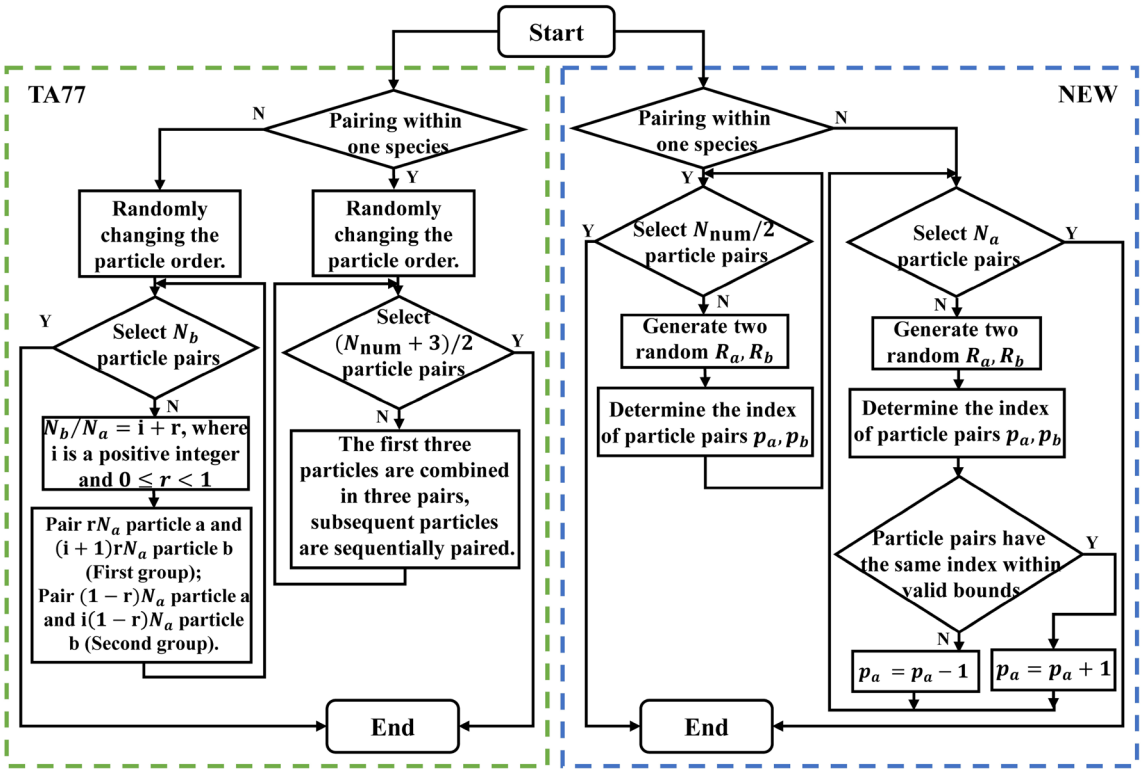

Fig. 3 (Color online) Flow chart comparison between the classical binary collision algorithm (TA77) and optimization algorithm (NEW)

(cycling each time step) rather than relying on a single-step traversal. Consequently, the statistical rise and fall of random pairing is attenuated by time-averaging effects in the long-duration simulations. This ensures the accuracy of the kinetic evolution.

3.1 D(d,n)^{3}He thermonuclear fusion

In the first benchmark, we created a simulation box with dimensions $L_x = 40\pi c/\omega_r$ and $L_y = 40\pi c/\omega_r$. We divided it into a 100×100 grid, with each cell containing an equal number of D ions. The number density of D ions was 10^{20}cm^{-3}, and the particles were weighted unequally. Thermonuclear fusion occurs when ions attain high temperatures. Because the cross-section for fusion depends strongly on the relative velocity of the particles, most fusion events occur between ions in the tails of the distribution, with kinetic energies several times higher than the mean energy of the plasma. Simulations were performed at different temperatures to evaluate the fusion reactivity and neutron spectra.

The neutron energy spectra obtained at different temperatures are shown in Fig. 4. TA77 and the optimization algorithm yielded similar neutron energy spectra. The full

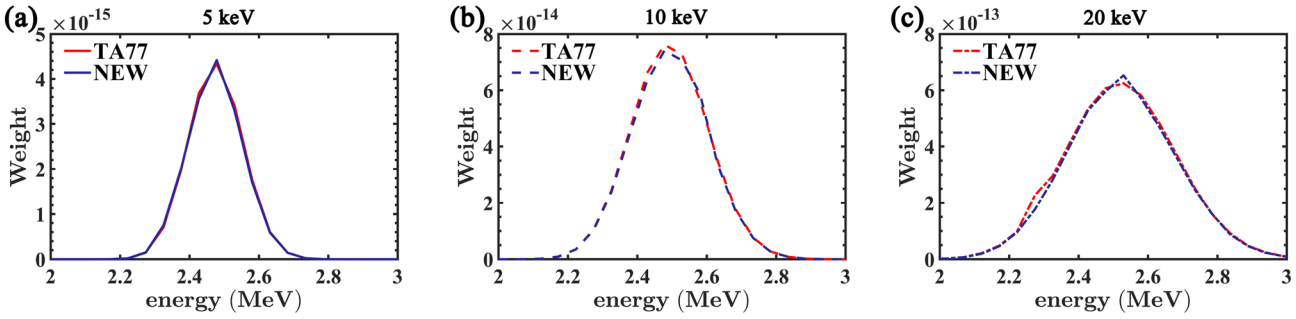

Fig. 4 (Color online) Neutron energy spectra for D(d,n)^{3}He fusion in plasmas with ion temperatures of **a** 5 keV, **b** 10 keV, and **c** 20 keV. Here, the simulation results with TA77 are shown in red and those with the optimized algorithm are shown in blue

width at half maximum (FWHM) of the neutron energy spectra increased as the initial ion temperature did. To validate the accuracy of the energy spectra generated by the optimization algorithm, the number of particles per cell (PPC) was fixed at 1000 for all the simulations. As illustrated in Fig. 5a, by varying the initial temperature of the D ions, we compared the effective FWHM of the simulated energy spectra with the analytical fit of Ballabio [39] using relativistic kinematics. The results reveal a remarkable agreement. The ultimate goal of the optimization algorithm is to improve the computational efficiency without altering the underlying physics. Figure 5b presents the CPU runtime for the D(d,n)^{3}He thermonuclear fusion simulation with an initial temperature of 10 keV. Compared with the simulation results of TA77, the CPU runtime reduced by 30.5%. Although the improvement in computational efficiency may not be universal, it is more evident when simulating a larger number of particles in complex collisions.

3.2 D(t,n)^{4}He thermonuclear fusion

To test the optimization algorithm in the case of unlike-particles and illustrate the case of unequal weighting, a simulation was conducted for D(t,n)^{4}He thermonuclear fusion. The initial conditions were similar to those of the first benchmark, with a density of 10^{20}cm^{-3} for both ions and an initial temperature of 5 keV. The neutron energy spectra were benchmarked by varying the initial numbers of D and T ions in each cell. As shown in Fig. 6a, when the initial number of D ions was less than that of the T ions ($N_D = 100$ ppc, $N_T = 1000$ ppc), the optimization algorithm matched the neutron energy spectrum from the TA77 simulation. In the other case, with the particle number reversed ($N_D = 1000$ ppc, $N_T = 100$ ppc), the results were as anticipated (see Fig. 6b). Furthermore, when equal weights were maintained and the initial density ratios of the D and T ions were varied, the neutron energy distribution was unaffected

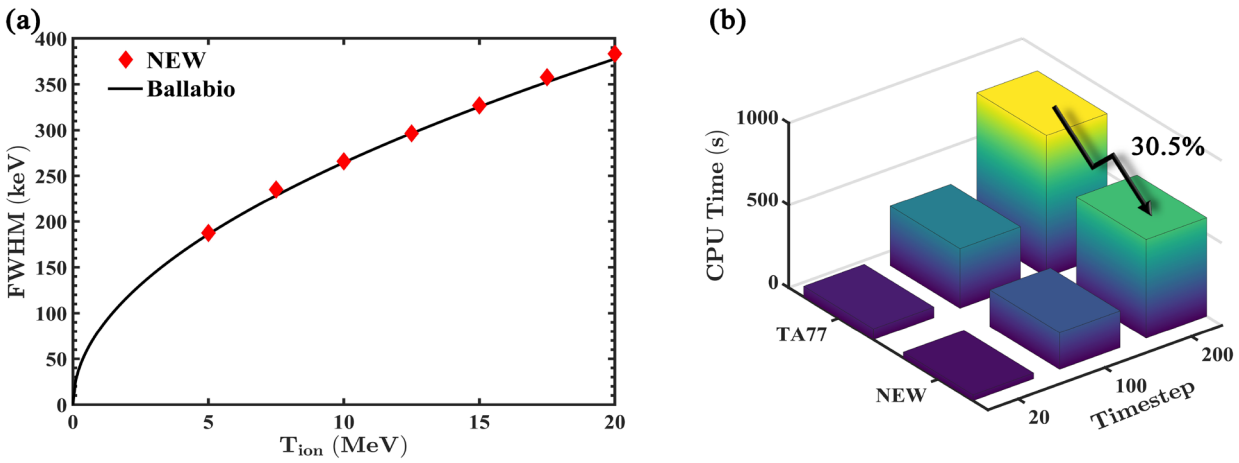

Fig. 5 (Color online) **a** Neutron energy full width at half maximum (FWHM) simulated by the optimized algorithm (NEW) is compared with the FWHM from the work of Ballabio [39]. **b** Total computation time for the D(d, n)^{3}He thermonuclear fusion simulation in both cases

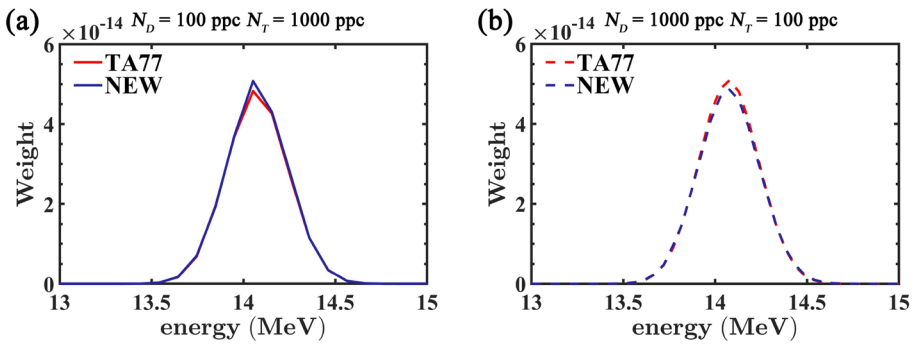

Fig. 6 (Color online) Neutron energy spectra for D(t,n)^{4}He fusion in plasmas with ion temperatures of 5 keV. **a** Case where the initial number of particles is $N_D = 100$ ppc, $N_T = 1000$ ppc, respectively. **b** Case where the initial number of particles is $N_D = 1000$ ppc and $N_T = 100$ ppc. Here, the simulation results with TA77 are shown in red and those with the optimized algorithm are shown in blue

by the variations in initial particle density. It influenced only the peak of the energy spectrum. Because the nuclear reaction mechanism of the second benchmark is different from that of the first, we performed multiple sets of simulations. The evolution of the neutron energy FWHM with ion temperature is shown in Fig. 7a. It again demonstrates the remarkable agreement between the simulation results of the optimization algorithm (NEW) and the theory (Ballabio). In addition, we compared the computational efficiency of $D(t,n)^4He$ thermonuclear fusion in both cases, as shown in Fig. 7b. This revealed a 25.5% reduction in the CPU runtime of the optimized algorithm compared with the TA77 results.

3.3 $D(d,n)^3He$ beam-target fusion

Two distinct tests were conducted to investigate the feasibility of the $D(d,n)^3He$ beam-target fusion using an optimization algorithm. In the simulations, $D(d,n)^3He$ beam-target fusion occurred when a high-energy beam impacted a cold stationary D ions target. In this case, the kinetic energy of the beam determined the magnitude of the fusion cross-section.

One group of D ions was considered as a stationary background with a density of $10^{20}cm^{-3}$. The other group with an equal density was considered as a projectile. Two simulations with projectile velocities of $0.05c$ and $0.1c$ were conducted. The resulting neutron spectra are presented in Fig. 8. Here, Fig. 8a and b illustrates the simulation results with initial velocities of 0.05 c and 0.1 c, respectively. As shown, the optimization algorithm accurately reproduced the energy spectra of the neutrons. Because the optimization algorithm partially removed the particle traversal steps, we illustrate the relative error in the total kinetics during the simulation in Fig. 9a. For the optimization algorithm, the peak of the relative error stabilized at ±0.02%. This is an insignificant energy deviation considering the effect of statistical fluctuations during the simulation. In addition, as shown in Fig. 9b, the optimization algorithm significantly reduced the CPU runtime (by 32.5%).

4 Conclusion

In this study, we focused on the computationally inefficient problem of multiparticle complex collisions in PIC simulations and optimized the classical PIC binary Coulomb

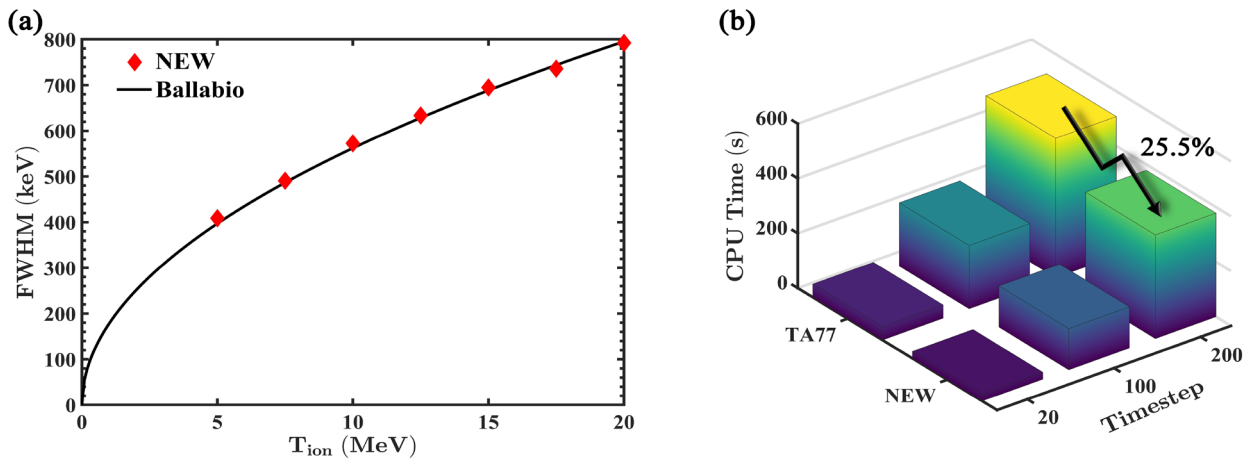

Fig. 7 (Color online) **a** Neutron energy FWHM simulated by the optimized algorithm (NEW) is compared with the FWHM from the work of Ballabio [39]. **b** Total computation time for the $D(t,n)^4He$ thermonuclear fusion simulation in both cases

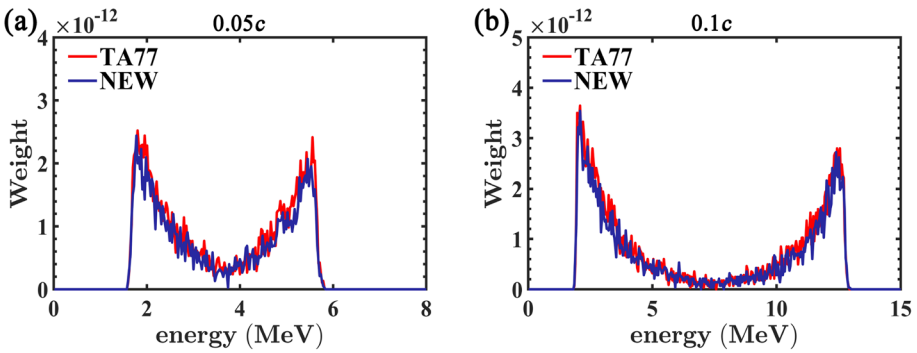

Fig. 8 (Color online) **a** Neutron energy spectra for $D(d,n)^3He$ beam-target fusion when the initial projectile velocity is 0.05 c. **b** Neutron energy spectra for $D(d,n)^3He$ beam-target fusion when the initial projectile velocity is 0.1 c. Here, the simulation results with TA77 are shown in red and those with the optimized algorithm are shown in blue

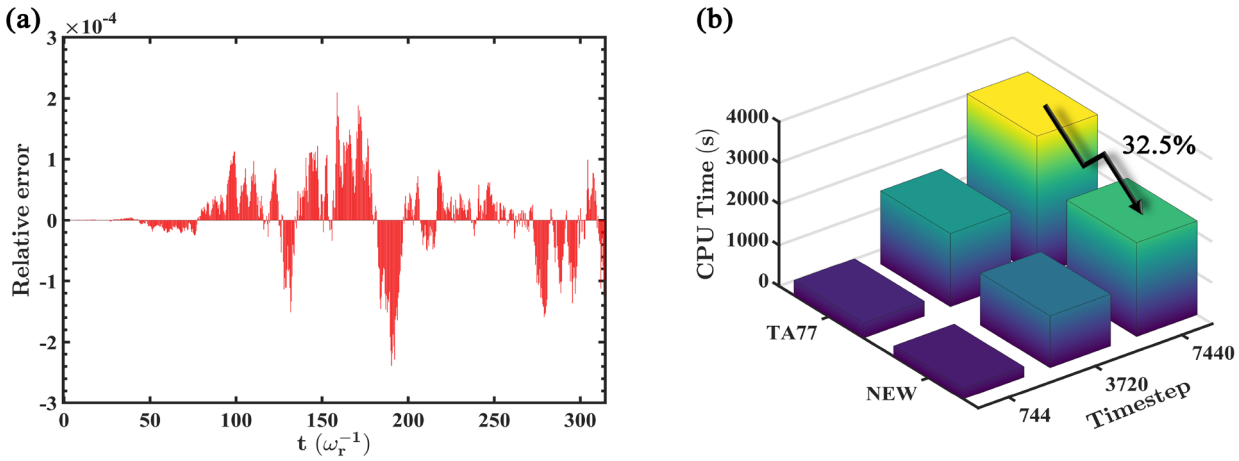

Fig. 9 (Color online) **a** Relative error in the total kinetic energy owing to the use of the optimization algorithm during the simulation. **b** Total computation time for the D(d,n)^{3}He beam-target fusion simulation in both cases

collision algorithm. The particle-pairing process was simplified by removing the particle address randomization step. Additionally, collision particle pairs were selected using random numbers. This eliminated the need to traverse all the particle address information within the simulation cell and significantly reduced the CPU runtime. The optimized algorithm was benchmarked in like-particles D(d,n)^{3}He, unlike-particles D(t,n)^{4}He, thermonuclear fusion, and beam-target fusion. The results show that the CPU runtime can be generally reduced by approximately 30% while maintaining physical accuracy. The minimal errors in energy and momentum enable the optimized algorithm to minimize its effect on the underlying physics.

For plasmas with a significantly high density ($\gtrsim 10^{27}$ m^{-3}) or significantly low temperature ($\lesssim 1$ keV), the collision frequency may increase significantly. This, in turn, would amplify the statistical error in the optimization algorithm (NEW). Under such conditions, the particle correlation effects (e.g., collective oscillations or strong coupling) may exceed the assumptions of the algorithm for independent collision events. We plan to further validate the robustness of the optimization algorithm in future studies. The generalization for parallel computations is apparent. This is because the optimization algorithm is performed cell-by-cell, and particles are "sorted" during the entire simulation. Therefore, the performance of the optimization algorithm is likely to improve substantially as the number of particles and complexity of the simulation system increase. More importantly, the optimization algorithm enables the evaluation of problems that would otherwise not be accessible owing to computational resource limitations. Thus, it provides technical support for subsequent large-scale particle collision simulations, including ICF and laser-driven neutron sources.

Author contributions All authors contributed to the study conception and design. Material preparation, data collection, and analysis were performed by Qian Dong, Chao-Zhi Li, Chen Xie, Zhi-Dong Chen, De-Bin Zou, Wen Luo, and Tong-Pu Yu. The first draft of the manuscript was written by Qian Dong, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data availability The data that support the findings of this study are openly available in Science Data Bank at https://cstr.cn/31253.11.sciencedb.j00186.00997 and https://www.doi.org/10.57760/sciencedb.j00186.00997

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. E.P. Alves, W.B. Mori, F. Fiuza et al., Numerical heating in particle-in-cell simulations with Monte Carlo binary collisions. Phys. Rev. E **103**, 013306 (2021). https://doi.org/10.1103/PhysRevE.103.013306
2. J.S. Ross, D.P. Higginson, D. Ryutov et al., Transition from collisional to collisionless regimes in interpenetrating plasma flows on the national ignition facility. Phys. Rev. Lett. **118**, 185003 (2017). https://doi.org/10.1103/PhysRevLett.118.185003
3. X.Y. An, M. Chen, J.L. Liu et al., Modeling of axion and electromagnetic fields interaction in particle-in-cell simulations. Matter Radiat. Extrem. **9**, 067204 (2024). https://doi.org/10.1063/5.0226159
4. E.M. Campbell, V.N. Goncharov, T.C. Sangster et al., Laser-direct-drive program: promise, challenge, and path forward. Matter Radiat. Extrem. **2**, 37–54 (2017). https://doi.org/10.1016/j.mre.2017.03.001
5. X.R. Jiang, F.Q. Shao, D.B. Zou et al., Energetic deuterium-ion beams and neutron source driven by multiple-laser interaction with pitcher-catcher target. Nucl. Fusion **60**, 076019 (2020). https://doi.org/10.1088/1741-4326/ab91f9

6. A. McIlvenny, D. Doria, L. Romagnani et al., Selective ion acceleration by intense radiation pressure. Phys. Rev. Lett. **127**, 194801 (2021). https://doi.org/10.1103/PhysRevLett.127.194801
7. T. Minami, C.M. Chu, O. McCusker et al., Ion acceleration with an intense short-pulse laser and large-area suspended graphene in an extremely thin target regime. High-energy density Phys. **55**, 101195 (2025). https://doi.org/10.1016/j.hedp.2025.101195
8. J.H. Cheng, Q. Xiao, J.G. Deng et al., Laser-assisted α decay of the deformed odd-A nuclei. Nucl. Sci. Tech. **36**, 69 (2025). https://doi.org/10.1007/s41365-024-01610-2
9. H. Chen, S.C. Wilks, D.D. Meyerhofer et al., Relativistic quasi-monoenergetic positron jets from intense laser-solid interactions. Phys. Rev. Lett. **105**, 015003 (2010). https://doi.org/10.1103/PhysRevLett.105.015003
10. J. Zhao, Y.T. Hu, Y. Lu et al., All-optical quasi-monoenergetic GeV positron bunch generation by twisted laser fields. Commun. Phys. **5**, 15 (2022). https://doi.org/10.1038/s42005-021-00797-9
11. F. Wan, C. Lv, K. Xue et al., Simulations of spin/polarization-resolved laser–plasma interactions in the nonlinear QED regime. Matter Radiat. Extrem. **8**, 064002 (2023). https://doi.org/10.1063/5.0163929
12. T.P. Yu, K. Liu, J. Zhao et al., Bright X/γ-ray emission and lepton pair production by strong laser fields: a review. Rev. Mod. Plasma Phys. **8**, 24 (2024). https://doi.org/10.1007/s41614-024-00158-3
13. J.D. Blahovec, L.A. Bowers, J.W. Luginsland et al., 3-D ICEPIC simulations of the relativistic klystron oscillator. IEEE Trans. Plasma Sci. **8**, 24 (2024). https://doi.org/10.1109/27.887733
14. Q. Dong, B.L. Wang, X.J. Duan et al., A dynamical particle merging and splitting algorithm for particle-in-cell simulations. Comput. Phys. Commun. **294**, 108913 (2024). https://doi.org/10.1016/j.cpc.2023.108913
15. S.M. Robert, J.L. Cambie, Octree particle management for DSMC and PIC simulations. J. Comput. Phys. **327**, 943–966 (2016). https://doi.org/10.1016/j.jcp.2016.01.020
16. J.P. Verboncoeur, Particle simulation of plasmas: review and advances. Plasma Phys. Control. Fusion **47**, A231 (2005). https://doi.org/10.1088/0741-3335/47/5A/017
17. E. Kawamura, C.K. Birdsall, V. Vahedi, Physical and numerical methods of speeding up particle codes and paralleling as applied to RF discharges. Plasma Sources Sci. Technol. **9**, 413 (2000). https://doi.org/10.1088/0963-0252/9/3/319
18. S.H. Ren, H.Y. Li, J.R. Shi et al., Calculation algorithm for the space charge force of a train with infinite bunches. Nucl. Sci. Tech. **36**, 96 (2025). https://doi.org/10.1007/s41365-025-01673-9
19. M. Ramsay, Dissertation, University of Warwick, (1975)
20. D. Tskhakaya, R. Schneider, Optimization of PIC codes by improved memory management. J. Comput. Phys. **225**, 829–839 (2007). https://doi.org/10.1016/j.jcp.2007.01.002
21. R.J. Procassini, B.I. Cohen, A comparison of particle-in-cell and Fokker-Planck methods as applied to the modeling of auxiliary-heated mirror plasmas. J. Comput. Phys. **102**, 39–48 (1992). https://doi.org/10.1016/S0021-9991(05)80003-8
22. T. Takizuka, H. Abe, A binary collision model for plasma simulation with a particle code. J. Comput. Phys. **25**, 205–219 (1977). https://doi.org/10.1016/0021-9991(77)90099-7
23. M.J. Lavell, A.J. Kish, A.T. Sexton et al., Verification of a Monte Carlo binary collision model for simulating elastic and inelastic collisions in particle-in-cell simulations. Phys. Plasmas **31**, 043902 (2024). https://doi.org/10.1063/5.0190352
24. H.S. Xie, A simple and fast approach for computing the fusion reactivities with arbitrary ion velocity distributions. Comput. Phys. Commun. **292**, 108862 (2023). https://doi.org/10.1016/j.cpc.2023.108862
25. M.E. Jones, D.S. Lemons, R.J. Mason et al., A grid-based Coulomb collision model for PIC codes. J. Comput. Phys. **123**, 169–181 (1996). https://doi.org/10.1006/jcph.1996.0014
26. W.M. Manheimer, M. Lampe, G. Joyce, Langevin representation of Coulomb collisions in PIC simulations. J. Comput. Phys. **138**, 563–584 (1997). https://doi.org/10.1006/jcph.1997.5834
27. K. Nanbu, Momentum relaxation of a charged particle by small-angle Coulomb collisions. Phys. Rev. E **56**, 7314–7314 (1997). https://doi.org/10.1103/PhysRevE.56.7314
28. B.I. Cohen, A.M. Dimits, D.J. Strozzi, A grid-based binary model for Coulomb collisions in plasmas. J. Comput. Phys. **234**, 33–43 (2013). https://doi.org/10.1016/j.jcp.2012.08.046
29. R.H. Miller, M.R. Combi, A Coulomb collision algorithm for weighted particle simulations. Geophys. Res. Lett. **21**, 1735–1738 (1994). https://doi.org/10.1029/94GL01835
30. K. Nanbu, S. Yonemura, Weighted particles in Coulomb collision simulations based on the theory of a cumulative scattering angle. J. Comput. Phys. **145**, 639–654 (1998)
31. D.P. Higginson, A. Link, A. Schmidt, A pairwise nuclear fusion algorithm for weighted particle-in-cell plasma simulations. J. Comput. Phys. **388**, 439–453 (2019). https://doi.org/10.1016/j.jcp.2019.03.020
32. D. Wu, Z.M. Sheng, W. Yu et al., A pairwise nuclear fusion algorithm for particle-in-cell simulations: weighted particles at relativistic energies. AIP Adv. **11**, 075003 (2021). https://doi.org/10.1063/5.0051178
33. Y. Sentoku, A.J. Kemp, Numerical methods for particle simulations at extreme densities and temperatures: weighted particles, relativistic collisions and reduced currents. J. Comput. Phys. **227**, 6846–6861 (2008). https://doi.org/10.1016/j.jcp.2008.03.043
34. J.R. Angus, Y.C. Fu, V. Geyko et al., Moment-preserving Monte-Carlo Coulomb collision method for particle codes. J. Comput. Phys. **531**, 113927 (2025). https://doi.org/10.1016/j.jcp.2025.113927
35. J. Derouillat, A. Beck, F. Pérez et al., Smilei: a collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation. Comput. Phys. Commun. **222**, 351–373 (2018). https://doi.org/10.1016/j.cpc.2017.09.024
36. G.S. Chulick, Y.E. Kim, R.A. Rice et al., Extended parameterization of nuclear-reaction cross sections for few-nucleon nuclei. Nucl. Phys. A **551**, 255–268 (1993). https://doi.org/10.1016/0375-9474(93)90481-C
37. V. Vahedi, M. Surendra, A Monte Carlo collision model for the particle-in-cell method: applications to argon and oxygen discharges. Comput. Phys. Commun. **87**, 179–198 (1995). https://doi.org/10.1016/0010-4655(94)00171-W
38. A.J. Christlieb, R. Krasny, J.P. Verboncoeur, A treecode algorithm for simulating electron dynamics in a Penning-Malmberg trap. Comput. Phys. Commun. **164**, 306–310 (2004). https://doi.org/10.1016/j.cpc.2004.06.076
39. L. Ballabio, J. Källne, G. Gorini, Relativistic calculation of fusion product spectra for thermonuclear plasmas. Nucl. Fusion **38**, 1723 (1998). https://doi.org/10.1088/0029-5515/38/11/310