JOURNAL OF COMPUTATIONAL PHYSICS **128,** 209–222 (1996)
ARTICLE NO. 0204

Vector Implementation of Nonlinear Monte Carlo Coulomb Collisions

W. X. Wang,* M. Okamoto,*,† N. Nakajima,*,† and S. Murakami*,†

**Department of Fusion Science, The Graduate University for Advanced Studies, Nagoya 464-01, Japan; and †Theory and Data Analysis Division, National Institute for Fusion Science, Nagoya 464-01, Japan*

Received March 7, 1995; revised January 25, 1996

A nonlinear Monte Carlo collision model based on Coulomb binary collisions is presented with emphasis on its efficient implementation. The presented Monte Carlo collision operator, which has a simple form, fulfills particle number, momentum, and energy conservation laws quasi-locally and is equivalent to the nonlinear Fokker–Planck operator in the limit of infinitesimally small time interval of the binary collisions. Two vectorizable algorithms are designed for its fast implementation. Drastic speedup is obtained by the algorithms on vector computers. Various test simulations regarding relaxation processes, electrical conductivity, etc. are carried out in the velocity space. The test results, which are in good agreement with theories, and timing results on vector computers show that the present collision model is practically applicable. © 1996 Academic Press, Inc.

1. INTRODUCTION

Collisional transport is a fundamental aspect of the physics of magnetically confined plasma. Neoclassical theory of transport, which has been well developed, is valid under two essential assumptions, namely $\rho_p \ll L_T$, L_n, and $M_p \ll 1$, where ρ_p is poloidal Larmor radius, L_T and L_n are scale lengths of the plasma temperature and the density, respectively, and M_p is the poloidal Mach number. In the edge region of toroidal systems, the situations with $\rho_p \geq L_T$, L_N and $M_p \geq 1$ are often observed (e.g., H-mode). In this case, the standard neoclassical theory cannot be applied. Monte Carlo simulation [1–3] may provide a powerful way for solving the problems. We are developing a Monte Carlo simulation code for the purpose of studying neoclassical transport near edge. The Monte Carlo method can be briefly summarized as follows. The simulation region is divided into a lot of spatial cells such that plasma in each cell can be considered to be uniform. Initially, particles are distributed randomly in each cell and in the velocity space in terms of given plasma conditions such as density profile and temperature profile. The drift motion of each particle is then followed in the magnetic field in which MHD equilibrium is employed. In the process of following particle drift motion, Coulomb binary collisions are introduced by an appropriate Monte Carlo operator. Finally, the transport properties of interest are evaluated in terms of relevant information.

Accurate and efficient calculation of Coulomb collisions is an essential step in the simulation. In a particle simulation model, Coulomb collisions can be implemented by a Monte Carlo operator. Construction of the Monte Carlo collision operator has been motivated by studies of many interested processes involving collisions in plasma physics, for example, plasma heating [4–6] in torus, gyrokinetic simulation [7–9]. Shanny *et al.* [10], for the purpose of electron plasma simulation, and Boozer and Kuo-Petravic [1], for the purpose of collisional transport in a stellerator, introduced respectively the Monte Carlo operators based on Lorentz gas collision model which describes electron–ion collisions. The Monte Carlo operator for the orbit-averaged Fokker–Planck equation was given by Eriksson and Hellender [6] and White *et al.* [11]. The application of such linear operators is limited due to, fundamentally, lack of momentum conservation. The momentum conservation, as required by the exact Fokker–Planck collision operator, is a necessary condition for the ambipolarity of particle fluxes and, therefore, is essential in many cases for correct simulation, for example, in the calculations of plasma flow (rotation) and bootstrap current. A linearized Fokker–Planck operator with conservation of energy and momentum was proposed by Catto and Tsang [12], as well as Xu and Rosenbluth [13]. In this operator, the source-sink terms which are determined from conservation of momentum and energy are added to the drag-diffusion terms that constitute a test-particle operator. The test-particle operator was implemented by a Monte Carlo scheme in the gyrokinetic particle simulation [13] to model the effect of ion–ion collisions on ion-temperature-gradient modes. Dimits and Cohen [14] presented the algorithms which implemented the linear collision operator with the energy and momentum conserving terms included, for δf particle simulation [9, 15, 16]. They treated the conserving terms as source terms in a gyrokinetic equation to restore energy and momentum conservation. Based on the discretization of linearized Fokker–Planck gyrokinetic equation, Tassarotto, White, and Zheng [17, 18] constructed the linear Monte Carlo operators, including momentum and energy conserving terms. However, their numerical implementations are not straightforward and convenient because of the complicated forms and tedious calculations of the conserving terms. A pioneer nonlinear Monte Carlo collision operator of a PIC model was proposed by Takizuka and Abe [19]. Their method was extended to gyrokinetic simulation by Ma, Sydora, and Dawson [20]. In this model, the scattering angle of a binary collision obeys a Gaussian distribution.

Which collision operator should be employed depends on which physical problem is considered. The linear operators made a common assumption that the background particles with which test particles collide are nearly Maxwellian. The change of a test-particle velocity is calculated with fewer floating-point operands since background particles are assumed to be Maxwellian. Thus, the linear collision operators have merit in their efficient implementation, compared with nonlinear operators (binary collision models). However, in simulating the plasma far from equilibrium, a nonlinear collision model may provide a more accurate description for Coulomb collisions, since the change of a particle velocity is determined by the binary collisions with neighboring particles which form a local background. Thus the non-Maxwellian effect can be taken into account. On the other hand, a large number of binary collisions which are required to ensure good statistics give rise to large enhancement of floating-point calculations. In the practical application of a nonlinear operator, the efficient implementation is a key ingredient.

In this paper, a nonlinear Monte Carlo operator is presented with emphasis on its efficient implementation. This operator is simple in its form and easy to implement. The basic features of the operator are that it quasi-locally fulfills basic conservation laws, namely, momentum, energy, and particle number conservations, which are characteristics of the exact Fokker–Planck operator; it is equivalent to the Fokker–Planck operator of Landau form. As a consequence, the basic collisional transport properties can be correctly described. For example, in particular, strict ambipolarity of particle fluxes is obtained automatically. The collision operator presented here can compare to the model due to Takizuka and Abe [19] in that both describe a nonlinear Landau collision integral, and it is similar to the latter in approach as both are binary collision models. In this paper, much attention is paid to the fast implementation on vector computers. Two vectorizable algorithms are designed, which as a key ingredient, extend the uses of the operator.

The paper is organized as follows. In Section 1, the Monte Carlo operator is presented in its original form and construction of the operator is detailed in order to demonstrate basic properties of the operator. Section 2 is devoted to the fast implementation of the operator. Two vectorizable algorithms of practical application are presented. In Section 3, the operator is tested carefully in the velocity space. The tests include temperature relaxation processes and electric conductivity, etc., as well as the timing on computers. The summary and discussion are given in Section 4.

2. MONTE CARLO COLLISION OPERATOR

A general and useful Monte Carlo collision operator should satisfy a twofold requirement. Physically, it must be applicable to the problem to be solved. Generally speaking, it should fulfill all the conservation laws, i.e., the conservation of particle number, momentum, and energy. Strictly, it is required to be equivalent to the nonlinear Fokker–Planck operator to perform an accurate simulation in which the collisional process plays an important role. Numerically, it should be feasible and convenient for implementation. In particular, the computational cost in time and memory should be acceptable by a current computer.

Here, first we present the Monte Carlo operator in its original form, which indeed satisfies the requirement in physics mentioned above. The Monte Carlo operator is constructed for the Coulomb binary collisions in the local space, a spatial cell with uniform plasma. Without loss of generality, a simple plasma system (electrons plus one species of ions) is employed for the convenience of presentation. Consider a spatial cell with electron density n_e and ion density n_i and, correspondingly, the model system with N_e electrons and N_i ions. We here impose a condition $N_e/n_e = N_i/n_i$, which is the representation, in the model system, of plasma neutrality. For a quasi-neutral plasma, Coulomb collision interactions between two particles occur within the distance of order λ_D due to the Debye shielding, where λ_D is the Debye radius. Thus, the typical size of a cell is the Debye radius. In many cases, however, one can extend the cell size if the plasma property across each cell does not vary substantially.

Let each particle collide with all other particles in the same cell in a time interval Δt. In the calculation of collisions, the particle positions in a cell are trivial and not of concern. In practice, the Monte Carlo operator gives the particle random kicks in velocity space with appropriate magnitudes and directions. The velocity alteration of a particle $a(m_a, \mathbf{v}_a, e_a)$ during a collision with a particle $b(m_b, \mathbf{v}_b, e_b)$ in a time interval Δt is determined as

$$\Delta \mathbf{v}_a = \frac{m_b}{m_a + m_b}(e^{\varepsilon \hat{n} \times} - 1)\mathbf{u}, \tag{1}$$

with small parameter ε given by

$$\frac{\varepsilon^2 u^3}{3} = \Delta t(4\pi e_a^2 e_b^2 \ln \Lambda)\left(\frac{1}{m_a} + \frac{1}{m_b}\right)^2 \gamma. \tag{2}$$

Justification of Eq. (1) with Eq. (2) will be shown later. The operation of the operator $e^{\varepsilon\hat{n}\times}$ on an arbitrary vector $\mathbf{w}$ is defined by the Taylor series as

$$e^{\varepsilon\hat{n}\times}\mathbf{w} = \mathbf{w} + \varepsilon\hat{n} \times \mathbf{w} + \frac{1}{2!}\varepsilon^2\hat{n} \times \hat{n} \times \mathbf{w} + ...,$$

$\mathbf{u} = \mathbf{v}_a - \mathbf{v}_b$ is the relative velocity before the collision, ln Λ is the Coulomb logarithm, and $\hat{n}$ is a random unit vector with a uniform distribution. The factor γ in Eq. (2) takes

$$\gamma = \begin{cases} n_b/N_b = n_a/N_a, & \text{unlike particle collisions} \\ n_a/(N_a - 1), & \text{like particle collisions } (a\text{-species}). \end{cases}$$

In the implementation, the operation $(e^{\varepsilon\hat{n}\times} - 1)\mathbf{u}$ in Eq. (1) is performed via the following approximation,

$$(e^{\varepsilon\hat{n}\times} - 1)\mathbf{u} = (\sin \varepsilon)\hat{n} \times \mathbf{u} + (1 - \cos \varepsilon)\hat{n} \times \hat{n} \times \mathbf{u}. \tag{3}$$

$\varepsilon \ll 1$ gives restriction to the step size Δt of the collision integration.

Next, we shall address the construction of the Monte Carlo operator and demonstrate the properties of the operator. Consider Coulomb collision of two model particles a and b, as shown in Fig. 1a. In three-dimensional velocity space, there are six quantities (velocity components of the two particles) that can change during the collision. However, energy conservation and momentum conservation impose four constraints. Thus, only two parameters can freely change. These two free parameters correspond to the impact parameter ρ (or scattering angle θ) and the azimuthal angle ϕ shown in Fig. 1b. The fact that there are two free parameters implies that two random variables are included in a 3D (in velocity space) Monte Carlo operator. In Takizuka and Abe's model [19] (hereafter referred as the T.A. model), they are selected to be direction angles Θ and Φ of the velocity $\mathbf{u}'$ after collision relative to the velocity $\mathbf{u}$ before collision (Fig. 1c). Here, two random variables are combined and selected to be a random unit vector $\hat{n}$ of the uniform distribution, as is seen later.

With momentum conservation,

$$m_a\mathbf{v}_a + m_b\mathbf{v}_b = m_a\mathbf{v}'_a + m_b\mathbf{v}'_b, \tag{4}$$

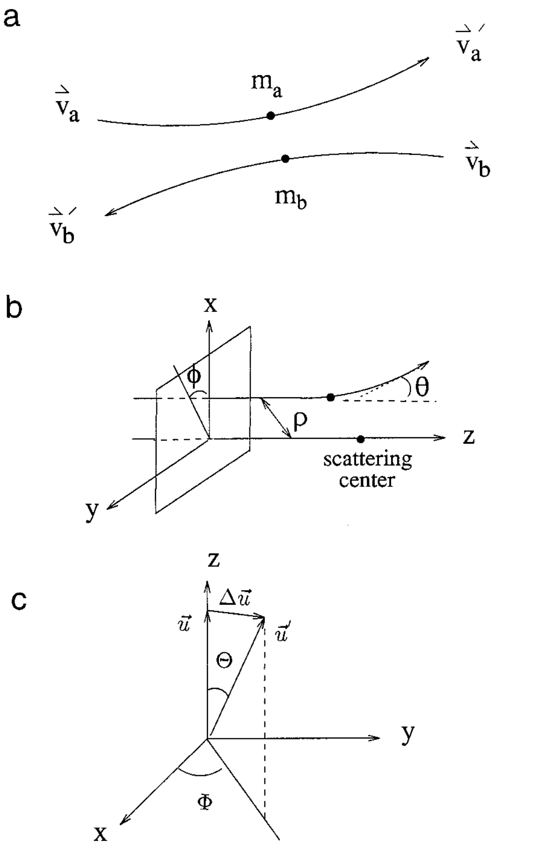

FIG. 1. A sketch of Coulomb binary collision.

energy conservation requires that the magnitude of a relative velocity does not change during the collision, i.e.,

$$|\mathbf{u}'| = |\mathbf{u}|. \tag{5}$$

Then, set the relative velocity after the collision as

$$\mathbf{u}' = \tilde{R}\mathbf{u} = [e^{\varepsilon\hat{n}\times}]\mathbf{u}, \tag{6}$$

where ε is an infinitesimal and nondimensional parameter to be determined later. It is readily shown that $\mathbf{u}'$ given by Eq. (6) satisfies Eq. (5). From Eqs. (4) and (6) we obtain Eq. (1), which, through $\hat{n}$ and ε, determines the change of velocity after the collision. In practical implementation, the use of approximation (3) does not violate Eq. (5), and therefore does not violate energy conservation.

The remaining task is to determine ε. ε is determined from the requirement that the Monte Carlo operator expressed by Eqs. (1) and (3) must be equivalent to the exact Fokker–Planck operator,

$$C_{ab} = -\frac{\partial}{\partial \mathbf{v}}\left[\frac{\langle\Delta\mathbf{v}\rangle}{\Delta t}f_a\right] + \frac{1}{2}\frac{\partial}{\partial \mathbf{v}}\frac{\partial}{\partial \mathbf{v}}\colon\left[\frac{\langle\Delta\mathbf{v}\Delta\mathbf{v}\rangle}{\Delta t}f_a\right], \tag{7}$$

where $\langle\Delta\mathbf{v}\rangle/\Delta t$ and $\langle\Delta\mathbf{v}\Delta\mathbf{v}\rangle/\Delta t$ are friction coefficient and diffusion tensor, respectively. To this end, we calculate the corresponding coefficients of the Monte Carlo operator as (if $b = a$, $N_b \rightarrow N_a - 1$ in the calculation),

$$\langle\Delta\mathbf{v}\rangle_{M.C.} \equiv \left\langle \sum_{j=1}^{N_b} \Delta\mathbf{v}_j \right\rangle_{\hat{n}} = -\sum_{j=1}^{N_b} \frac{m_b}{m_a+m_b}\frac{\varepsilon^2 u_j^3}{3}\frac{\mathbf{u}_j}{\mathbf{u}_j^3} + O(\varepsilon^4), \tag{8}$$

$$\begin{aligned}\langle\Delta\mathbf{v}\Delta\mathbf{v}\rangle_{M.C.} &\equiv \left\langle \sum_{i=1}^{N_b} \Delta\mathbf{v}_i \sum_{j=1}^{N_b} \Delta\mathbf{v}_j \right\rangle_{\hat{n}} \\ &= \sum_{j=1}^{N_b} \langle\Delta\mathbf{v}_j\Delta\mathbf{v}_j\rangle_{\hat{n}} + O(\varepsilon^4) \\ &= \sum_{j=1}^{N_b}\left(\frac{m_b}{m_a+m_b}\right)^2 \frac{\varepsilon^2 u_j^3}{3}\frac{u_j^2\,\mathbf{I}-\mathbf{u}_j\mathbf{u}_j}{u_j^3} + O(\varepsilon^4),\end{aligned} \tag{9}$$

where

$$\langle\Delta\mathbf{v}_j\rangle_{\hat{n}} \equiv \frac{1}{4\pi}\int \Delta\mathbf{v}_j\,d\Omega, \quad \langle\Delta\mathbf{v}_j\Delta\mathbf{v}_j\rangle_{\hat{n}} \equiv \frac{1}{4\pi}\int \Delta\mathbf{v}_j\Delta\mathbf{v}_j\,d\Omega$$

(note that the probability of a random unit vector $\hat{n}$ being located in a solid angle $d\Omega$ is $d\Omega/4\pi$). We expect that the Monte Carlo operator can correctly reproduce the friction coefficient and the diffusion tensor. Compared to the analytical coefficients, it is found that the quantity $\varepsilon^2 u_j^3/3$ in Eqs. (8) and (9) should be set as

$$\frac{\varepsilon^2 u_j^3}{3} = \Delta t(4\pi e_a^2 e_b^2 \ln\Lambda)\left(\frac{1}{m_a}+\frac{1}{m_b}\right)^2 \frac{n_b}{N_b}. \tag{10}$$

Assuming that there are enough model particles to form a local background of correct velocity distribution, the summations can reproduce the average over the background distribution $f_b(\mathbf{v})$:

$$\sum_{j=1}^{N_b} \rightarrow \frac{N_b}{n_b}\int d\mathbf{v}'\, f_b(\mathbf{v}'). \tag{11}$$

Thus, as $\Delta t \rightarrow 0$, the Monte Carlo operator can give the friction coefficient and diffusion tensor,

$$\lim_{\Delta t\rightarrow 0}\frac{\langle\Delta\mathbf{v}\rangle_{\mathrm{M.C.}}}{\Delta t} = -\frac{4\pi e_a^2 e_b^2 \ln\Lambda}{m_a}\int d\mathbf{v}'\, f_b(\mathbf{v}')\,\frac{\mathbf{u}}{u^3}\left(\frac{1}{m_a}+\frac{1}{m_b}\right), \tag{12}$$

$$\lim_{\Delta t\rightarrow 0}\frac{\langle\Delta\mathbf{v}\Delta\mathbf{v}\rangle_{\mathrm{M.C.}}}{\Delta t} = \frac{4\pi e_a^2 e_b^2 \ln\Lambda}{m_a}\int d\mathbf{v}'\, f_b(\mathbf{v}')\,\frac{u^2\mathbf{I}-\mathbf{uu}}{u^3}\frac{1}{m_a}. \tag{13}$$

To make the operator more physically understandable, we give an insight into the relation between the random unit vector $\hat{n}$ and Θ and Φ. Set $\mathbf{u}$ along the $\hat{z}$ direction (Fig. 1c). In Cartesian coordinates the random unit vector can be expressed as

$$\hat{n} = \sin\vartheta\cos\varphi\hat{x} + \sin\vartheta\sin\varphi\hat{y} + \cos\vartheta\hat{z} \tag{14}$$

with $\cos\vartheta$ uniformly taking a random value between -1 and 1, and φ uniformly taking a random value between 0 and 2π. Then from the operator, we obtain

$$\cos\Theta = \frac{\mathbf{u}\cdot\mathbf{u}'}{u^2} = 1-(1-\cos\varepsilon)\sin^2\vartheta, \tag{15}$$

$$\tan\Phi = \frac{u_y'}{u_x'} = \tan(\varphi-\alpha) \tag{16}$$

with $\cot\alpha \equiv \cos\vartheta\tan(\varepsilon/2)$. From Eq. (15) we have $0 \leq \Theta \leq \varepsilon$. When $\varepsilon \ll 1$ the operator gives small-angle scattering which characterizes Coulomb collisions in a plasma.

Now we examine the distributions of Φ and Θ. From Eq. (16), it is readily found that Φ can take any value between 0 and 2π with equal probability. This point reflects the fact that $\mathbf{u}'$ (or $\Delta\mathbf{u}$) is isotropic in the direction perpendicular to $\mathbf{u}$. In a binary collision, the probability density for the scattering angle Θ is

$$p(\Theta,\varepsilon) = \begin{cases} \dfrac{1}{2\sqrt{1-\cos\varepsilon}}\dfrac{\sin\Theta}{\sqrt{\cos\Theta-\cos\varepsilon}}, & 0 \leq \Theta \leq \varepsilon, \\ 0, & \Theta > \varepsilon. \end{cases} \tag{17}$$

It depends on the step size Δt and the relative speed u via ε. Taking into account the distribution of relative speed, $f(u)$, we obtain the distribution of scattering angle Θ as

$$P(\Theta,\Delta t) = -\frac{d}{d\Theta}\int_0^{(3\Delta t\nu_{ab0}/\Theta^2)^{1/3}v_t} \frac{\sqrt{\cos\Theta-\cos\varepsilon}}{\sqrt{1-\cos\varepsilon}} f(u)\,du, \tag{18}$$

where v_t is the local thermal velocity and

$$\nu_{ab0} = \frac{4\pi e_a^2 e_b^2 \ln\Lambda n_L}{v_t^3}\left(\frac{1}{m_a}+\frac{1}{m_b}\right)^2$$

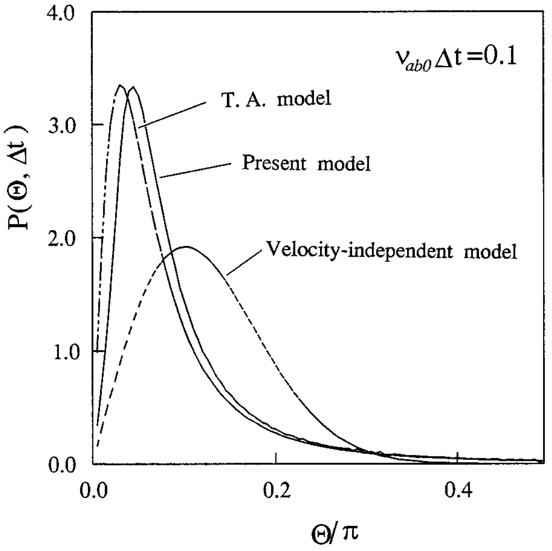

FIG. 2. Distribution of the scattering angle Θ with $\nu_{ab0}\Delta t = 0.1$.

with $n_L = \min(n_a, n_b)$. This distribution for $\nu_{ab0}\Delta t = 0.1$ is plotted in Fig. 2, where $f(u)$ takes the Maxwellian form

$$f(u) = \sqrt{\frac{2}{\pi}}\, v_t^{-3} u^2 \exp\left(-\frac{u^2}{2v_t^2}\right).$$

These distributions of the T.A. model [19], as well as the velocity-independent model [20], are also plotted in Fig. 2 for comparison. The probabilities of having a large angle deflection (≥60°) are compared in Table I. (Note that, without loss of generality, Algorithm 1 in Section 3 of the present model is employed in these comparisons for the sake of convenience.) It is seen that in ensuring small-angle collisions, the present model and the T.A. model have nearly the same performance. In the velocity-independent model [20], which is based on the T.A. model assuming a local thermal equilibrium, the large-angle deflections are reduced since the thermal velocity is used instead of the individual particle velocity in the Gaussian distribution given by Ref. [19].

3. FAST IMPLEMENTATION—ALGORITHMS SUITABLE FOR VECTOR CALCULATION

The Monte Carlo operator presented in the last section is simple in its form, and its implementation is straightforward and convenient. We here remark on one point about the implementation.

Coulomb collisions in a plasma take place simultaneously; i.e., one particle may simultaneously experience the forces exerted by a number of other particles around it, due to the character of Coulomb long-range interaction. In the Monte Carlo model, however, a particle velocity changes step by step in the time interval Δt as it collides with other particles one by one. The one-by-one collisions may affect the long time statistical properties of the system. Determining the collision order in an optimal way is necessary in the model. Obviously, determining the collision order in a completely random way may greatly improve the statistical properties. One simple way to determine collision order is shown in Fig. 3. First, all particles (electrons and ions) in a cell are randomly arranged in a line. The particle with index $I(1)$ collides with all other particles behind it first, then the particle with index $I(2)$ does, and so on. The influence of collision order on the long-time statistical error is examined in Section 4(3). Although the collision order determined by the above way is not completely random, examination shows that it is effective to give satisfactory results and, also, fewer calculations for sorting are needed this way.

Monte Carlo calculation of collisions is very time consuming. In the operator presented in Section 3, the number of binary collisions to be calculated is $N(N-1)/2$ at one step, where N is the total particle number in the cell. The computational time cost for a cell is approximately proportional to N^2. To reduce statistical errors, a large number of particles are needed in each cell. In addition, to apply it to a practical simulation involving space configuration, a lot of cells may be needed to reflect the spatial variety of the plasma. Thus, its fast implementation with high efficiency is essential for it to be practically applicable. Vector calculation provides a very effective way for the speedup. The implementation of the operator in each time interval Δt consists of three steps: generating uniform random unit vectors, arranging the collision order, and calculating the changes of particle velocities, among which the final step contains most of the floating-point calculations and is the most time-consuming. In each time step of interval Δt, one particle collides with all other particles in the same cell, and its velocity changes many times. The calculation of velocity changes according to the order given

TABLE I

Probability for Large-Angle Deflection ($\Theta \geq 60°$)

$\nu_{ab0}\Delta t =$	0.5	0.1	0.01	0.001
Present model	1.57×10^{-1}	3.86×10^{-2}	4.22×10^{-3}	4.30×10^{-4}
T.A. model	1.65×10^{-1}	4.21×10^{-2}	4.70×10^{-3}	4.82×10^{-4}

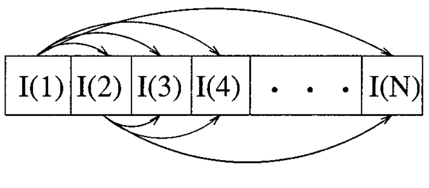

FIG. 3. Determination of collision order.

by Fig. 3 are not vectorized. Hereafter, the algorithm which implements the operator in Section 3 according to the collision order given by Fig. 3 is referred to as the scalar algorithm. To vectorize the calculations, one should arrange the collision pairs which are independent of each other so that these calculations can be processed as vectors. We next present two vectorizable algorithms.

Algorithm 1. Imagine that the time interval Δt is divided into many small time intervals Δt_i. At each small time interval Δt_i, particles in the same cell are paired in a random way for collisions. We expect that, in the time interval Δt, each particles may pair with all other particles in the same cell so as to reproduce the average over the background distribution. We next explain the details.

To pair particles for collisions, we use the same method as in the T.A. model [19]. At first, the particles of same species are randomly arranged in lines. Then, pairing particles is performed as follows. Like particles are paired as shown in Fig. 4. If the particle number is even, particles are paired in order from the top of the line (Fig. 4a). If the particle number is odd, the first three particles are paired in three pairs (Fig. 4b). Unlike particles are paired as shown in Fig. 5. When $N_e = N_i$, electrons and ions are paired in order from the top of the lines (Fig. 5a). If $N_e \neq N_i$, for example $N_e > N_i$ ($N_e/N_i = i + r$, where i is a positive integer and $0 \leq r < 1$), electrons and ions are divided into two groups, the first group with $(i + 1)rN_i$ electrons and rN_i ions and the second group with $i(1 - r)N_i$ electrons and $(1 - r)N_i$ ions. Each ion of the first group is selected $i + 1$ times to pair an electron of the first group, and each ion of the second group is selected i times to pair an electron of the second group (Fig. 5b).

Having paired the particles and generated $\hat{n}$ (the latter is easily vectorized), we calculate the velocity change after a collision using Eqs. (1) and (3). However, Eq. (2), used for giving the small parameter ε, is modified here to

$$\frac{\varepsilon^2 u^3}{3} = \Delta t_i (4\pi e_a^2 e_b^2 \ln \Lambda)\left(\frac{1}{m_a} + \frac{1}{m_b}\right)^2 \gamma, \tag{19}$$

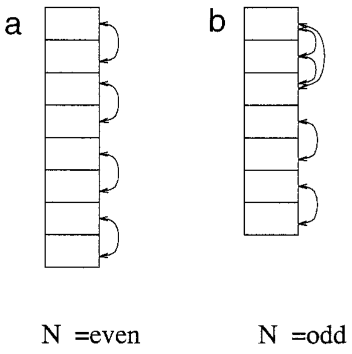

FIG. 4. After index randomization, like-particles are paired for binary collisions. Particle number: (a) N = even; (b) N = odd.

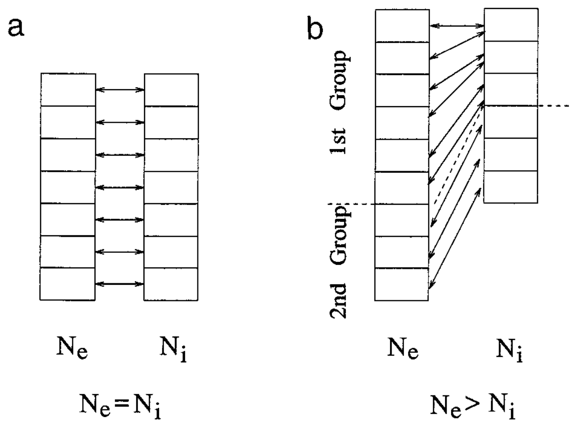

FIG. 5. Electrons and ions are paired for collisions: (a) $N_e = N_i$; (b) $N_e > N_i$.

with

$$\gamma = \begin{cases} n_a/2, & \text{for first three like particle collisions when } N_a \text{ is odd,} \\ n_L \equiv \min(n_a, n_b), & \text{otherwise.} \end{cases}$$

We can understand that the modified operator, as the operator presented in Section 3, is equivalent to the exact Fokker–Planck collision operator from the following two observations: (1) determining collision pairs by the above way means sampling $\mathbf{v}_b$ (or $\mathbf{v}_a$) in $\mathbf{u} \equiv \mathbf{v}_a - \mathbf{v}_b$ according to background distributions $f_b(\mathbf{v})$ (or $f_a(\mathbf{v})$); (2) in Fig. 5b, the probabilities of an electron (or an ion) being in the first group and the second group are, respectively, $(i + 1)rN_i/N_e = (i + 1)r/(i + r)$ (or $rN_i/N_i = r$) and $i(1 - r)N_i/N_e = i(1 - r)/(i + r)$ (or $(1 - r)N_i/N_i = 1 - r$). Saying exactly that the modified operator can reproduce the friction coefficient and diffusion tensor of the nonlinear Fokker–Planck operator with the accuracy of order $O(\Delta t)$.

From Fig. 4, it is seen that for $N = even$, all like-particle collisions are independent and can be processed in vector way; for $N = odd$, except the first three binary collisions which are calculated in scalar way, other collisions are independent and can be processed in the vector way. From Fig. 5, it is seen that: for $N_e = N_i$ all unlike-particle binary collisions are independent and can be processed in the vector way; for $N_e > N_i$, the binary collisions in the first group can be calculated by two vectorized DO-loops (in each DO-loop, the collisions are independent), and the binary collisions in the second group are independent and can be processes in the vector way. Therefore, with the method of pairlike collisions, the calculation of velocity changes can be vectorized now.

Algorithm 2. We note that Algorithm 1 is effective when $N_e = N_i$ = even, and, however, it is logically complicated for programming in other cases, especially when $N_e = odd \neq N_i =$ odd. In general, the complexity in logic may give rise to additional computational cost. In practice, the particle number in a cell is not fixed, and the cases without $N_e = N_i = even$ are often met. Algorithm 2, as next is seen is more practically applicable in the sense that it is universally effective to handle the general case.

For the convenience of presentation, an example with seven electrons and six ions is employed here. At first, as in Algorithm 1, electrons (and ions) in a cell are randomly arranged in a line (Fig. 6a). In a step of time interval Δt, for unlike-particle collisions, let one electron collide with all ions in the same cell. The ergodic unlike-particle collision pairs are formed by the method of ''round robin,'' as shown in Fig. 6b (where the final electron in each round draws a bye). For like-particle collisions, we divide the electrons (and the ions) into two groups, group A with $N_{eA} = \mathrm{int}(N_e/2)$ (= 3 for our example) electrons and group B with $N_{eB} = \mathrm{int}[(N_a + 1)/2]$ (= 4 for our example) electrons, as shown in Fig. 6a, and we let one particle in a group collides with all particles in another group. The like-particle collision pairs are then formed between the two groups in the same way as the unlike-particle collisions (Fig. 6c). Now, the binary collisions in each round shown in Fig. 6 are ready for vector calculation.

The velocity change after a collision is calculated by Eq. (1), Eq. (2), and Eq. (3) in which, for like-particle collisions of species a, the factor γ is modified as

$$\gamma = \frac{n_a}{2(N_{cn}/N_a)},$$

where $N_{cn} \equiv N_{aA}N_{aB}$ is the like-particle collision number of species a in the time interval Δt.

Note that particles that belong to the same group do not collide with each other in a time step. If necessary, we can immediately improve this point by further dividing a group into two subgroups. The particle collisions between two subgroups are performed in the same way as shown in Fig. 6c without additional difficulty. Correspondingly, the factor γ given above changes as the like-particle collision number N_{cn} changes.

The algorithm presented here is highly vectorizable thanks to the way that particles are grouped for collisions. This algorithm, as the scalar algorithm, and Algorithm 1 give the friction coefficient and diffusion tensor of the nonlinear Fokker–Planck operator with the accuracy of order $O(\Delta t)$.

4. TEST

The Monte Carlo operator presented here should be tested carefully in order to be applied. For this purpose, we have simulated various problems in the velocity space concerning relaxation processes, deviation from Maxwellian distribution, and electrical conductivity. For simplicity, a simple plasma system with $n_e = n_i$, thus $N_e = N_i$, and charge number $Z = 1$ is employed in the simulations.

1. *Relaxation between Electron Temperature T_e and Ion Temperature T_i*

In the simulation, initially set both electrons and ions to be Maxwellian but with different temperatures $T_{e0} \neq T_{i0}$. Due to Coulomb collisions, electrons and ions will relax to an equilibrium, and T_e and T_i will approach same temperature. This process is described by

$$\frac{dT_e}{dt} = \frac{T_i - T_e}{\tau_T^{e/i}}, \tag{20}$$

$$n_e T_e + n_i T_i = n_e T_{e0} + n_i T_{i0}, \tag{21}$$

where

$$\tau_T^{e/i} = \frac{1}{2\sqrt{2}} \left(\frac{T_e}{T_{e0}} + \frac{m_e}{m_i} \frac{T_i}{T_{e0}} \right)^{3/2} \tau_0$$

with the relaxation time defined as

$$\tau_0 = \frac{m_i}{m_e} \frac{3\sqrt{m_e}}{4\sqrt{\pi} e_e^2 e_i^2 n_i \ln \Lambda} T_{e0}^{3/2}.$$

The result of the simulation by using the scalar algorithm with $N_e = N_i = 400$ and $T_{e0}/T_{i0} = 2$ is shown in Fig. 7, in which the Monte Carlo operator gives good agreement with theory.

2. *Relaxation between Longitudinal Temperature $T_\parallel$ and Transverse Temperature $T_\perp$*

Now we consider the thermal isotropization process in a single species particle system in which the particles are initially in the distribution

$$f_{a0} = n_a \left(\frac{m_a}{2\pi T_{\parallel 0}} \right)^{1/2} \left(\frac{m_a}{2\pi T_{\perp 0}} \right) \exp\left(-\frac{m_a}{2T_{\parallel 0}} v_{\parallel 0}^2 - \frac{m_a}{2T_{\perp 0}} v_{\perp 0}^2 \right)$$

with $T_{\parallel 0} \neq T_{\perp 0}$. In theory, the equations here can describe the relaxtion between $T_\parallel$ and $T_\perp$

$$\frac{dT_\perp}{dt} = -\frac{1}{2} \frac{dT_\parallel}{dt} = -\frac{T_\perp - T_\parallel}{\tau_T^a}, \tag{22}$$

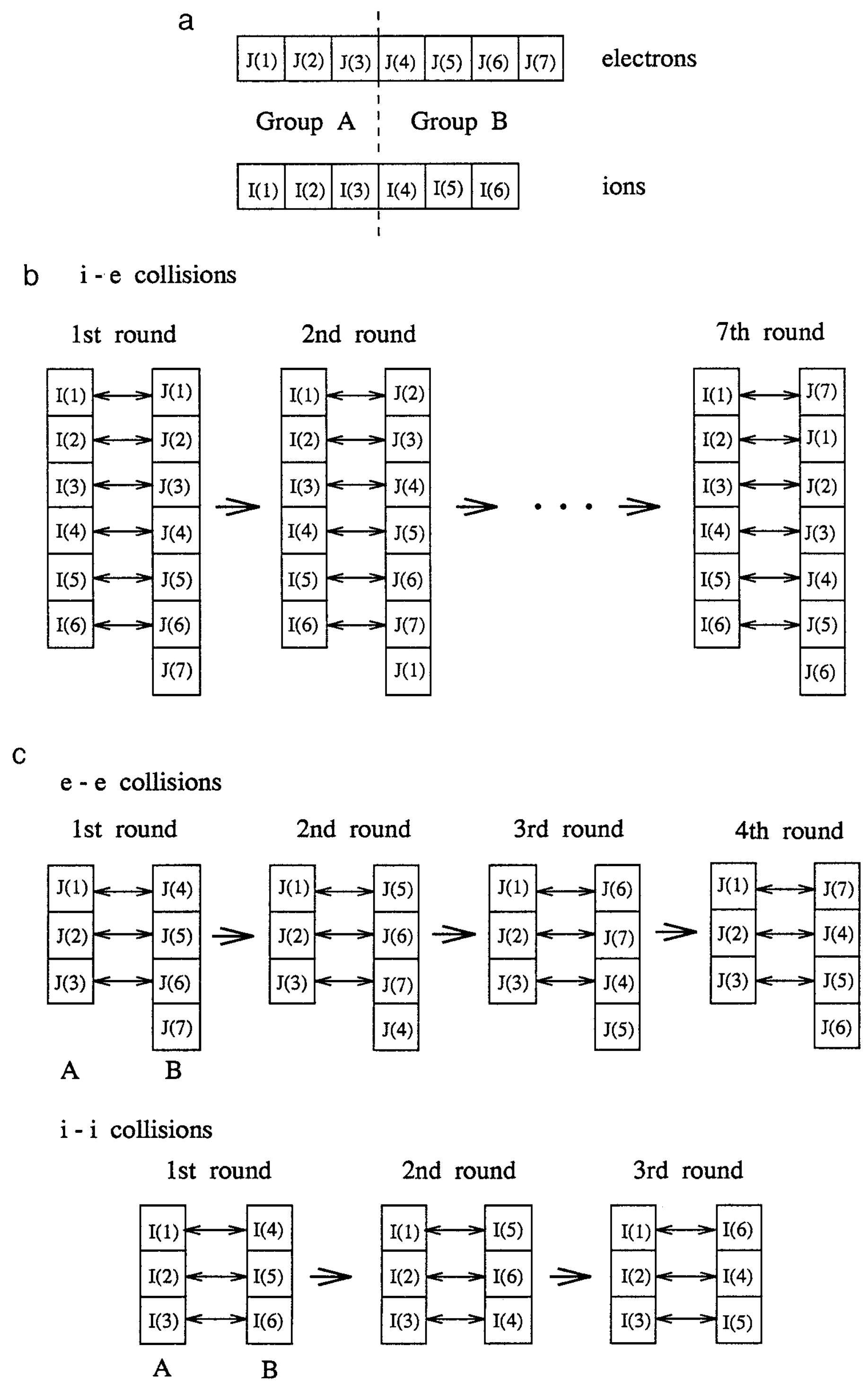

FIG. 6. Particles are grouped for collisions (an example of 7 electrons and 6 ions): (a) electrons (and ions) are randomly arranged in a line and divided into Groups A and B; (b) i-e collisions are performed by the way of "round robin," where in each round the final electron draws a bye; (c) like-particle collisions between Groups A and B are performed by the same way.

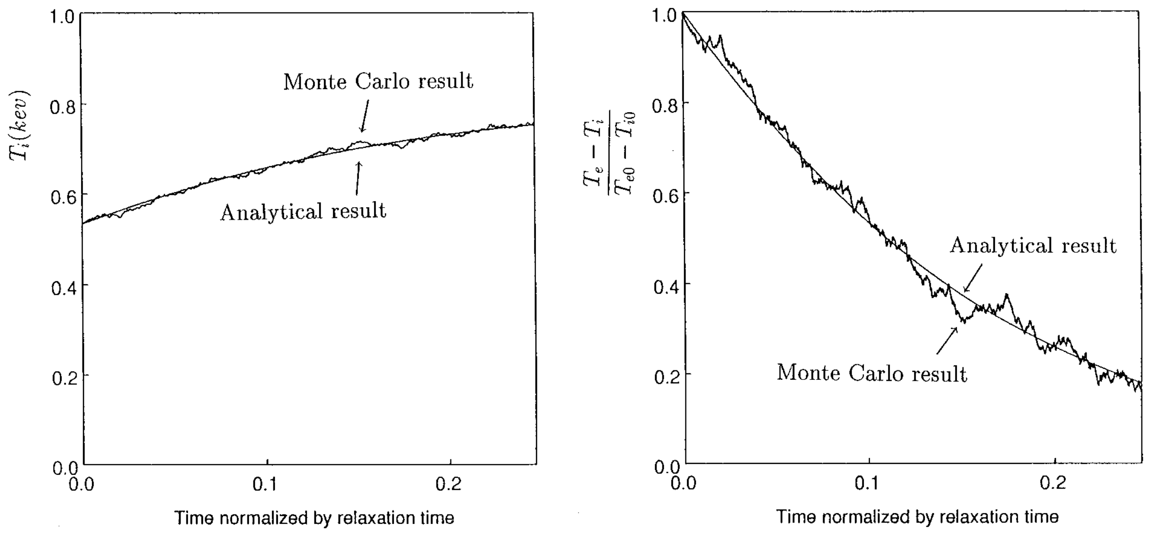

FIG. 7. Temporal relaxation between electron temperature T_e and ion temperature T_i (by the scalar algorithm).

where

$$\tau_T^a = \frac{2}{3}\left(\frac{T_\parallel}{T_{a0}}\right)^{3/2} A^2 \tau_a$$

$$\begin{cases} \left[-3 + (A+3)\dfrac{\tan^{-1}\sqrt{A}}{\sqrt{A}}\right]^{-1}, & A > 1, \\ \left[-3 + (A+3)\dfrac{\tanh^{-1}\sqrt{-A}}{\sqrt{-A}}\right]^{-1}, & A < 1, \end{cases}$$

with the relaxation time here defined as

$$\tau_a = \frac{3\sqrt{m_a}}{4\sqrt{\pi}e_a^4 n_a \ln\Lambda} T_{a0}^{3/2},$$

$T_{a0} = (2T_{\perp 0} + T_{\parallel 0})/3$, and $A = T_\perp/T_\parallel - 1$.

This process is modeled by the Monte Carlo operator. The time evolutions of difference between $T_\parallel$ and $T_\perp$ are illustrated in Fig. 8 with $N_e = 2000$, $T_{\parallel 0} = 1.5$ keV, and $T_{\perp 0} = 2.5$ keV. Here the scalar algorithm is used. The good agreement between the Monte Carlo results and the analytical result is obtained under different time step sizes (Δt in Fig. 8b is 4 times that in Fig. 8a).

3. *Deviation from Equilibrium*

To examine the influence of collision order on the long-time statistical error, we apply the scalar algorithm to a model plasma system (500 electrons + 500 ions) which is initially in equilibrium, i.e., both electrons and ions are Maxwellian and have the same temperature. Figure 9 show the time evolution of the ion temperature. The nonrandomness collisions give rise to the deviation of the ion (and the electron) temperature from the equilibrium value. Randomization of collision order by the method shown in Fig. 3 can effectively reduce this error.

As a test of the accuracy of the operator, a system with 1000 initially Maxwellian particles is followed in time to observe the deviation from the isotropic distribution. The observation is shown in Fig. 10, from which we can conclude two points: the fluctuations of three temperature components T_x, T_y, and T_z are within 5%, and the deviation from the isotropic distribution does not increase with time.

4. *Electrical Conductivity*

The next test is related to the parallel electrical conductivity. Initially set the electrons and ions to be in equilibrium. A magnetic field and an electric field are introduced along the $\hat{z}$ direction. Due to different electrical accelerations, the relative motion between electrons as a whole and ions as a whole occurs in the $\hat{z}$ direction and, as a result, a current $j_\parallel$ along the $\hat{z}$ direction is created. Meanwhile, due to ohmic heating the electron temperature increases. In this case, since the system is not in the steady state, the simple Ohm's law $j_\parallel = \sigma_\parallel E$ is not valid, where $\sigma_\parallel$ denotes parallel electrical conductivity. Instead, the following equations describe the problems:

$$\frac{3}{2} n_e \frac{dT_e}{dt} = \frac{j_\parallel^2}{\sigma_\parallel} - 3\sqrt{2} n_e \left(\frac{T_{e0}}{T_e}\right)^{3/2} \frac{T_e - T_i}{\tau_0}, \qquad (23)$$

$$\frac{3}{2} n_i \frac{dT_i}{dt} = 3\sqrt{2} n_e \left(\frac{T_{e0}}{T_e}\right)^{3/2} \frac{T_e - T_i}{\tau_0}, \qquad (24)$$

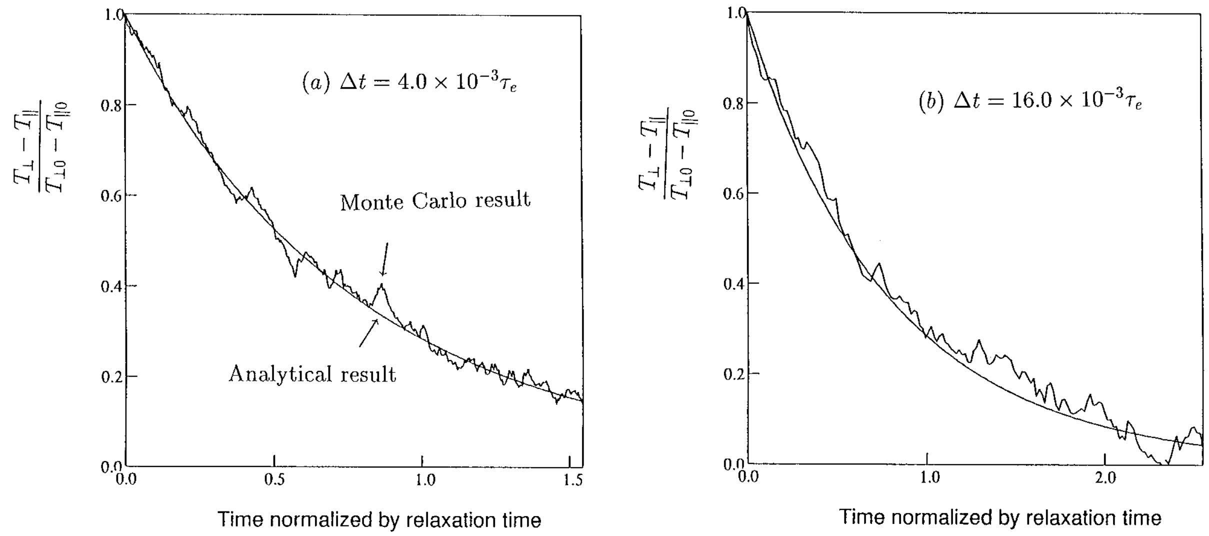

FIG. 8. Temporal relaxation between longitudinal temperature $T_\parallel$ and perpendicular temperature $T_\perp$ (by the scalar algorithm): (a) $\Delta t = 4.0 \times 10^{-3}\tau_e$; (b) $\Delta t = 16.0 \times 10^{-3}\tau_e$.

$$\frac{dV_e}{dt} = -\frac{e}{m_e}E + \frac{e}{m_e}\frac{j_\parallel}{\sigma_\parallel}, \tag{25}$$

$$\frac{dV_i}{dt} = \frac{e}{m_i}E - \frac{e}{m_i}\frac{n_e}{n_i}\frac{j_\parallel}{\sigma_\parallel}, \tag{26}$$

$$j_\parallel = -en_e(V_e - V_i). \tag{27}$$

We solve Eqs. (23)–(27) by the Runge–Kutta method to obtain the analytical result. When we solve the equations, the theoretical parallel electrical conductivity

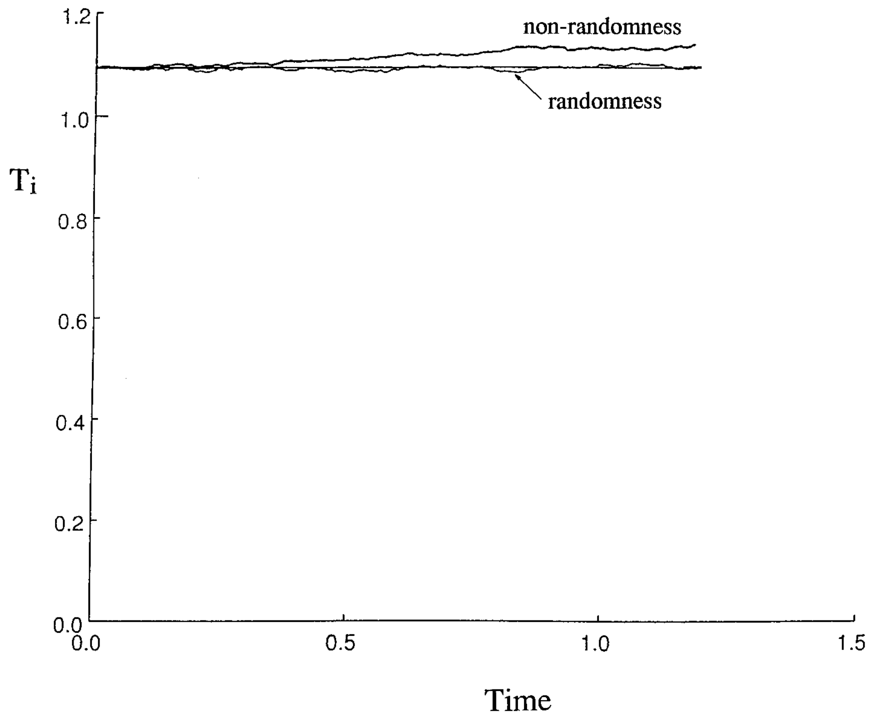

FIG. 9. Deviations of ion temperature from the equilibrium value with randomness and nonrandomness in collision order.

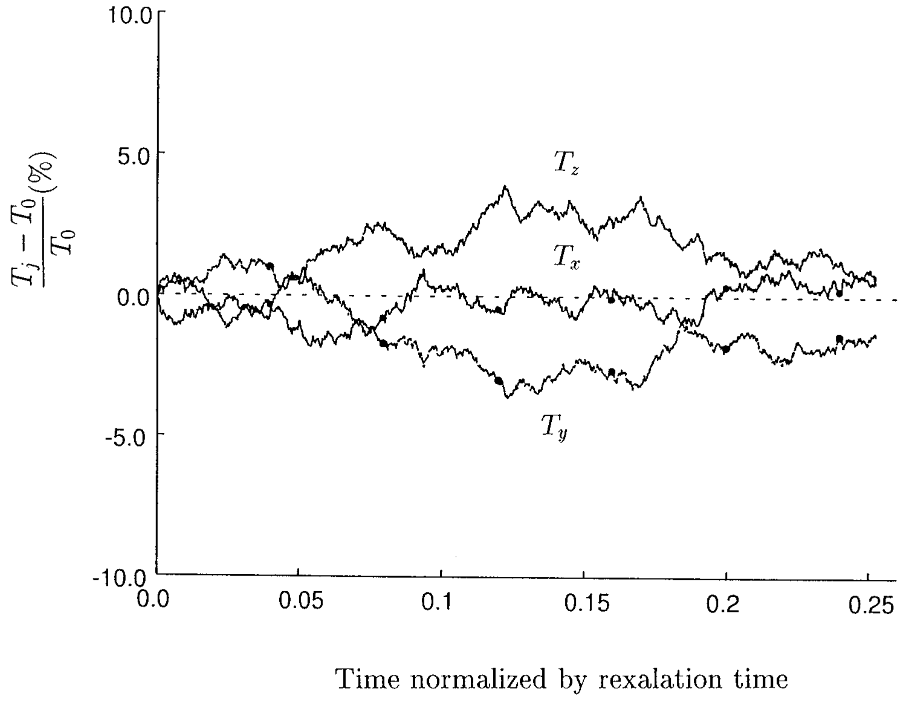

FIG. 10. Deviation from isotropic distribution.

$$\sigma_\parallel = 1.96 \frac{3T_e^{3/2}}{4\sqrt{2\pi}\ln \Lambda e^2 \sqrt{m_e}} \tag{28}$$

is used.

The Monte Carlo result obtained by Algorithm 1 with $N_e = N_i = 1000$, $B = 0.01$ Tesla and $E = 0.2E_d$ (E_d denotes the Dreicer field), as well as the analytical result, are shown in Fig. 11, in which T_e and $j_\parallel$ are plotted versus t/τ_e. We can see that the Monte Carlo result agrees well with the theory. In addition, a series of simulations with different parameters has been carried out and it shows that the Monte Carlo operator can correctly give the dependences

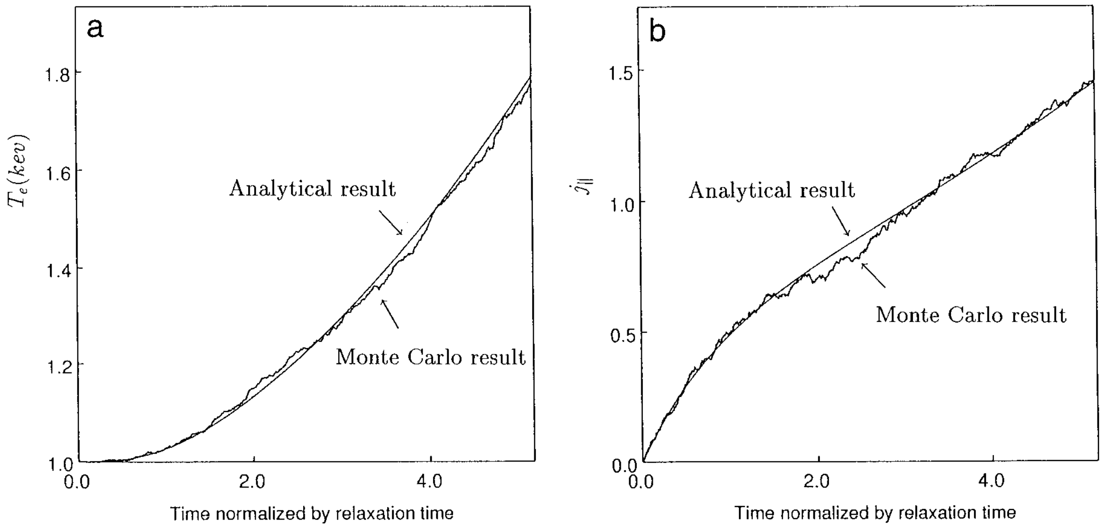

FIG. 11. The results of a test regarding parallel electrical conductivity (by Algorithm 1). Time evolution of (a) electron temperature T_e due to Ohmic heating and (b) parallel current $j_\parallel$.

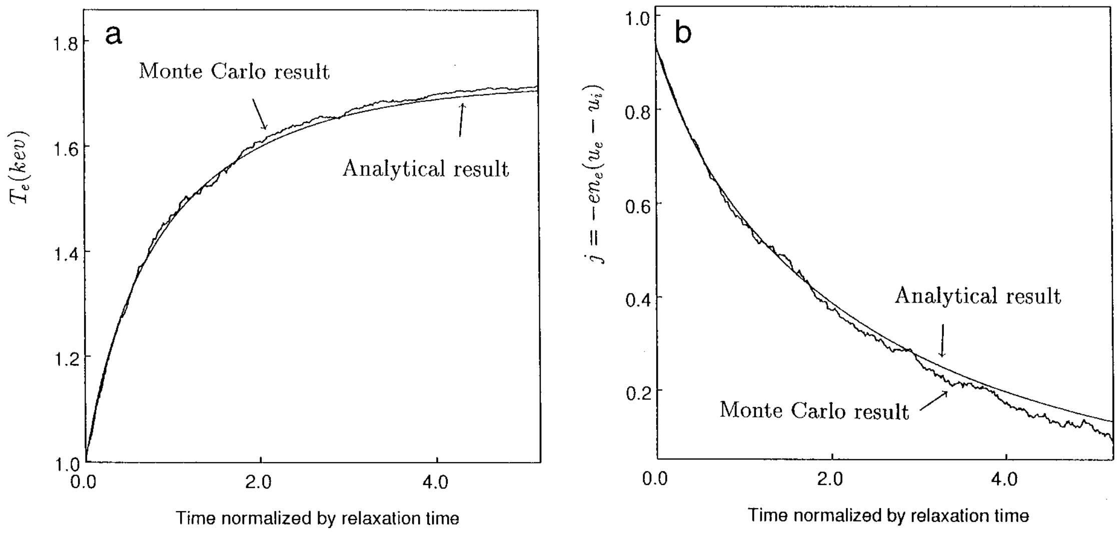

FIG. 12. Relaxation of initially shifted Maxwellian distribution (by Algorithm 1): (a) electron temperature T_e versus t/τ_e; (b) current j versus t/τ_e.

of $\sigma_\|$ expressed in Eq. (28); i.e., the numerical factor is about 1.96 and $\sigma_\|$ is proportional to $T_e^{3/2}$ and is independent of B, E, and plasma densities.

5. *Relaxation of Initially Shifted Maxwellian Distribution*

We again use a system of 1000 electrons and 1000 ions, and employ Algorithm 1. Initially, the electrons are in a shifted Maxwellian distribution,

$$f_{e0} = n_e\left(\frac{m_e}{2\pi T_{e0}}\right)^{3/2} \exp\left[-\frac{m_e}{2T_{e0}}(\mathbf{v}-\mathbf{V}_{e0})^2\right],$$

and the ions are Maxwellian with $T_{i0} = T_{e0}$. In the first stage, collisional dissipation transfers the electron kinetic energy to thermal energy and leads the electrons to reach equilibrium. The temperature T_e and current j evolutions given by the Monte Carlo simulation with $\frac{1}{2}m_e V_{e0}^2/T_{e0} = 1.09$ are shown in Fig. 12. For comparison, the analytical results which are obtained by Eqs. (23)–(27) with electric field $E = 0$ are also shown.

6. *Timing*

The three schemes (scalar algorithm, Algorithm 1, and Algorithm 2) are implemented and all can give the correct results for above tests. The timing results on an SX-3 supercomputer (main memory: 4 gigabytes; calculation speed: 6.4 gigaflops) are tabulated in Table II, where the CPU time for calculating velocity changes per collision (first column) and the total CPU time per collision (second column) are listed. It is found that, in the calculation of velocity changes, about 20 times speedup of the vector implementation (by Algorithm 1 and Algorithm 2) over the scalar implementation (by the scalar algorithm) is obtained. It should be mentioned that the CPU time per collision is almost particle-number independent and that the speedup is computer dependent. On a VXP computer (main memory: 128 Mbytes; calculation speed: 285 Mflops) the vector implementation using Algorithm 1 and Algorithm 2 is four to five times faster than the scalar implementation using the scalar algorithm. For comparison, the Monte Carlo results obtained by Algorithm 2 are illustrated in Fig. 13, which are the equilibration of T_e and T_i, and the thermal isotropization between $T_\|$ and $T_\perp$.

The scalar implementation efficiency of the present operator is compared with that of the T.A. model [19] on a scalar computer UXP. The total CPU time per collision of the present operator is 7.17 μs which is slightly smaller than that of the T.A. model, 7.58 μs. In other words, the amount of floating-point calculations per collision of the two models are nearly the same.

5. SUMMARY AND DISCUSSION

A nonlinear Monte Carlo collision model based on Coulomb binary collisions has been developed with the emphasis on its fast implementation in a vector computer.

In the model, the momentum conservation and energy conservation are satisfied since the binary collisions are elastic. Because the binary collision pairs are formed from a spatial cell, the conservations are quasi-local, i.e., within some spatial cell. The presented Monte Carlo operator is equivalent to the nonlinear Fokker–Planck operator because the friction coefficient and diffusion tensor can be

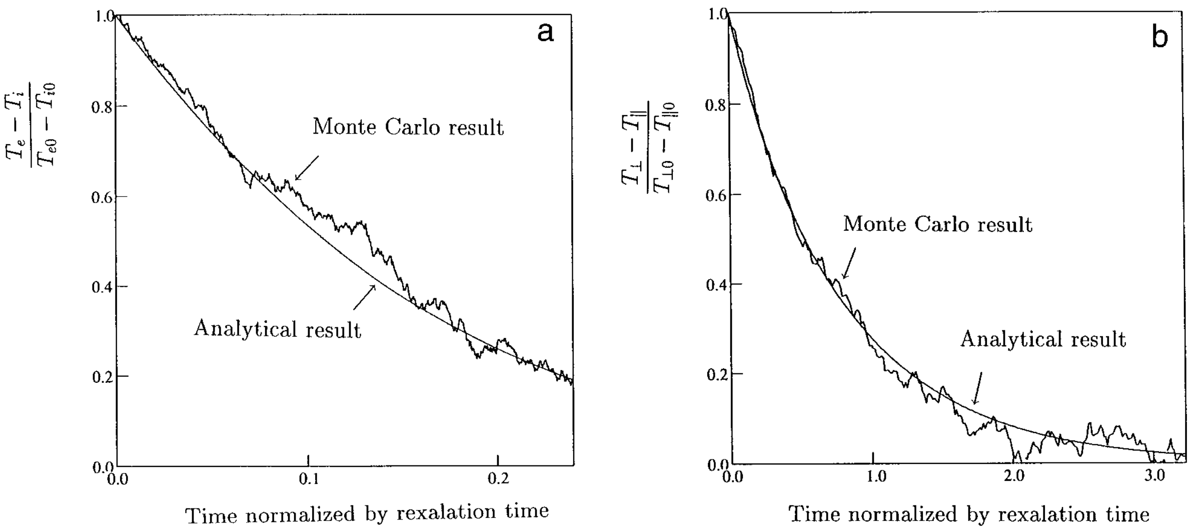

FIG. 13. Thermal relaxation processes simulated by using Algorithm 2: (a) temporal relaxation between T_e and T_i with $T_e/T_i = 2$ and $N_e = N_i = 513$; (b) Temporal relaxation between $T_\parallel$ and $T_\perp$ with $T_\perp/T_\parallel = 2$ and $N_e = 2111$.

correctly reproduced in the limit of infinitesimally small Δt. The small-angle scattering characteristic is assured by the small parameter ε. All the above properties can be compared to those of the earlier binary collision model [19].

The implementation of the operator is straightforward and covenient due to its simple form. The efficiency of the scalar implementation is on the same level as that of the earlier model [19], i.e., the CPU times per collision of the two models are nearly the same. For the practical application, two vectorizable algorithms have been designed. A high efficiency for speedup has been achieved on the vector computers by the two algorithms. On the VPX vector computer the speedup is four to five times, and on the SX-3 supercomputer it is about 17 times. The CPU time per collision of vector implementation on the SX-3 is about 0.2 μs.

The relaxation processes, electrical conductivity, and Ohmic heating in the uniform plasma have been simulated as the test of the model. The Monte Carlo results show good agreement with the theories.

Nonlinear collision calculations are necessary for some accurate simulation. The nonlinear operator constructed in the general case may be widely useful for problems in both magnetized and unmagnetized plasmas because it makes fewer assumptions. The fast implementation can increase its applicability. In principle, it is potentially useful for general nonlinear problems involving the evolution of distribution functions far from equilibrium where the linear operators cannot be applied. Such an important example is the electron transport process in laser produced plasma.

TABLE II

CPU Times per Collision on SX-3

Scheme	CPU for calculating velocity change	Total CPU
Scalar algorithm	3.50	3.55
Algorithm 1[a]	0.15	0.20
Algorithm 1[b]	0.18	0.24
Algorithm 2	0.15	0.20

Note. All times in μs.
[a] $N_e = N_i = even.$
[b] $N_e = odd \neq N_i = odd.$

ACKNOWLEDGMENTS

The authors are grateful to Dr. T. Takizuka for his fruitful comment. They thank Dr. R. Kanno for useful discussions during the preparation of the paper. One of the authors (Wang) wishes to acknowledge the hospitality of the Theory and Data Analysis Division of the National Institute for Fusion Science. He thanks Professor Tieqiang Chang for his continuous encouragement. He is supported by the Japanese Ministry of Education, Science, and Culture (MONBUSHO).

REFERENCES

1. A. H. Boozer and G. Kuo-Petravic, *Phys. Fluids* **24,** 851 (1981).
2. R. H. Fowler, J. A. Rome, and J. F. Lyon, *Phys. Fuids* **28,** 338 (1985).
3. W. Lotz and J. Nührenberg, *Phys. Fluids* **31,** 2984 (1988).
4. S. Murakami, M. Okamoto, N. Nakajima, M. Ohnish, and H. Okada, *Nucl. Fusion* **34,** 913 (1994).
5. M. A. Kovanen, W. G. F. Core, and T. Hellsten, *Nucl. Fusion* **32,** 787 (1992).
6. L.-G. Eriksson and P. Helander, *Phys. Plasmas* **1,** 308 (1994).
7. W. W. Lee, *J. Comput. Phys.* **72,** 243 (1987).

8. T. Tajima, *Computational Plasma Physics* (Addison–Wesley, Redwood City, CA, 1989).

9. A. M. Dimits and W. W. Lee, *J. Comput. Phys.* **107,** 309 (1993).

10. R. Shanny, J. M. Dawson, and J. H. Greene, *Phys. Fluids* **10,** 1281 (1967).

11. R. B. White, A. H. Boozer, R. Goldston, H. Hay, J. Albert, and C. F.F. Karey, in *Plasma Physics and Controlled Nuclear Fusion Research* (International Atomic Energy Agency, Vienna, 1982), Vol. 3, p. 391.

12. P. J. Catto and K. T. Tsang, *Phys. Fluids B* **20,** 396 (1991).

13. X. Q. Xu and M. N. Rosenbluth, *Phys. Fluids B* **3,** 627 (1991).

14. A. M. Dimits and B. I. Cohen, *Phys. Rev. E* **49,** 709 (1994).

15. M. Kotchenreuther, *Bull. Am. Phys. Soc.* **34,** 2107 (1988).

16. G. DiPeso, E. C. Morse, and R. W. Ziolkowki, *J. Comput. Phys.* **96,** 325 (1991).

17. M. Tessarotto, R. B. White, and L. J. Zheng, *Phys. Plasmas* **1,** 951 (1994).

18. M. Tessarotto, R. B. White, and L. J. Zheng, *Phys. Plasmas* **1,** 2591 (1994).

19. T. Takizuka and H. Abe, *J. Comput. Phys.* **25,** 205 (1977).

20. S. Ma, R. D. Sydora, and J. M. Dawson, *Comput. Phys. Commun.* **77,** 190 (1993).