Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 223 (2007) 436–450
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

An efficient explicit numerical scheme for diffusion-type equations with a highly inhomogeneous and highly anisotropic diffusion tensor

O. Larroche *

CEA/DIF, BP 12, 91680 Bruyères le Châtel, France

Received 10 November 2005; received in revised form 18 August 2006; accepted 19 September 2006
Available online 2 November 2006

Abstract

A locally split-step explicit (LSSE) algorithm was developed for efficiently solving a multi-dimensional advection-diffusion type equation involving a highly inhomogeneous and highly anisotropic diffusion tensor, which makes the problem very ill-conditioned for standard implicit methods involving the iterative solution of large linear systems. The need for such an optimized algorithm arises, in particular, in the frame of thermonuclear fusion applications, for the purpose of simulating fast charged-particle slowing-down with an ion Fokker–Planck code. The LSSE algorithm is presented in this paper along with the results of a model slowing-down problem to which it has been applied.

Keywords: Advection-diffusion; Explicit schemes; Fokker–Planck equation; Thermonuclear fusion

1. Introduction

In preparation for the numerical simulation of ignition and thermonuclear burn in inertial confinement fusion targets [1] with our ion Fokker–Planck code "FPion" [2,3], efficient methods are investigated for the numerical treatment of collisions of a distribution of hot α particles in the presence of the much colder bulk DT plasma. The Fokker–Planck operator for Coulomb collisions is an advection-diffusion operator in particle velocity space [4]. In the present case, the slowing-down and diffusion terms in velocity space induced by the cold plasma are both very localized and highly anisotropic, which makes implicit schemes very ill-conditioned and thus not easily amenable to iterative solving methods such as conjugate-gradient [5,6] or the like. On the other hand, the simpler Jacobi iteration method [7] can be shown to be equivalent to explicit time-stepping, and thus impractical when locally high values of the diffusion tensor constrain the time step to very small values. The high anisotropy also makes the use of the same alternating-direction implicit (ADI) scheme [8] that we use for the bulk plasma (see Refs. [2,3]) questionable in the case of fast-particle slowing-down. Let us make

* Tel.: +33 1 69264879.
E-mail address: olivier.larroche@cea.fr

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.09.016

it clear that the anisotropy of the diffusion tensor constrains the choice of a numerical algorithm even when the distribution function itself happens to be isotropic, which is the case, for example, at the center of the hot spot in a spherically symmetric ICF pellet implosion. Due to strong spatial gradients in some parts of an ICF target at stagnation (e.g., the boundary between the central hot spot and the colder and denser outer fuel layer), the α-particle distribution function will become anisotropic in velocity space in those regions of configuration space.

We thus developed a locally split-step explicit (LSSE) algorithm for efficiently solving a multi-dimensional diffusion equation involving a highly-inhomogeneous diffusion tensor **K**. The principle of this algorithm is to perform time-step splitting only in cells where it is actually needed due to locally high values of the diffusion tensor, so as to satisfy the time-step condition for stability

$$\frac{\delta t}{\delta v^2}\mathrm{Tr}(\mathbf{K}) \leqslant \frac{1}{2}$$

where δt is the time step and δv is the mesh size, on a per-cell basis. The computational cost of this scheme (number of operations N_{op} required for advancing the system over a time Δt, using an overall time step δt) in the case of Coulomb-type diffusion ($|\mathbf{K}|(v) \sim 1/v$) will be shown to be the sum of two parts:

$$N_{\mathrm{op}}(\Delta t, \delta t, N, v_{\max}) = \mathrm{O}\left(\frac{\Delta t}{\delta t} N \log N\right) + \mathrm{O}\left(\frac{\Delta t}{\tau_{\mathrm{c}}(v_{\max})} N^{\frac{d+2}{d}}\right)$$

where N is the number of cells in a discretized velocity space of dimension d, $v_{\max}$ is the largest value of the discretized velocity and $\tau_{\mathrm{c}}(v)$ is the Coulomb slowing-down time for a particle with velocity v. Put in other words, the LSSE scheme, when used to advance the simulation of a thermonuclear fusion plasma over the α particle slowing-down time, is expected to need about as many operations as a *single* step of an implicit scheme.

A small numerical model of an advection-diffusion equation with parameters accounting for the effect of Coulomb ion–ion collisions on the velocity distribution has been implemented, and results from this model using the LSSE algorithm will be shown in this paper, for demonstrating the potential benefits of that algorithm. However, the LSSE algorithm is still not implemented in our Fokker–Planck code FPion.

In the case of a more general diffusion tensor for other physical applications of the diffusion problem, no general formula for the computational cost can be given unless an analytical expression of the field dependence of the tensor is known, but the principle of the algorithm ensures that the largest possible time-step is used in each mesh, thus minimizing the overall computational cost for any tensor dependence. Let us emphasize that the method presented in this paper aims at optimizing the time integration of the diffusion equation (or any conservation equation indeed), irrespective of the details of the space (or, in the present case, velocity space) dependences of the specific problem under study. For information about the velocity space aspect of the numerical solution of the Fokker–Planck equation, the reader is referred to references [9,10], and references therein.

2. Collisional relaxation of alpha particles from fusion reactions

The fusion reactions in a thermonuclear deuterium–tritium (DT) plasma generate 3.5 MeV α particles which subsequently slow down through Coulomb collisions with the electrons and ions in the plasma. In the case of inertial confinement fusion [1] at the time of ignition, the latter is in the form of a "hot spot" ($k_{\mathrm{B}}T \approx 7$ keV) surrounded by a dense shell of colder ($k_{\mathrm{B}}T \approx 0.7$ keV) plasma. The typical collision mean-free-path of the energetic α particles is comparable with the size of the hot spot, so that an accurate treatment of the thermalization process requires a kinetic description. In that formalism, the effect of Coulomb collisions on the α particle velocity distribution function f_α is described by a Fokker–Planck equation in velocity space [11], which is essentially an advection-diffusion equation (summation over repeated indexes is assumed throughout the text):

$$\frac{\partial f_\alpha}{\partial t} + \frac{\partial}{\partial v_i}\left(u_i f_\alpha - K_{ij}\frac{\partial f_\alpha}{\partial v_j}\right) = 0 \tag{1}$$

where

$$u_i = -4\pi\Gamma_{\alpha\beta}\frac{m_\alpha}{m_\beta}\frac{\partial S_\beta}{\partial v_i}$$

is an advection "velocity" (in velocity space) and

$$K_{ij} = -4\pi\Gamma_{\alpha\beta}\frac{\partial^2 T_\beta}{\partial v_j \partial v_i}$$

is a velocity space diffusion tensor. These quantities are computed from the so-called "Rosenbluth potentials" [4] S_β and T_β which are defined by the equations

$$\frac{\partial^2 S_\beta}{\partial v_i \partial v_i} = f_\beta \quad \text{and} \quad \frac{\partial^2 T_\beta}{\partial v_i \partial v_i} = S_\beta$$

where f_β designates the velocity distribution function of target particles and $\Gamma_{\alpha\beta}$ is the following expression:

$$\Gamma_{\alpha\beta} = \frac{4\pi e^4 Z_\alpha^2 Z_\beta^2}{m_\alpha^2}\log\Lambda_{\alpha\beta}$$

where m_s and $Z_s e$ are the mass and electric charge, respectively, of particles of species s and $\log\Lambda_{\alpha\beta}$ is the so-called "Coulomb logarithm" (see [12]). When the distribution function of the target particles f_β is isotropic, the diffusion tensor reads

$$K_{ij} = -4\pi\Gamma_{\alpha\beta}\left(\left(\delta_{ij} - \frac{v_i v_j}{v^2}\right)\frac{1}{v}\frac{\mathrm{d}T_\beta}{\mathrm{d}v} + \frac{v_i v_j}{v^2}\frac{\mathrm{d}^2 T_\beta}{\mathrm{d}v^2}\right)$$

where $v = (v_i v_i)^{1/2}$. The Rosenbluth potentials are plotted on Fig. 1 in the case of a thermal equilibrium (Maxwellian) distribution function.

2.1. Fast particle slowing-down by the cold bulk plasma

On the fast-particle velocity range, the target particle distribution function is highly localized, so that the associated Rosenbluth potentials are close to their vanishing-temperature form (see Fig. 1):

$$f_\beta \approx n_\beta \delta(\vec{v}) \Rightarrow S_\beta(v) \approx -\frac{n_\beta}{4\pi v} \quad \text{and} \quad T_\beta(v) \approx -\frac{n_\beta}{8\pi}v$$

so that

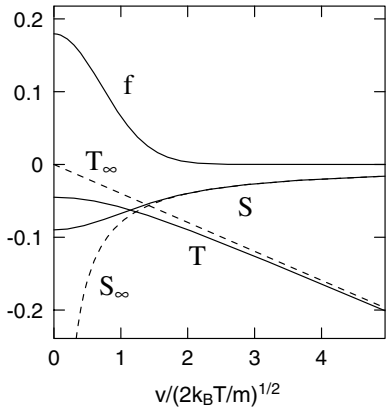

Fig. 1. The Rosenbluth potentials S and T are plotted as functions of the velocity v scaled to the thermal velocity $(k_B T/m)^{1/2}$ in the case of a Maxwellian distribution function f. Curves labelled S_∞ and T_∞ are the vanishing-temperature limits of S and T, respectively, however plotted on the same finite-temperature velocity scale for easier comparison.

$$u_i = -4\pi\Gamma_{\alpha\beta}\frac{m_\alpha}{m_\beta}\frac{\partial S_\beta}{\partial v_i} \approx -\Gamma_{\alpha\beta}n_\beta\frac{m_\alpha}{m_\beta}\frac{v_i}{v^3}$$

The slowing-down advection velocity is often written in the form

$$u_i = -\frac{v_i}{\tau_c(v)}$$

where

$$\tau_c(v) = \frac{m_\beta v^3}{\Gamma_{\alpha\beta}n_\beta m_\alpha} \tag{2}$$

is the characteristic slowing-down time of particles of species α due to Coulomb collisions with particles of species β (assumed at rest). The slowing-down velocity is approximately divergence-free in velocity space, except inside the bulk of the cold target-particle distribution, so that the advective part of the Fokker–Planck equation takes on the form:

$$\frac{\partial f_\alpha}{\partial t} + u_i\frac{\partial f_\alpha}{\partial v_i} = -f_\alpha\frac{\partial u_i}{\partial v_i} = 4\pi\Gamma_{\alpha\beta}\frac{m_\alpha}{m_\beta}f_\beta f_\alpha \approx 4\pi\Gamma_{\alpha\beta}\frac{m_\alpha}{m_\beta}n_\beta\delta(\vec{v})f_\alpha$$

On the fast-particle velocity range, diffusion is highly anisotropic, the dominant part of the diffusion current being transverse:

$$-K_{ij}\frac{\partial f_\alpha}{\partial v_j} \approx -\frac{1}{2}\Gamma_{\alpha\beta}\frac{n_\beta}{v}\left(\delta_{ij} - \frac{v_i v_j}{v^2}\right)\frac{\partial f_\alpha}{\partial v_j} \tag{3}$$

As a result, (1) the hot particle distribution is stretched and smoothed out by collision terms outside of the cold particle region and (2) a cold component fed by slowed-down particles appears in the cold particle region (see Fig. 2).

2.2. Two-component description of distribution functions

The facts mentioned above suggest that all distributions be expressed as the sum of a cold and a hot components:

$$f_\alpha = f_{\alpha,c} + f_{\alpha,h}; \quad f_\beta = f_{\beta,c} + f_{\beta,h}$$

Collisions of the hot, smooth component $f_{\alpha,h}$ onto cold particles are governed by the divergenceless part of the slowing-down term and by the diffusion term, which take particles out of $f_{\alpha,h}$ (there is no source term for $f_{\alpha,h}$ from Fokker–Planck collisions with cold particles):

$$\left(\frac{\partial f_{\alpha,h}}{\partial t}\right)_c = 4\pi\frac{\partial f_{\alpha,h}}{\partial v_i}\sum_\beta\left(\Gamma_{\alpha\beta}\frac{m_\alpha}{m_\beta}\frac{\partial S_{\beta,c}}{\partial v_i}\right) - 4\pi\frac{\partial}{\partial v_i}\left(\frac{\partial f_{\alpha,h}}{\partial v_j}\sum_\beta\left(\Gamma_{\alpha\beta}\frac{\partial^2 T_{\beta,c}}{\partial v_j\partial v_i}\right)\right)$$

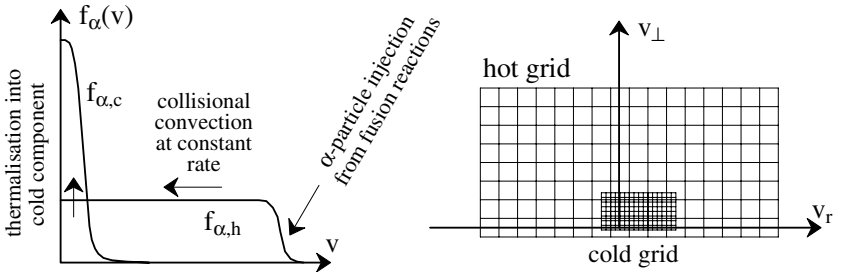

Fig. 2. Left: schematic plot of the hot and cold components of the α particle velocity distribution. Right: schematic representation of the discretization grids in cylindrical coordinates for those components (azimuthal symmetry is assumed about the longitudinal (v_r) velocity axis).

The cold component $f_{\alpha,c}$ is fed by the source term omitted from the above equation:

$$\frac{\partial f_{\alpha,c}}{\partial t} = 4\pi \frac{\partial}{\partial v_i}\left(f_{\alpha,c} \sum_\beta \left(\Gamma_{\alpha\beta} \frac{m_\alpha}{m_\beta} \frac{\partial S_\beta}{\partial v_i} \right) \right) - 4\pi \frac{\partial}{\partial v_i}\left(\frac{\partial f_{\alpha,c}}{\partial v_j} \sum_\beta \left(\Gamma_{\alpha\beta} \frac{\partial^2 T_\beta}{\partial v_j \partial v_i} \right) \right) + 4\pi f_{\alpha,h} \sum_\beta \left(\Gamma_{\alpha\beta} \frac{m_\alpha}{m_\beta} f_{\beta,c} \right).$$

3. Discretization of the hot component in the distribution

Fortunately, since the hot component $f_{\alpha,h}$ is everywhere smooth in velocity space, we can discretize it on a coarse grid. From the point of view of (approximately transverse) diffusion, $f_{\alpha,h}$ is close to equilibrium, because the fusion reaction source is isotropic.

3.1. Need for a specific scheme for advancing the hot component in time

Unfortunately, the slowing-down and diffusion terms applied to the hot component are highly peaked in magnitude near the cold component region, and diffusion is highly anisotropic outside of the cold component region. There is thus a need for some sort of "stiff" numerical method to deal with this diffusion problem. In the past, implicit schemes have been used. In 2D, an ADI scheme is implemented in our code FPion [3]. However, this scheme is not adapted to highly anisotropic situations, due to explicit cross-derivative terms (boxed terms below):

$$\frac{f^{n+\frac{1}{2}} - f^n}{\delta t/2} - D_{xx} f^{n+\frac{1}{2}} = D_{yy} f^n + \boxed{D_{xy} f + D_{yx} f}$$

$$\frac{f^{n+1} - f^{n+\frac{1}{2}}}{\delta t/2} - D_{yy} f^{n+1} = D_{xx} f^{n+\frac{1}{2}} + \boxed{D_{xy} f + D_{yx} f}$$

where f^n is the discretized distribution value at time $n\delta t$ and $D_{ij}\cdot$ stands for

$$\frac{\partial}{\partial v_i}\left(K_{ij} \frac{\partial \cdot}{\partial v_j} \right)$$

with no index summation implied. On the other hand, a straightforward implicit scheme requires the solution of a very large, ill-conditioned linear system. In the following, we will demonstrate how a simple explicit scheme can be modified to efficiently deal with the problem under study.

4. Dealing with the stability condition

For illustrating purposes, we will use a very straightforward conservative, centered-difference, explicit discretization on a rectangular grid for the diffusion equation

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial v_i}\left(K_{ij} \frac{\partial f}{\partial v_j} \right)$$

Let us emphasize that the principle of the time integration scheme presented here applies to any space discretization scheme which preserves the conservation form of the underlying equation, and is not restricted to the present simple example. Our simple scheme reads (here for definiteness on a cartesian 2D (v_x, v_y) grid, assuming mesh dimensions $\delta v_y = \delta v_x = \delta v$):

$$\frac{f_{ij}^{n+1} - f_{ij}^n}{\delta t} = -\frac{F_{xi+\frac{1}{2}j} - F_{xi-\frac{1}{2}j}}{\delta v} - \frac{F_{yij+\frac{1}{2}} - F_{yij-\frac{1}{2}}}{\delta v}$$

where the field quantity is taken at cell centers $f_{ij} = f(v_x = i\delta v, v_y = j\delta v)$ and the fluxes on cell boundaries. The discrete fluxes read

$$F_{xi+\frac{1}{2}j} = -K_{xxi+\frac{1}{2}j}\frac{f^n_{i+1j} - f^n_{ij}}{\delta v} - K_{xyi+\frac{1}{2}j}\frac{f^n_{i+1j+1} + f^n_{ij+1} - f^n_{i+1j-1} - f^n_{ij-1}}{4\delta v}$$

$$F_{yij+\frac{1}{2}} = -K_{yyij+\frac{1}{2}}\frac{f^n_{ij+1} - f^n_{ij}}{\delta v} - K_{yxij+\frac{1}{2}}\frac{f^n_{i+1j+1} + f^n_{i+1j} - f^n_{i-1j+1} - f^n_{i-1j}}{4\delta v}$$

The von Neumann stability condition for this scheme in the case of a constant homogeneous diffusion tensor **K** is well-known:

$$\frac{\delta t}{\delta v^2}\mathrm{Tr}(\mathbf{K}) \leqslant \frac{1}{2} \tag{4}$$

and leads to a time-step limitation which is usually considered prohibitive in problems where large values of the diffusion coefficient can occur.

4.1. The stability condition can be applied locally

The time-step limitation is "physically" related to the fact that in an explicit scheme, fluxes are computed once and for all from known field values, and then applied steadily during δt. Thus arbitrarily (and erroneously) high field variations can occur in a given cell if δt is not limited, leading to a breakdown of the computation. This local interpretation suggests that the stability condition (4) might be applied on each cell boundary *independently* from the others. When the maximum value $\delta t_{\max}$ as given by (4) is reached on a given boundary, the flux across it must be updated from the most recent field values available. Then the "tap" can be opened up again for an other $\delta t_{\max}$, until the required final time $t + \delta t$ is reached. This procedure is equivalent to using an effective value $\delta t_{\max}$ for the local time step, instead of the global value δt. This idea can be useful when the diffusion tensor varies strongly across the simulation region (as it does in the charged-particle Coulomb slowing-down problem), allowing one to use a small time step only where it is actually needed, and a larger time step everywhere else. To actually save computing time, cells must first be sorted according to their local value of $\delta t_{\max}$ and then addressed by the computation process only when they actually need to.

5. A locally-split-step explicit algorithm

A definite implementation of the above idea is as follows. For every iteration with the global time step δt, the following computations are performed:

First step – On each interface, the number of times that the time step must be halved to meet the stability criterion, based on the local value of the diffusion tensor, is evaluated. As an example (see Fig. 3), let us assume that fluxes indicated by light-weight arrows require δt, medium-weight arrows require $\delta t/2$ and heavy arrows require $\delta t/4$.

Second step – Whenever a flux needs to be updated, the field values on which it depends must be updated too (see Fig. 4). This sets a minimum update frequency for each cell.

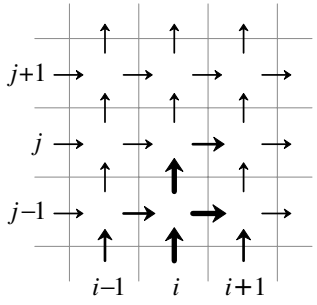

Fig. 3. An example of a flux pattern through discretization interfaces. Heavier arrows indicate larger diffusion tensors, leading to smaller time step limits for stability.

Third step – The field value in a given cell changes due to fluxes through its boundaries. Hence, the update frequency for each cell must be greater than or equal to the update frequencies for the fluxes on its boundaries; this condition may or may not be already enforced by the previous one (in step 2), depending on the specific scheme under study. This finally sets update frequencies (number of time-step halvings) for all fluxes and cells (see Fig. 5).

Fourth step – Cells and interfaces are first sorted according to their update frequency. This leads to the following sorted arrays of indexes:

`ijcell[nsplit][n] for n=0...nbcell[nsplit]-1` : index list of the `nbcell[nsplit]` cells where δt is halved `nsplit` times

`il2jbnd[nsplit][n] for n=0...nbfx[nsplit]-1` : index list of the `nbfx[nsplit]` x-boundaries where δt is halved `nsplit` times

`ijl2bnd[nsplit][n] for n=0...nbfy[nsplit]-1` : index list of the `nbfy[nsplit]` y-boundaries where δt is halved `nsplit` times

Last step – The computation *per se* then proceeds through a loop over the smallest split time-step found (namely, δt halved `nsplitmax` times):

```
for (int n=1; n<=(1<<nsplitmax); n++)
  for (int nsplit=nsplitmax; nsplit>=0; nsplit--)
    if ((n % (1<<nsplitmax-nsplit))==0){
      float dtsplit = dt/(1<<nsplit);
      // advance cells where δt is halved nsplit times:
      for (int indcell=0; indcell<nbcell[nsplit]; indcell++) {
        int ij = ijcell[nsplit][indcell];
             // (from sorted list of cells)
        Field[ij] += dtsplit*(Flux_x[i+1/2,j]-Flux_x[i-1/2,j]
                     +Flux_y[i,j+1/2]-Flux_y[i,j-1/2]);
      }
      // update fluxes where δt is halved nsplit times:
      if (nsplit>0) {
        // otherwise we don't need fluxes anymore (last iteration)
        for (int indflx=0; indflx<nbfx[nsplit]; indflx++) {
          int il2j = il2jbnd[nsplit][indflx];
               // (from sorted list of x-fluxes)
          // update x-flux on interface (i+1/2,j):
          Flux_x[il2j] = ...;
        }
        for (int indfly=0; indfly<nbfy[nsplit]; indfly++) {
          int ijl2 = ijl2bnd[nsplit][indfly];
               // (from sorted list of y-fluxes)
          // update y-flux on interface (i,j+1/2):
          Flux_y[ijl2] = ...;
        }
      }
    }
```

6. Computational cost of the LSSE algorithm for coulomb diffusion

The sorting of cells and boundaries using an efficient algorithm (e.g., "Heapsort" [13]) will take on the order of $N\log N$ operations for each time step where N is the number of cells in the velocity space grid, leading for a simulation duration Δt with a global time step δt to a first contribution to the computational cost:

$$N_{\mathrm{op1}} \sim \mathrm{O}\left(\frac{\Delta t}{\delta t} N \log N\right)$$

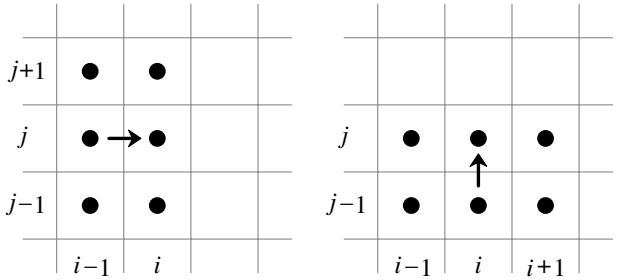

Fig. 4. For a simple space-centered discretization of the diffusion equation, due to the anisotropy of the diffusion tensor, the flux on a given interface (arrow) depends on the field values (circles) in all six cells surrounding that interface.

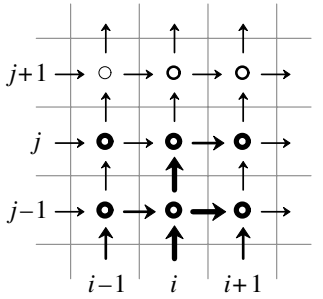

Fig. 5. Update frequencies for field values (circles) are deduced from those for fluxes. Heavier circles indicate larger update frequencies.

From (3), the order of magnitude of the diffusion coefficient in cell n is

$$K_n \sim \frac{v_n^2}{\tau_c(v_n)} \approx \frac{v_{\max}^2}{\tau_c(v_{\max})} \frac{v_{\max}}{v_n}$$

where $v_{\max}$ is the typical fast-particle velocity and $\tau_c(v)$ is the Coulomb slowing-down time defined by (2). The number of operations for advancing f_α over the simulation duration Δt in all N cells scales as

$$N_{\mathrm{op2}} \sim \sum_{n=1}^{N} \frac{\Delta t}{\delta t_n}$$

where the locally-split time step δt_n in cell n is constrained by the local value of the diffusion coefficient K_n, leading to

$$N_{\mathrm{op2}} \sim \sum_{n=1}^{N} \frac{K_n \Delta t}{\delta v^2} \approx \frac{v_{\max}^2 \Delta t}{\delta v^2 \tau_c(v_{\max})} \sum_{n=1}^{N} \frac{v_{\max}}{v_n}$$

where δv is the width of a velocity grid cell and v_n is the modulus of the velocity in cell n. If the dimension d of the simulation space is greater than 1, the sum on the right-hand side is dominated by large-velocity terms:

$$\sum_{n=1}^{N} \frac{v_{\max}}{v_n} \sim \int_0^{v_{\max}} \frac{v_{\max}}{v} \frac{v^{d-1} \mathrm{d}v}{\delta v^d} \sim \frac{v_{max}^d}{\delta v^d} \sim N$$

so that

$$N_{\mathrm{op2}} \sim \frac{v_{\max}^2 \Delta t}{\delta v^2 \tau_c(v_{\max})} N \sim \mathrm{O}\left(\frac{\Delta t}{\tau_c(v_{\max})} N^{\frac{d+2}{d}} \right)$$

This is to be compared with the number of operations for advancing f_α over Δt with a time step δt using a direct implicit resolution. A direct method involves, at every time step, the solution of a linear system of order N with $\mathrm{O}\left(N^{\frac{d-1}{d}}\right)$ non-vanishing diagonals, which takes about

$$N_{\mathrm{op}}(\mathrm{implicit}) \sim \mathrm{O}\left(\frac{\Delta t}{\delta t} N^{\frac{3d-2}{d}}\right)$$

operations. In the present case of a sparse matrix, more efficient, iterative methods are known, such as the various flavours of the conjugate-gradient algorithm [5,6]. The asymptotic convergence rate of those methods is known [5] to scale as $1/\sqrt{p}$ where p is the condition number of the matrix of the linear system at hand. For usual problems of numerical analysis this number in turn scales as $p \approx \mathrm{O}(N)$, and due to the matrix sparsity the number of operations per iteration scales as N. The overall number of operations for advancing f_α over Δt with a time step δt with such a method is thus expected to scale as

$$N_{\mathrm{op}}(\mathrm{iterative}) \sim \mathrm{O}\left(\frac{\Delta t}{\delta t} N^{3/2}\right)$$

If we wish to resolve at least the largest collision time in the system, then we need at most $\delta t \approx \tau_{\mathrm{c}}(v_{\max})$, so that

$$N_{\mathrm{op}}(\mathrm{LSSE}) \sim (N_{\mathrm{op}}(\mathrm{implicit}))^{\frac{d+2}{3d-2}} \sim (N_{\mathrm{op}}(\mathrm{iterative}))^{\frac{2d+4}{3d}} \tag{5}$$

at least as long as the time step remains sufficiently large that sorting does not take up most of the computing time, that is, provided that

$$\delta t \geqslant \frac{\log N}{N} \tau_{\mathrm{c}}(v_{\max})$$

From (5) we thus conclude that the efficiency of the LSSE method for large systems is equivalent to (for two-dimensional problems) or better than (for three-dimensional problems) that of a direct implicit method, but not quite as good as that of an iterative method. However, this conclusion might be mitigated by the large number of operations per iteration required by the latter, and by the fact that the specific behaviour of the diffusion tensor in the case of Coulomb collisions is expected to lead to larger condition numbers p than are expected in more or less homogeneous problems. More importantly, the solution provided by the LSSE method, due to local time-step splitting, is expected to be closer to the solution of the actual time-dependent problem under study, whereas implicit methods effectively yield, at every time step, a stationary solution of the corresponding time-dependent problem unless the time step δt is comparable to the shortest physical time scale in the system. In the present case of the collisional relaxation of fusion α particles, this is all the more true since the Coulomb slowing-down time becomes very small in the vicinity of the cold field particles. In addition in this case, the specific behaviour of the relaxation process as described in Section 2.1 allows us to use the LSSE scheme on a coarse grid only (the hot-particle grid in Fig. 2), thus alleviating its potentially higher computational cost while taking advantage of the better accuracy that it provides.

7. Explicit treatment of the slowing-down term

Although a straightforward centered-difference discretization of the advection equation

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial v_i}(u_i f) = 0$$

is known to be always unstable [14], the diffusion term in the Fokker–Planck equation (1) provides a stabilizing effect if diffusion is strong enough. We check that this is indeed the case for the Coulomb slowing-down of fast particles. In the homogeneous case, the von Neumann stability analysis of a straightforward explicit discretization of the Fokker–Planck equation with the centered-difference fluxes

$$\begin{aligned} F_{xi+\frac{1}{2}j} &= u_{xi+\frac{1}{2}j} \frac{f^n_{i+1j} + f^n_{ij}}{2} - K_{xxi+\frac{1}{2}j} \frac{f^n_{i+1j} - f^n_{ij}}{\delta v} - K_{xyi+\frac{1}{2}j} \frac{f^n_{i+1j+1} + f^n_{ij+1} - f^n_{i+1j-1} - f^n_{ij-1}}{4\delta v} \\ F_{yij+\frac{1}{2}} &= u_{yij+\frac{1}{2}} \frac{f^n_{ij+1} + f^n_{ij}}{2} - K_{yyij+\frac{1}{2}} \frac{f^n_{ij+1} - f^n_{ij}}{\delta v} - K_{yxij+\frac{1}{2}} \frac{f^n_{i+1j+1} + f^n_{i+1j} - f^n_{i-1j+1} - f^n_{i-1j}}{4\delta v} \end{aligned} \tag{6}$$

leads to the following stability conditions:

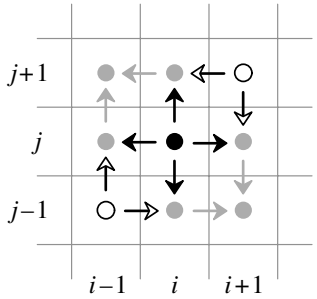

Fig. 6. A simple space-centered discretization of the 2D diffusion equation exhibits non-positivity when the diffusion tensor is strongly anisotropic with eigendirections pointing away from the discretization axes, as displayed in the case where the distribution f_a is initially localized in a single cell (black dot). Applying the fluxes (6) with the diffusion tensor given by (7) for one time step leads to positive values of the distribution along the main eigendirection (grey dots) and negative values across (open dots), due to spurious cross-derivative fluxes (open arrows).

$$\left(\frac{u\delta t}{\delta v}\right)^2 \leqslant \frac{2\mathrm{Tr}(\mathbf{K})\delta t}{\delta v^2} \leqslant 1$$

In the case of the Coulomb slowing-down of fast particles, the relevant part of the diffusion tensor to take into account is the (much smaller) radial part:

$$K_\parallel \approx -4\pi\Gamma_{\alpha\beta}\frac{\mathrm{d}^2 T_\beta}{\mathrm{d}v^2} \approx \frac{\Gamma_{\alpha\beta} n_\beta v_\beta^2}{v^3} \quad \text{and} \quad u \approx 4\pi\Gamma_{\alpha\beta}\frac{m_\alpha}{m_\beta}\frac{\partial S_\beta}{\partial v_i} \approx \frac{m_\alpha}{m_\beta}\frac{\Gamma_{\alpha\beta} n_\beta}{v^2}$$

where n_β and v_β are the density and the thermal velocity of the target particles. The stability criteria then read:

$$\delta t \leqslant 2\left(\frac{m_\beta}{m_\alpha}\right)^2 \frac{v_\beta^2 v}{\Gamma_{\alpha\beta} n_\beta} \quad \text{and} \quad \delta t \leqslant \frac{\delta v^2 v}{\Gamma_{\alpha\beta} n_\beta}$$

These conditions ensure the stability of the LSSE scheme, but not its positivity due to diffusive cross-currents, as can be seen by estimating those in the case where f_a is localized in a single cell. This is illustrated in Fig. 6 in the worst case of complete anisotropy with eigenvectors at 45° to the discretization axes, where the diffusion tensor takes on the following form:

$$K_{ij} = \frac{K_\parallel}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{7}$$

Additional flux-limiting techniques are thus needed to mitigate that effect, but are out of the scope of this paper which focusses on the time integration of conservation equations.

8. An example simulation of fast-particle relaxation

As a test case for our new numerical scheme, the relaxation of a distribution of hot charged particles through ion–ion collisions in the presence of a cold ion background was computed using the LSSE algorithm. The simulation solved the Fokker–Planck equation (1) in velocity space, governing collisions of test particles with target ions (ion–electron collisions were not taken into account in this calculation), assuming azimuthal symmetry around the v_r axis. The discretization grid was thus two-dimensional in cylindrical coordinates $(v_r, v_\perp)$, with, in the present case, 101×50 cells respectively. The initial condition, as depicted in the top left frame of Fig. 7 at time $t = 0$, is located in a single cell, and the distribution function of target particles is a Maxwellian with temperature T_β and mean velocity $v_\beta = 0$. Velocities are expressed in units of the thermal velocity and time in units of the corresponding thermal collision time:

$$v_0 = \left(\frac{2k_B T_\beta}{m_\beta}\right)^{1/2} \quad \text{and} \quad t_0 = \tau_c(v_0) = \frac{m_\beta v_0^3}{\Gamma_{\alpha\beta} n_\beta m_\alpha}$$

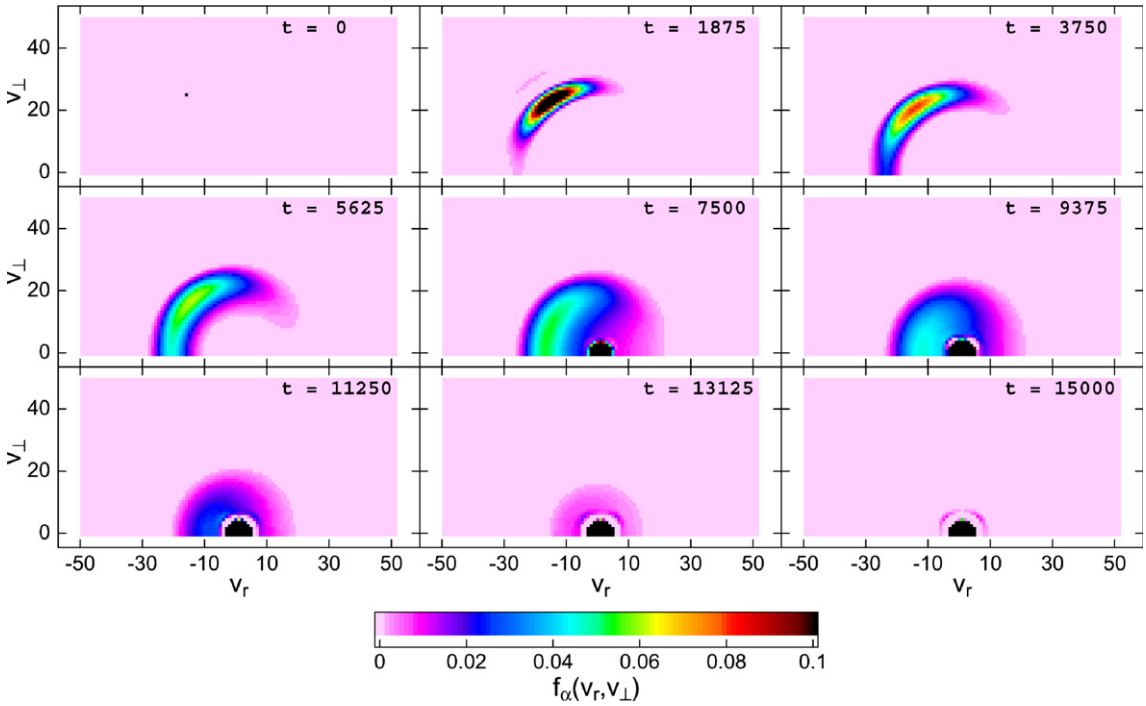

Fig. 7. Successive snapshots (from left to right and top to bottom) of a Fokker–Planck simulation of the relaxation of an initially localized distribution f_α of energetic α particles due to Coulomb collisions with the ions of a background cold deuterium-tritium plasma. The 2D cylindrical velocity space $(v_r, v_\perp)$ is normalized to the thermal velocity of the background ions. Time is normalized to the corresponding thermal collision time. The calculation displayed used 200 iterations of the LSSE scheme with an overall (unsplit) time step of 75 thermal collision times ($\delta t = 75$).

For conditions typical of an ICF capsule hot spot [1], t_0 is of the order of 20 fs. The global time step was $\delta t = 75$ thermal collision times.

The first stage of the relaxation, for $t = 0$–7500 (in the units defined above), exhibits the expected strongly anisotropic diffusion in the tangential direction, along with steady convection of hot particles towards the target plasma region. In a second, thermalization stage, from $t = 7500$ onwards, as slowed-down particles start reaching low velocities, since the slowing-down current divergence is localized on target particles, a cold component builds up in the target particle region in velocity space as explained in Sections 2.1 and 2.2, which ultimately absorbs all hot particles. In that stage, diffusion becomes more and more isotropic but the physical time scale gets down to the short cold collision time. The LSSE algorithm allows us to use a global time step comparable with the initial slowing-down and diffusion time scales, which are much larger than the cold particle collision time, the latter being taken into account in the local time-step refinement process.

We checked that the final state reached by the test particle distribution is, within the limited accuracy allowed by the finite resolution of the numerical grid in the cold particle region, a Maxwellian at the temperature T_β of the cold target particle distribution, which is held constant in this calculation. A better agreement would of course be reached by using a refined discretization scheme in velocity space (e.g., Chang–Cooper weighting [15]), but this question is out of the scope of the present work. Let us mention that the velocity dependence of the cold component of the distribution cannot be estimated on the figures shown, because the graphical scale in those plots was adjusted for a better display of the hot component, which is our main concern in this work, causing the cold component to be very quickly truncated due to the much larger numerical values taken by the distribution function there.

As a diagnostics of the scheme, the number of cells *nbcell* in which the time step was split in two *n* times was printed as a function of *n* at the end of the calculation:

nbcell[0] =0
nbcell[1] =906
nbcell[2] =2512
nbcell[3] =1194
nbcell[4] =316
nbcell[5] =89
nbcell[6] =33

8.1. Comparison with a straightforward explicit scheme

To demonstrate the benefit of using the LSSE scheme in terms of computing time, the same problem was solved again using a straightforward global explicit scheme, with a much smaller time step for stability. The results are displayed in Fig. 8. The differences in the values found for the distribution function at successive times are very small, thus demonstrating that the accuracy of the LSSE scheme is not degraded with respect to the straightforward explicit scheme, although some cells in the present calculation are updated only 32 times less often than in the global explicit scheme. However the computing time is in this case ≈9 times as large as for the LSSE case. The maximum CPU time reduction allowed by the LSSE scheme can be estimated on the basis of the total number of times N_u that a cell is updated during the calculation:

$$N_u = \sum_{n=1}^{N} \frac{\Delta t}{\delta t_n}$$

where, as before, N is the number of cells in the discretized grid, and δt_n is the local time step in cell number n. In the case of the global explicit scheme, this number is obviously:

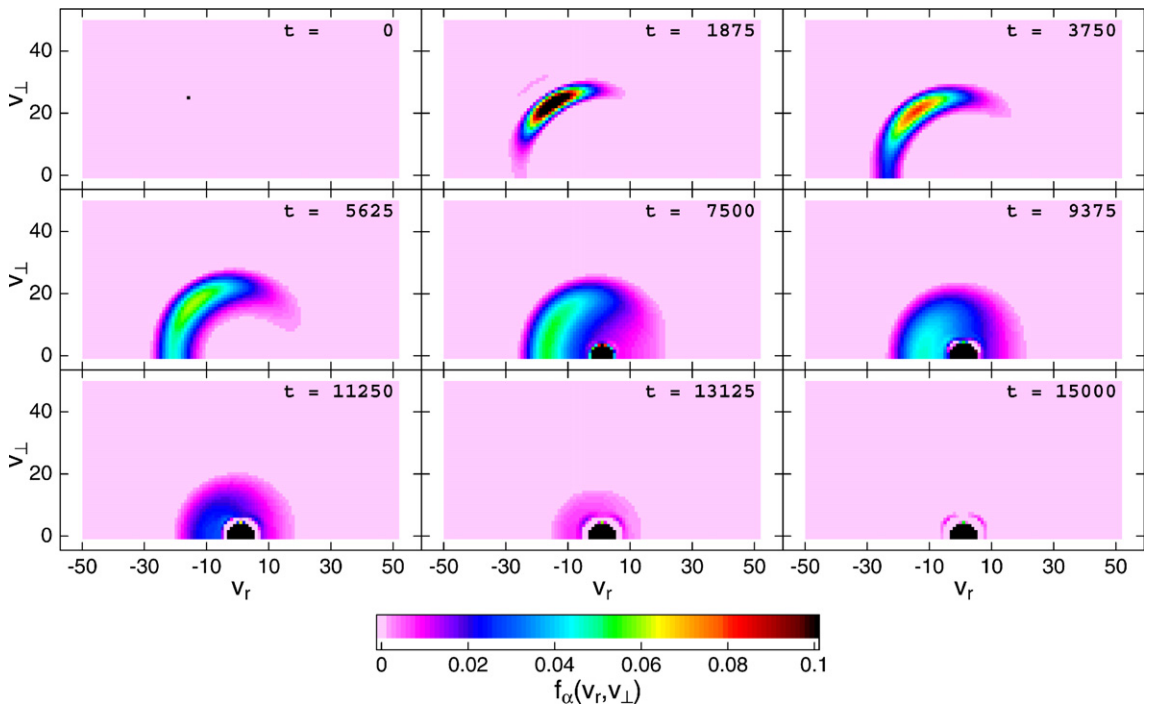

Fig. 8. The same problem as in Fig. 7 is solved using a straightforward explicit scheme for comparison. The calculation used 12,800 iterations of the explicit scheme with a time step equal to the smallest split time step found in the LSSE scheme ($\delta t = 1.171875$).

$$N_{\mathrm{u}}(\mathrm{expl}) = N\frac{\Delta t}{\delta t_{\mathrm{expl}}}$$

whereas in the case of the LSSE scheme:

$$N_{\mathrm{u}}(\mathrm{LSSE}) = \sum_{n_s=0}^{n_{s,\max}} \mathtt{nbcell}[n_s]\frac{2^{n_s}\Delta t}{\delta t_{\mathrm{LSSE}}}$$

Inserting the figures quoted above, the update number ratio is calculated:

$$\frac{N_{\mathrm{u}}(\mathrm{expl})}{N_{\mathrm{u}}(\mathrm{LSSE})} \approx 12.05$$

which shows that a fairly good CPU time reduction (by a factor 9, as compared with the largest possible value 12) was achieved in this example calculation. Part of the discrepancy between the expected and observed reductions is due to the computational overhead brought about by the cell-sorting stage in the LSSE scheme.

8.2. Comparison with a standard implicit scheme

The same problem was solved again using a standard bi-conjugate gradient inversion routine from Ref. [13], namely the `LINBCG` routine. Two different values of the global time step were used. In the first case (displayed in Fig. 9), the time step was the same as in the LSSE case (compare with Fig. 7), for comparison with our new algorithm in terms of CPU time. The calculation displayed in Fig. 9 took 5 times as much CPU time as the reference LSSE case of Fig. 7, but the accuracy was satisfactory, with only very small differences in the field values found with respect to Figs. 7 and 8. In the second case (displayed in Fig. 10), a much larger time step was used to check whether the implicit scheme can compete with our new scheme in terms of CPU time while keeping a reasonable accuracy. Although that case still took twice as much CPU time as the reference

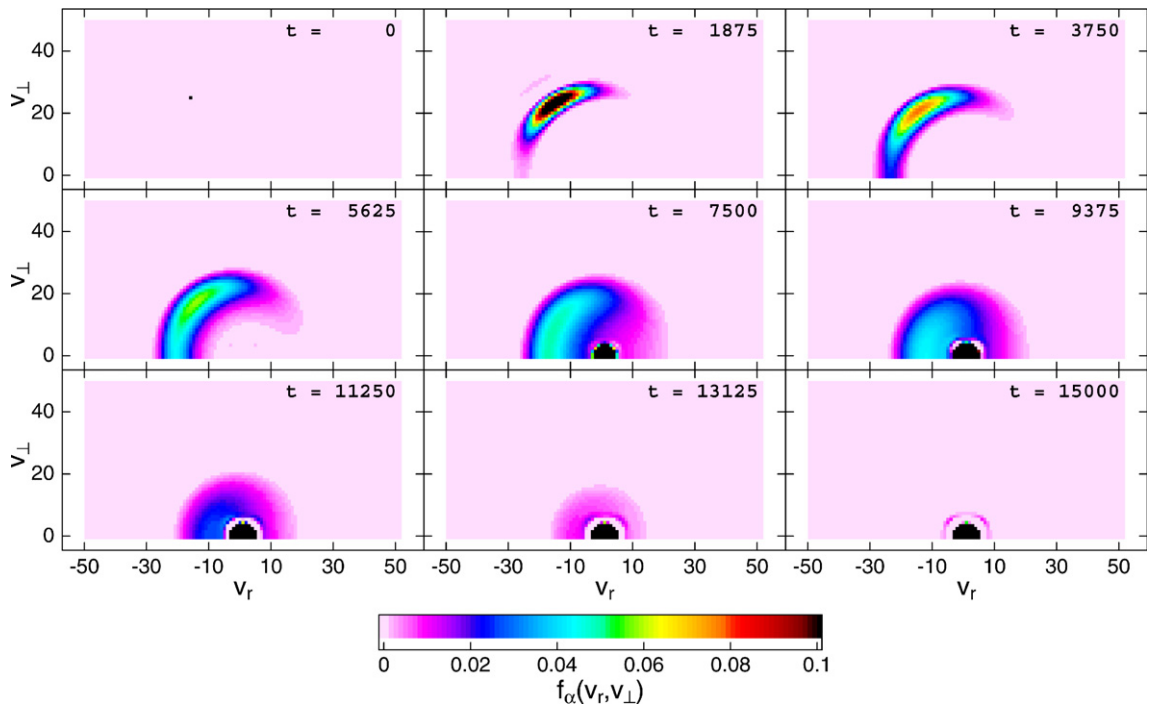

Fig. 9. The same problem as in Fig. 7 is solved using a standard implicit scheme. The calculation used 200 iterations of the `LINBCG` routine from Ref. [13] with a time step equal to the global time step used in the LSSE scheme ($\delta t = 75$).

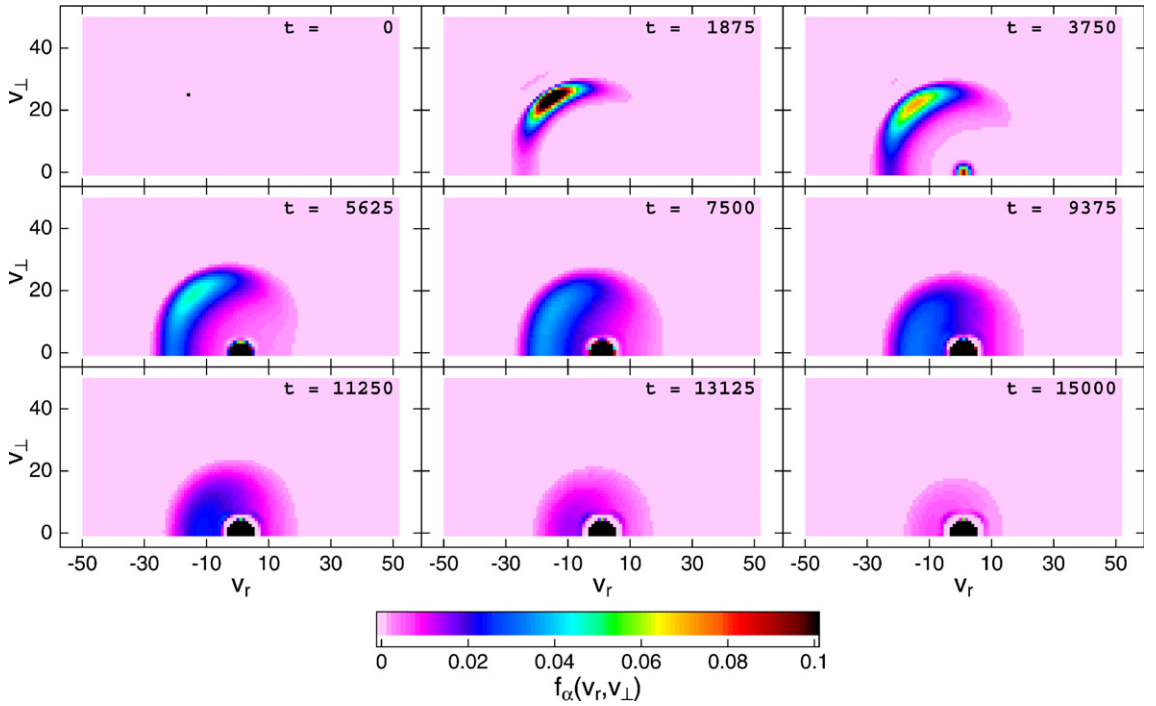

Fig. 10. The same problem as in Fig. 7 is solved using a standard implicit scheme, but with a much larger time step than in Fig. 9. The calculation used 24 iterations of the LINBCG routine with a time step $\delta t = 625$.

LSSE case, due to the large time step there are now noticeable discrepancies in the results; among other things, the thermal component at low velocity starts building up much earlier.

Let us mention that CPU time does not scale linearly with the number of iterations in the implicit case because the bi-conjugate gradient algorithm used is an iterative method, and the number of iterations needed to reach convergence is not fixed. Only the convergence tolerance (maximum acceptable relative error) is fixed (at 10^{-5} in the present case). The CPU time needed can be reduced if the tolerance is relaxed to larger values, at the price of a diminished accuracy of the scheme.

As a conclusion, due to the large amount of computation needed for every iteration in the implicit scheme, in the case displayed (which is relevant to the envisioned application of fast particle slowing-down in ICF), the fastest time-integration method turns out to be the LSSE scheme, even when a reduced accuracy is accepted. This does not contradict the scaling laws of Section 6 which give the asymptotic behaviour of the scheme in the limit of very fine grids with a large number of cells N.

9. Conclusion and prospects

We reviewed the relaxation of a distribution of fast particles (e.g., α particles from thermonuclear fusion reactions) in a plasma, due to ion–ion Coulomb collisions. Initially the dominant processes are a transverse diffusion leading to isotropization of the fast-particle distribution, along with a radial slowing-down at constant rate in velocity space. As a result, the fast particle distribution gets flattened by collisions. When particles start reaching low velocities typical of field particles, a cold component builds up in the distribution and thermalizes.

For the purpose of including fast alpha particle dynamics in our Vlasov–Fokker–Planck code FPion, we addressed the kinetic numerical simulation of a population of fast particles in the presence of the much colder background plasma. Specific schemes are needed for dealing with that situation due to the strong dependence of collisional coefficients on relative particle velocity. We describe a specially-taylored explicit scheme, involving a local time-step refinement process, which seems well-adapted to that problem.

Further work should address the generation of secondary fast particles through large-angle scattering, to be modelled by a Boltzmann collision term, before a concrete implementation of our new scheme into FPion can be attempted.

In addition, we believe that the method proposed can be valuable in other fields of computational physics where diffusion problems with highly inhomogeneous and/or highly anisotropic diffusion tensors arise.

Acknowledgement

We wish to mention stimulating discussions on this work with Dr D. Colombant.

References

[1] J.D. Lindl, Inertial Confinement Fusion – The Quest for Ignition and Energy Gain using Indirect Drive, Springer-Verlag, New York, 1998.
[2] F. Vidal, J.-P. Matte, M. Casanova, O. Larroche, Phys. Rev. E 52 (1995) 4568.
[3] O. Larroche, Eur. Phys. J. D 27 (2003) 131.
[4] M.N. Rosenbluth, W.M. MacDonald, D.L. Judd, Phys. Rev. 107 (1957) 1.
[5] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, 2003.
[6] D.S. Kershaw, J. Comput. Phys. 26 (1978) 43.
[7] F.B. Hildebrand, Introduction to Numerical Analysis, 2nd ed., Dover Publication, New York, 1974.
[8] G.I. Marchuk, Methods of Numerical Mathematics, Springer-Verlag, New York, 1982.
[9] M. Lemou, J. Comput. Phys. 157 (2000) 762.
[10] C. Buet, S. Cordier, J. Comput. Phys. 179 (2002) 43.
[11] N.A. Krall, A.W. Trivelpiece, Principles of Plasma Physics, McGraw-Hill, New York, 1973.
[12] D.V. Sivukhin, in: M.A. Leontovich (Ed.), Reviews of Plasma Physics, vol. 4, Consultants Bureau, New York, 1966, p. 93.
[13] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, Cambridge University Press, Cambridge, 1992.
[14] R.D. Richtmyer, K.W. Morton, Difference Methods for Initial-value Problems, second ed., Interscience Publ., Wiley & Sons, New York, 1967.
[15] J.S. Chang, G. Cooper, J. Comput. Phys. 6 (1970) 1.