Journal of Computational Physics **157,** 654–682 (2000)
doi:10.1006/jcph.1999.6395, available online at http://www.idealibrary.com on IDEAL®

An Implicit Energy-Conservative 2D Fokker–Planck Algorithm

II. Jacobian-Free Newton–Krylov Solver

L. Chacón,[*] D. C. Barnes,[†] D. A. Knoll,[‡] and G. H. Miley[*]

[*]*Fusion Studies Laboratory, University of Illinois at Urbana-Champaign, 103 South Goodwin Avenue, Urbana, Illinois 61801; [†]X-PA, Los Alamos National Laboratory, MS B259, Los Alamos, New Mexico 87545; and [‡]X-HM, Los Alamos National Laboratory, MS D413, Los Alamos, New Mexico 87545*
E-mail: chacon@lanl.gov, dbarnes@lanl.gov, nol@lanl.gov, g-miley@uiuc.edu

Received March 11, 1999; revised October 9, 1999

Energy-conservative implicit integration schemes for the Fokker–Planck transport equation in multidimensional geometries require inverting a dense, non-symmetric matrix (Jacobian), which is very expensive to store and solve using standard solvers. However, these limitations can be overcome with Newton–Krylov iterative techniques, since they can be implemented *Jacobian-free* (the Jacobian matrix from Newton's algorithm is never formed nor stored to proceed with the iteration), and their convergence can be accelerated by *preconditioning* the original problem. In this document, the efficient numerical implementation of an implicit energy-conservative scheme for multidimensional Fokker–Planck problems using multigrid-preconditioned Krylov methods is discussed. Results show that multigrid preconditioning is very effective in speeding convergence and decreasing CPU requirements, particularly in fine meshes. The solver is demonstrated on grids up to 128×128 points in a 2D cylindrical velocity space (v_r, v_p) with implicit time steps of the order of the collisional time scale of the problem, τ. The method preserves particles exactly, and energy conservation is improved over alternative approaches, particularly in coarse meshes. Typical errors in the total energy over a time period of 10τ remain below a percent. © 2000 Academic Press

Key Words: implicit plasma simulation; energy-conservative Fokker–Planck; non-symmetric systems; preconditioned Krylov methods; multigrid methods.

1. INTRODUCTION

The Fokker–Planck equation can be shown [1] to satisfy important intrinsic symmetries such as particle, momentum, and energy conservation, and preservation of the positivity of the solution. It also satisfies the H-theorem, which implies that the Maxwell–Boltzmann distribution is the solution in thermal equilibrium. The preservation of these symmetries in the numerical approximation to the problem is essential to model the physics adequately. Particle conservation is straightforward to achieve numerically [2]; however, energy conservation is not.

Energy-conservative algorithms for Fokker–Planck problems were first addressed by Epperlein [3], who showed that numerical energy conservation is conceptually and practically possible in one-dimensional Fokker–Planck problems for any time step size, provided that the energy moment of the Fokker–Planck collision operator cancels numerically. However, the generalization of Epperlein's energy-conservative method for multidimensional Fokker–Planck problems in velocity space is non-trivial because the implicit time integration requires inverting a dense, non-symmetric Jacobian matrix. This matrix, which stems from the integral nature of the coefficients in the equation, is very expensive to form (the number of operations to form it scales as $O(N^2)$, where N is the number of unknowns) and store (it would require 2 GB of free memory in 8-byte precision for a two-dimensional 128×128 mesh), and its inversion is very inefficient with both stationary iterative techniques and direct solvers.

The development of a suitable difference scheme that improves the numerical cancellation of the energy moment of the Fokker–Planck collision operator in multidimensional geometries has been successfully accomplished by the authors in Ref. [4]. It is the objective of this paper to propose a solver that deals efficiently with the dense, non-symmetric algebraic problem that results from such formulation, with minimal storage and runtime requirements.

The energy-conservative Fokker–Planck solver developed herein involves three different classes of algebraic problems, namely, non-symmetric dense systems (stemming from the Newton–Raphson algorithm), non-symmetric sparse systems, and symmetric positive definite (SPD) sparse systems. A direct approach would be very inefficient due to the prohibitive storage requirements of the dense systems and due to the inefficiency of standard solvers (such as direct solvers or stationary iterative techniques) for large, non-symmetric matrices. Krylov iterative methods, however, are well suited for this task [5] because of their improved efficiency over standard solvers and because they are suitable for a *Jacobian-free* [6, 7] implementation of Newton's algorithm, i.e., without ever forming or storing the Jacobian matrix, thus alleviating storage requirements. Furthermore, convergence in Krylov methods can be accelerated by *preconditioning* the system [8], which consists in operating on the original matrix with the inverse of a new matrix—called a preconditioner—that, while approximating the eigenvalue spectra of the original matrix, is easily invertible. In this work, a sparse preconditioning matrix is formed by lagging integral information at the previous time step [9]. An approximate inverse to this preconditioning matrix is constructed using simple multigrid methods [10, 11]. An advantage of this technique is that it renders the number of Krylov iterations virtually independent of the number of mesh points (as opposed to the power scaling typical of stationary iterative techniques). Among the available Krylov techniques, the *c*onjugate *g*radient (CG) method [12] is used for SPD systems; the *g*eneralized *m*inimal *r*esiduals (GMRES) method [13] is chosen for non-symmetric systems because of its robust convergence properties, particularly in Jacobian-free applications [7].

The rest of the presentation in this paper assumes that the reader is familiar with the work in Ref. [4], and is organized as follows. Section 2 reviews the energy-conservation issues of the Fokker–Planck equation in a multidimensional velocity space that are crucial for the development of an energy-conservative solver. Section 3 discusses the time integration technique of the governing equation to preserve both energy and particles. Section 4 deals with the specifics of the Krylov algorithm, the Jacobian-free technique, the preconditioning step, and the details of the numerical implementation of these techniques in this specific application, and Section 5 presents some results to illustrate the performance and limitations of the solver.

2. ENERGY CONSERVATION ISSUES OF THE FOKKER–PLANCK EQUATION IN A MULTIDIMENSIONAL VELOCITY SPACE

For the purpose of studying the energy conservation numerical issues of the Fokker–Planck equation, the convection and field transport terms present in the most general form of the Boltzmann transport equation can be ignored (equivalent to assuming that the plasma is field-free and spatially homogeneous). For a single species problem, this results in the following simplified Fokker–Planck equation [4],

$$\frac{\partial f}{\partial t} = L(f) = -\Gamma \frac{\partial}{\partial \mathbf{v}} \cdot \left[f \frac{\partial H(f)}{\partial \mathbf{v}} - \frac{1}{2} \frac{\partial}{\partial \mathbf{v}} \cdot \left(\frac{\partial^2 G(f)}{\partial \mathbf{v} \partial \mathbf{v}} f \right) \right] = -\frac{\partial}{\partial \mathbf{v}} \cdot \mathbf{J}_{FP}, \tag{1}$$

where $\Gamma = 4\pi e^4 \lambda / m^2$, with λ the Coulomb logarithm, and e, m the charge and mass of the species under consideration. In Eq. (1), $\mathbf{J}_{FP}$ is the Fokker–Planck flux, defined as

$$\mathbf{J}_{FP} = \Gamma \left[f \frac{\partial H(f)}{\partial \mathbf{v}} - \frac{1}{2} \frac{\partial}{\partial \mathbf{v}} \cdot \left(\frac{\partial^2 G(f)}{\partial \mathbf{v} \partial \mathbf{v}} f \right) \right]$$

which is formed by a friction term (proportional to f) and a diffusion term (proportional to $\partial f / \partial \mathbf{v}$). The friction and diffusion coefficients are expressed in terms of the Rosenbluth potentials [14] $H(f)$ and $G(f)$, defined as

$$\nabla_{\mathbf{v}}^2 H = -8\pi f \tag{2}$$

$$\nabla_{\mathbf{v}}^2 G = H. \tag{3}$$

The Fokker–Planck flux can be reformulated as [4]

$$\mathbf{J}_{FP} = -\Gamma \left[\overline{T}[H, H] + \frac{1}{2} \frac{\partial}{\partial \mathbf{v}} \cdot \left(\frac{\partial^2 G(f)}{\partial \mathbf{v} \partial \mathbf{v}} f \right) \right],$$

where $\overline{T}[H, H]$ is a symmetric, bilinear operator on H, analogous to the Maxwell Stress tensor in electromagnetic theory [15], given by

$$\overline{T}[H, H] = \frac{1}{8\pi} \left[\frac{\partial H}{\partial \mathbf{v}} \frac{\partial H}{\partial \mathbf{v}} - \frac{\overline{I}}{2} \left(\frac{\partial H}{\partial \mathbf{v}} \right)^2 \right]. \tag{4}$$

Here, $\overline{I}$ is the identity dyadic. Then, the single species Fokker–Planck equation transforms into

$$\frac{\partial f}{\partial t} = \Gamma \frac{\partial}{\partial \mathbf{v}} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \left[\overline{T}[H, H] + \frac{1}{2} \frac{\partial^2 G(f)}{\partial \mathbf{v} \partial \mathbf{v}} f \right]. \tag{5}$$

This is the so-called tensor Fokker–Planck formulation [4]. Energy conservation can be shown from this formulation directly, without reverting to an integral formulation of the Rosenbluth potentials, as in the Landau formulation of the Fokker–Planck collision operator. Proof is given in Ref. [4]; the parts relevant to the present discussion are reproduced below.

The rate of change of energy in an ensemble of particles distributed according to f is given by

$$\frac{\partial E}{\partial t} = \int_{\Omega_\infty} d\mathbf{v} \frac{v^2}{2} \frac{\partial f}{\partial t}. \tag{6}$$

In this equation, Ω_∞ represents the infinite velocity domain. After introducing Eq. (5) and integrating by parts once, Eq. (6) reads as

$$\begin{aligned}\frac{\partial E}{\partial t} &= -\int_{\Omega_\infty} d\mathbf{v} \frac{v^2}{2} \frac{\partial}{\partial \mathbf{v}} \cdot \mathbf{J}_{FP} = \int_{\Omega_\infty} d\mathbf{v}\mathbf{v} \cdot \mathbf{J}_{FP} \\ &= \Gamma\left\{ \int_{\Omega_\infty} d\mathbf{v}\mathbf{v} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \bar{T}[H,H] + \frac{1}{2} \int_{\Omega_\infty} d\mathbf{v}\mathbf{v} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \left[\frac{\partial^2 G(f)}{\partial \mathbf{v} \partial \mathbf{v}} f\right]\right\}. \end{aligned} \tag{7}$$

The boundary integrals, which are zero at infinity (provided that f, G, H are regular at infinity), are omitted here. The second integral in Eq. (7) is integrated again by parts to find, according to the definitions of H and G,

$$\int_{\Omega_\infty} d\mathbf{v}\,\mathbf{v} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \left[\frac{\partial^2 G(f)}{\partial \mathbf{v}\, \partial \mathbf{v}} f\right] = -\int_{\Omega_\infty} d\mathbf{v}\, f \nabla_{\mathbf{v}}^2 G(f) = \frac{1}{8\pi} \int_{\Omega_\infty} d\mathbf{v} H \nabla_{\mathbf{v}}^2 H. \tag{8}$$

Then, the energy moment yields [4]

$$\frac{\partial E}{\partial t} = \int_{\Omega_\infty} d\mathbf{v} \frac{v^2}{2} \frac{\partial f}{\partial t} = -\Gamma \int_{\Omega_\infty} d\mathbf{v} \left[\mathbf{v} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \bar{T}[H,H] + Q[H,H]\right] = 0, \tag{9}$$

where the bilinear operator $Q[H, H]$ is defined as

$$Q[H,H] = \frac{H \nabla_{\mathbf{v}}^2 H}{16\pi}. \tag{10}$$

The result in Eq. (9) has been obtained for an infinite velocity domain, but also applies for finite domains provided that the distribution function is absolutely confined in them [4].

Two distinct steps are identified in the numerical description of the problem, namely, the discretization in time and the discretization in velocity space. Both steps are crucial for the adequate preservation of particles and energy. An energy-conservative discretization of the Fokker–Planck collision operator in velocity space has been developed for a 2D cylindrical velocity space (v_r, v_p) (Fig. 1) by the authors in Ref. [4], which ensures that Eq. (9) (necessary condition to develop an energy-conservative Fokker–Planck solver) be satisfied numerically *within* the domain. In this scheme, the Rosenbluth potentials H and G have to be obtained from a second-order discretization of the differential problems

$$Q[H,H] = -\frac{Hf}{2} \tag{11}$$

$$\nabla^2 G = H. \tag{12}$$

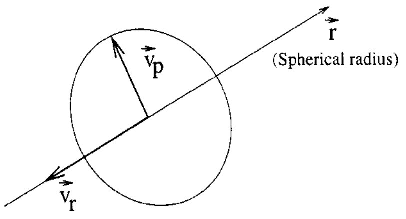


FIG. 1. Diagram of the local cylindrical velocity coordinate system considered in this work. Cylindrical symmetry is assumed. The spherical radius vector is included for reference.

Notice that the linear Poisson problem in H (Eq. (2)) has been replaced by a non-linear differential problem in terms of $Q[H, H]$ (Eq. (10)). The algebraic formulation and the solution algorithm for these two problems is discussed in detail in Subsection 4.4.

However, the numerical cancellation of Eq. (9) fails at the boundaries. The cancellation error has been predicted [4] to scale as $\sim(1/Nv_{limit})$, where N is the total number of mesh points and v_{limit} is the velocity domain limit. The issue of the numerical propagation of these errors in time is crucial and will be addressed in Subsection 5.3.

3. IMPLICIT ENERGY-CONSERVATIVE TIME INTEGRATION OF THE FOKKER–PLANCK EQUATION

An efficient numerical time integration of Eq. (5) requires an implicit discretization in time, namely

$$\frac{f^{n+1} - f^n}{\Delta t} = L(f^{n+1}) + O(\Delta t), \tag{13}$$

where n indicates the time level in the integration procedure, and $L(f)$ is the Fokker–Planck collision operator, which can be regarded as a diffusion operator in velocity space. This formulation is absolutely stable numerically, and time steps are only limited by accuracy considerations. However, a pure implicit formulation of the Fokker–Planck collision operator is not practical due to the non-linear coefficients, which are a function of the Rosenbluth potentials. The simplest integration approach is to implement an iterative procedure on the coefficients until convergence is achieved,

$$\frac{f_k^{n+1} - f^n}{\Delta t} = L\left[f_k^{n+1}, G\left(f_{k-1}^{n+1}\right), H\left(f_{k-1}^{n+1}\right)\right], \tag{14}$$

where $k = 1, 2, \ldots$ is the iteration level. The initial guess in this iteration, $f_{k=0}^{n+1}$, is obtained by solving Eq. (14) with the Rosenbluth potentials determined with the solution at the previous time level, f^n. The discretization of Eq. (14) in velocity space results in a sparse matrix, which does not pose storage problems and can be inverted either with standard solvers or preconditioned Krylov methods. Although exact particle conservation is possible, large energy errors may result from the formulation in Eq. (14) with large time steps [3]. Thus, an alternative, energy-conservative, non-linear implicit time iteration is needed.

3.1. *Energy-Conservative Time Integration Algorithm*

The distribution function at the present time level, f^{n+1}, can be regarded as the solution of the non-linear problem

$$\xi(f) = \frac{f - f^n}{\Delta t} - L(f) = 0. \tag{15}$$

The root can be obtained using the Newton–Raphson iterative technique,

$$\left(\frac{\partial \xi}{\partial f}\right)_k (f_{k+1} - f_k) = -\xi(f_k). \tag{16}$$

Here, k represents the non-linear iteration level. Introducting $\xi(f)$, and noting that $L(f)$ is bilinear on f (the friction and diffusion coefficients are linear operators on f) and hence $(\frac{\partial L}{\partial f})_k f_k = 2L(f_k)$ [3], the following formulation results:

$$\frac{f_{k+1} - f^n}{\Delta t} = \left(\frac{\partial L(f)}{\partial f}\right)_k f_{k+1} - L(f_k). \tag{17}$$

Upon convergence, the solution at the $(n+1)$ time level is obtained by identifying $f^{n+1} = f_{k+1}$. Equation (17) conserves energy in every step of the iteration (i.e., for any k), as is proven next. The energy moment of Eq. (17) is given by

$$\frac{\Delta E^{k+1}}{\Delta t} = \int d\mathbf{v}\frac{v^2}{2}\frac{f_{k+1} - f^n}{\Delta t} = \int d\mathbf{v}\frac{v^2}{2}\left(\frac{\partial L(f)}{\partial f}\right)_k f_{k+1} - \underbrace{\int d\mathbf{v}\frac{v^2}{2}L(f_k)}_{=0}. \tag{18}$$

The last term in this expression represents the energy moment of the Fokker–Planck collision operator; although it cancels theoretically, it does not numerically *unless* an energy-conservative discretization is used [4]. This is assumed in what follows.

By definition of the Jacobian, we can write, for an arbitrary function g,

$$\begin{aligned}\left(\frac{\partial L(f)}{\partial f}\right)_{f_0} g &= \lim_{\epsilon\to 0}\frac{L(f_0 + \epsilon g) - L(f_0)}{\epsilon} \\ &= \Gamma\frac{\partial}{\partial \mathbf{v}}\cdot\frac{\partial}{\partial \mathbf{v}}\cdot\left[2\overline{T}[H(f_0), \delta H(g)] + \frac{1}{2}\frac{\partial^2 G(f_0)}{\partial\mathbf{v}\partial\mathbf{v}}g + \frac{1}{2}\frac{\partial^2[\delta G(g)]}{\partial\mathbf{v}\partial\mathbf{v}}f_0\right],\end{aligned} \tag{19}$$

where $\delta H(g)$ and $\delta G(g)$ are defined as

$$\delta H(g) = \left(\frac{\partial H}{\partial f}\right)_{f_0} g \tag{20}$$

$$\delta G(g) = \left(\frac{\partial G}{\partial f}\right)_{f_0} g. \tag{21}$$

The subscript f_0 means that the Jacobian is calculated at $f = f_0$. In obtaining (19), the bilinear, symmetric nature of $\overline{T}[H, H]$ has been taken into account. With these results,

Eq. (18) simplifies to

$$\frac{\Delta E^{k+1}}{\Delta t} = \Gamma \int d\mathbf{v}\frac{v^2}{2}\frac{\partial}{\partial \mathbf{v}} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \left[2\overline{T}[H(f_k), \delta H(f_{k+1})] + \frac{1}{2}\frac{\partial^2 G(f_k)}{\partial \mathbf{v}\partial \mathbf{v}} f_{k+1} + \frac{1}{2}\frac{\partial^2[\delta G(f_{k+1})]}{\partial \mathbf{v}\partial \mathbf{v}} f_k\right]. \quad (22)$$

This equation remains completely general as to how $H(f)$ and $G(f)$ are defined. In order to find the specific constitutive relations of δH and δG from Eqs. (20) and (21), the definitions of H and G given in Eq. (11) and Eq. (12), respectively, must be utilized. Finding the constitutive relation of $\delta G(g)$ is straightforward from Eq. (12) and yields

$$\nabla_{\mathbf{v}}^2[\delta G(g)] = \delta H(g). \quad (23)$$

However, finding the constitutive relation of $\delta H(g)$ is more involved, since H is the root of the following non-linear functional (Eq. (11)),

$$\kappa[H(f),\ f] = Q[H, H] + \frac{fH}{2} = 0. \quad (24)$$

Differentiating $\kappa[H(f),\ f]$ with respect to f and operating the result on g, yields

$$\frac{d\kappa[H(f),\ f]}{df}g = \frac{\partial \kappa}{\partial H}\frac{\partial H}{\partial f}g + \frac{\partial \kappa}{\partial f}g = \frac{\partial \kappa}{\partial H}\delta H(g) + \frac{\partial \kappa}{\partial f}g = 0, \quad (25)$$

where Eq. (20) has been used. The partial derivatives yield

$$\frac{\partial \kappa}{\partial H}\delta H = Q[H, \delta H] + Q[\delta H, H] + \frac{f\delta H}{2} \quad (26)$$

$$\frac{\partial \kappa}{\partial f}g = \frac{gH}{2} \quad (27)$$

and, hence, the constitutive relation for $\delta H(g)$ is found,

$$Q[H(f), \delta H(g)] + Q[\delta H(g), H(f)] = -\frac{H(f)g + \delta H(g)f}{2}. \quad (28)$$

For consistency, Eq. (28) has to revert back to Eq. (11) when $g \equiv f$. This is indeed the case since $H(f)$ is a homogeneous function of the first degree [i.e., $H(af) = aH(f)$], and hence

$$\delta H(g \equiv f) = \left(\frac{\partial H}{\partial f}\right)_f f = \lim_{\epsilon \to 0}\frac{H[(1+\epsilon)f] - H(f)}{\epsilon} = H(f).$$

Equation (11) is regained upon substitution of this result in Eq. (28).

At this point, the same integrations described in Eqs. (7)–(8) are performed on Eq. (22) to obtain

$$\frac{\Delta E^{k+1}}{\Delta t} \propto \int d\mathbf{v}\mathbf{v} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \overline{T}[H(f_k), \delta H(f_{k+1})] - \int d\mathbf{v}\frac{H(f_k) f_{k+1} + \delta H(f_{k+1}) f_k}{4}$$

which, by Eq. (28), can be written as

$$\frac{\Delta E^{k+1}}{\Delta t} \propto \int d\mathbf{v}\left[\mathbf{v}\cdot\frac{\partial}{\partial \mathbf{v}}\cdot \bar{\bar{T}}[H(f_k), \delta H(f_{k+1})] + \frac{Q[H(f_k), \delta H(f_{k+1})] + Q[\delta H(f_{k+1}), H(f_k)]}{2}\right] = 0. \tag{29}$$

This result cancels by virtue of Eq. (9), true for *any* function H regular at infinity, and hence true for $H + \delta H$,

$$\int d\mathbf{v}\left[\mathbf{v}\cdot\frac{\partial}{\partial \mathbf{v}}\cdot \bar{\bar{T}}[H+\delta H, H+\delta H] + Q[H+\delta H, H+\delta H]\right] = 0 \tag{30}$$

and by virtue of the bilinear nature of both $\bar{\bar{T}}[H, H]$ (which is symmetric) and $Q[H, H]$,

$$\bar{\bar{T}}[H+\delta H, H+\delta H] = \bar{\bar{T}}[H, H] + 2\bar{\bar{T}}[H, \delta H] + \bar{\bar{T}}[\delta H, \delta H]$$
$$Q[H+\delta H, H+\delta H] = Q[H, H] + Q[H, \delta H] + Q[\delta H, H] + Q[\delta H, \delta H].$$

Introducing these expressions back in Eq. (30), the terms in $[H, H]$ and $[\delta H, \delta H]$ cancel by Eq. (9), leading to the result in Eq. (29). This result implies that energy is conserved in any step of Newton's iteration (i.e., for any k), provided that Eq. (30) is ensured by using an energy-conservative discretization of the Fokker–Planck collision operator [4]. Hence, upon convergence, $\Delta E^{n+1}/\Delta t = 0$.

Thus, the implicit time discretization is only conservative if coupled with Newton's iterative technique. Epperlein [3] was the first to realize the properties of such a combination when he considered a linearized form of the Fokker–Planck collision operator (i.e., a single Newton iteration) and arrived at a similar result for a spherically symmetric, one-dimensional velocity space. Equation (29) proves this in a more general fashion, for every step in Newton's method, and for any geometry and dimensionality in velocity space.

Convergence in Newton's technique is fast, provided the initial guess is within the radius of convergence. In transient problems, the radius of convergence depends strongly on the time step used in the integration. To ensure convergence irrespectively of the magnitude of the time step, an adaptive time step scheme has been implemented.

3.2. *Adaptive Time Step Scheme*

The convergence of Newton's iterative technique for a given problem is extremely dependent on the initial condition. In particular, Newton's method will have a quadratic convergence rate if the initial guess for the solution falls into the radius of convergence. However, it will take far longer—and even diverge—if the initial guess falls outside of this radius. In the non-linear problem in Eq. (15) (for which the initial guess is the solution at the previous time level), the radius of convergence is typically a strong function of the time step, Δt. Intuitively, in situations far from equilibrium, the larger the implicit time step Δt is, the more different the solutions at successive time levels are. In some cases, there is a threshold in Δt above which the updated solution of the distribution function becomes negative.

In order to prevent negative solutions, a time correction scheme has been devised to damp the Newton update, so that Newton's method converges to a physical solution in almost any situation, regardless of the size of Δt. The essence of the technique is to introduce a time gauge γ that determines if Δt is too large, and, if so, provides an adequate value of Δt

to proceed with the first Newton iterations. Then, as the iteration procedure advances, the time step is slowly corrected towards the original value of Δt. In this way, each step in the iteration procedure starts with an initial guess that falls within the radius of convergence of Newton's algorithm.

The artificial time gauge γ is introduced in the problem by subtracting $\frac{f}{\gamma}$ from the right and left hand sides of the original problem, as

$$\frac{\partial f}{\partial t} - \frac{f}{\gamma} = L(f) - \frac{f}{\gamma}.$$

The left hand side can be grouped using the integrating factor technique to give

$$\frac{\partial}{\partial t}\left[e^{-\frac{t}{\gamma}} f\right] = e^{-\frac{t}{\gamma}}\left[L(f) - \frac{f}{\gamma}\right].$$

Integrating this expression from t^n to $t^{n+1} = t^n + \Delta t$ yields

$$e^{-\frac{t^{n+1}}{\gamma}} f^{n+1} - e^{-\frac{t^n}{\gamma}} f^n = \int_{t^n}^{t^{n+1}} e^{-\frac{t}{\gamma}}\left[L(f) - \frac{f}{\gamma}\right]. \tag{31}$$

The integral can be approximated by

$$\int_{t^n}^{t^{n+1}} e^{-\frac{t}{\gamma}}\left[L(f) - \frac{f}{\gamma}\right] \approx -\left[L(f^{n+1}) - \frac{f^{n+1}}{\gamma}\right]\gamma\left(e^{-\frac{t^{n+1}}{\gamma}} - e^{-\frac{t^n}{\gamma}}\right) + O(\Delta t)$$

which leads, upon substitution in Eq. (31), to the following modified first order implicit discretization:

$$\frac{f^{n+1} - f^n}{\eta} = L(f^{n+1}). \tag{32}$$

Here, η is the modified time step, $\eta = \gamma(1 - e^{-\Delta t/\gamma})$, and has the limits

$$\Delta t \ll \gamma \Rightarrow \eta \to \Delta t$$
$$\Delta t \gg \gamma \Rightarrow \eta \to \gamma.$$

Hence, η is limited by γ when Δt is too large and falls back to the original Δt if it is small compared to γ.

Obviously, the effectiveness of Eq. (32) in avoiding unphysical results during Newton's iteration will very much depend on the wisdom in choosing the time gauge, γ. In this work, γ is chosen as half the time step that would render a negative distribution function in an explicit scheme,

$$\gamma = \frac{1}{2}\frac{\max_{i,j}\left(f_{i,j}^n\right)}{\max_{i,j}\left[-(\partial f/\partial t)_{i,j}^n = -L_{i,j}(f^n)\right]}.$$

Note that the expression above selects the time scales corresponding to $L(f^n) < 0$. Also, although the two maxima may not occur exactly for the same (i, j) node, they will be very close since, in a diffusion problem, the largest negative rate of change usually corresponds to the peak of the distribution. This selection has proven effective in actual simulations (Subsection 5.2).

The practical implementation of the adaptive time step scheme is as follows:

- First, the modified time step $\eta = \eta_0$ is determined as indicated above. If $\eta_0 > 0.8\Delta t$, then Δt is used directly, without any time correction, until convergence is achieved.
- If $\eta_0 < 0.8\Delta t$, then η_0 is used as the time step for the first Newton iteration.
- In every subsequent Newton iteration, η in Eq. (32) is set to $\eta = k\eta_0$, where k is the iteration number, until $\eta > 0.8\Delta t$.
- At this point, η is set to Δt, and the Newton–Raphson algorithm proceeds until convergence is achieved.

4. THE SOLVER ENGINE

Figure 2 presents a flow chart of the Fokker–Planck solver algorithm discussed in the previous sections, with the crucial differential equations to be solved numerically. According to this diagram, it is clear that the energy-conservative Fokker–Planck solver has to deal efficiently with three different classes of algebraic problems, namely:

1. Non-symmetric dense systems, stemming from the discretization in velocity space of Eq. (17). The Jacobian matrix $(\partial L/\partial f)_k$ is non-symmetric and dense because the friction and diffusion coefficients are integral expressions of f. The solution of this system yields the updated solution of the distribution function f^{n+1}.
2. Non-symmetric sparse systems, stemming from the Newton iterative treatment of the non-linear constitutive relation of H and δH (Eqs. (11) and (28)).
3. SPD sparse systems, stemming from the constitutive relations of G and δG (Eqs. (12) and (23)).

If standard solvers (such as direct solvers or stationary iterative techniques) were to be used in this context, the problem simply could not be handled due to prohibitive storage requirements and due to the inefficiency of these solvers for large, non-symmetric matrices.

Krylov iterative methods, however, are very well suited for this task because of their improved efficiency over standard solvers and because they can be implemented Jacobian-free [6, 7], i.e., without ever forming (and storing) the Jacobian matrix. The better efficiency of these methods in multidimensional problems can be appreciated in Table I, where the

TABLE I
Computational Complexities of Various Solvers for the Poisson Equation in Cartesian Coordinates

Inversion method	1D	2D	3D
Direct solver	$O(N)$	$O(N^2)$	$O(N^{7/3})$
Jacobi/Gauss-Seidel	$O(N^3)$	$O(N^2)$	$O(N^{5/3})$
SOR with optimal ω	$O(N^2)$	$O(N^{3/2})$	$O(N^{4/3})$
Conjugate Gradient (CG)	$O(N^2)$	$O(N^{3/2})$	$O(N^{4/3})$
Preconditioned CG (w/SOR)	$O(N^{3/2})$	$O(N^{5/4})$	$O(N^{7/6})$
Multigrid	$O(N)$	$O(N)$	$O(N)$

Note. Orderings are valid as the number of unknowns $N \to \infty$ and have been obtained based on results from Ref. [19] (for direct solvers, J/GS/SOR, and CG/PCG) and Ref. [18] (for MG).

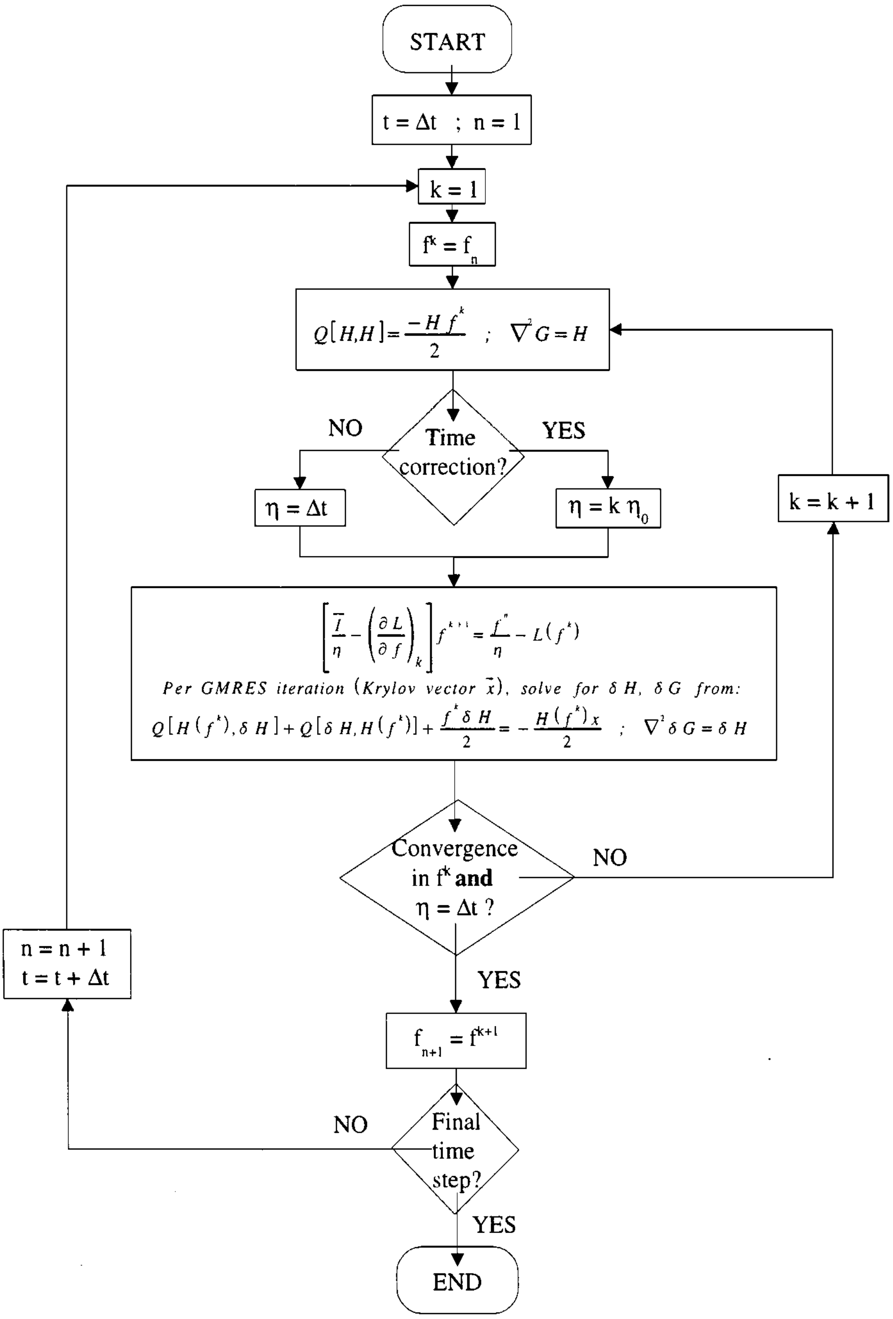

START
$t = \Delta t \; ; \; n = 1$
$k = 1$
$f^k = f_n$
$Q[H,H] = \frac{-H f^k}{2} \; ; \; \nabla^2 G = H$
Time correction?
NO
YES
$\eta = \Delta t$
$\eta = k\,\eta_0$
$k = k + 1$
$\left[\frac{\bar{I}}{\eta} - \left(\frac{\partial L}{\partial f}\right)_k\right] f^{k+1} = \frac{f^n}{\eta} - L(f^k)$
Per GMRES iteration (Krylov vector $\vec{x}$), solve for δH, δG from:
$Q[H(f^k), \delta H] + Q[\delta H, H(f^k)] + \frac{f^k \delta H}{2} = -\frac{H(f^k)x}{2} \; ; \; \nabla^2 \delta G = \delta H$
Convergence in f^k and $\eta = \Delta t$?
NO
YES
$n = n + 1$
$t = t + \Delta t$
$f_{n+1} = f^{k+1}$
Final time step?
NO
YES
END

FIG. 2. Flow-chart of the energy conservative algorithm, with the details of all the algebraic problems that need to be solved, as well as the integration of the time adaptive scheme.

computational complexity (number of operations) of the Krylov CG technique (with and without preconditioning) is compared against that of standard iterative techniques and multigrid (MG) methods for the reference case of the laplacian operator in Cartesian coordinates in one, two, and three dimensions. The comparison indicates that the least efficient form of CG (i.e., without preconditioning and without any external parameter) is already as efficient as the successive over-relaxation (SOR) technique with optimal overrelaxation parameter ω. Furthermore, the CG's efficiency is significantly boosted by *preconditioning* the problem (in this case, using approximate factorization), almost matching that of MG methods.

Although, according to Table I, MG is the most efficient alternative for the Laplacian reference case, Krylov techniques are preferred for the non-linear application of interest here because MG cannot be implemented Jacobian-free. Thus, a MG implementation would require forming and storing the Jacobian matrix for each Newton iteration, eliminating the advantage. Nevertheless, the MG superior convergence properties are incorporated in the Jacobian-free Krylov solver via the preconditioning step. The next sections explain these concepts further. The implementation of these techniques in each of the different algebraic problems enumerated above will also be discussed in detail.

4.1. *Introduction to Krylov Methods*

Krylov iterative algorithms [5] belong to the family of semi-iterative conjugate methods. The term "semi-iterative" indicates that, while these techniques are theoretically exact in as many iterations as the range of the matrix, they provide very good estimates much sooner. This convergence property relates to the fact that these methods minimize the residual between the exact and the approximate solutions at every iteration.

Krylov algorithms are conjugate in that they use conjugate vectors to solve the system. Two vectors $\mathbf{d}_i$, $\mathbf{d}_j$ are said to be conjugate with respect to a non-singular matrix A if $\mathbf{d}_i \cdot A \cdot \mathbf{d}_j = 0, i \neq j$. If a *complete* basis of conjugate vectors $\{\mathbf{d}_i\}_{i=1}^N$ is known for the matrix A, the solution to the linear system $A\mathbf{x} = \mathbf{b}$ can be trivially found in the following way,

$$\mathbf{x} = \sum_i y_i \mathbf{d}_i \Rightarrow \mathbf{d}_j \cdot A \cdot \mathbf{x} = \sum_i y_i \mathbf{d}_j \cdot A \cdot \mathbf{d}_i = \mathbf{d}_j \cdot \mathbf{b} \Rightarrow y_j = \frac{\mathbf{d}_j \cdot \mathbf{b}}{\mathbf{d}_j \cdot A \cdot \mathbf{d}_j}. \tag{33}$$

The task of the Krylov algorithms is to build this basis as the iteration proceeds. This is done by orthogonalizing the Krylov subspace $\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\}$, where k is the iteration number ($1 \leq k \leq N$), and $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ is the first residual ($\mathbf{x}_0$ is the initial guess). Once the orthogonal subspace is formed, the coefficients y_j are obtained from Eq. (33), and the approximate $\mathbf{x}$ is tested for convergence. This is done iteratively, until the specified convergence criterion on the residual is met. This process can be done very efficiently for SPD matrices (CG [12]), because the orthogonalization of the Krylov subspace can be done from a recurrence relation involving only the last two Krylov vectors found. Furthermore, because CG minimizes the functional $E(\mathbf{y}) = \frac{1}{2}(\mathbf{y}, A\mathbf{y}) - (\mathbf{y}, \mathbf{b})$ (which for SPD matrices has a unique minimum at $\mathbf{y} = \mathbf{x}$), it guarantees convergence in as many iterations as the range of the system.

However, for non-symmetric non-definite matrices the situation is more complex, because the orthogonalization process involves multiple Krylov vectors, and $E(\mathbf{y})$ is not necessarily minimal for $\mathbf{y} = \mathbf{x}$. Among the available Krylov techniques for non-symmetric systems [16], GMRES [13] is chosen. In GMRES, the minimization is done directly over the residual by obtaining the $\{y_j\}$ coefficients from the corresponding least squares problem; hence, convergence is guaranteed. In addition, GMRES is very robust in numerical Jacobian-free applications [7]. The downside of GMRES is that the iteration procedure requires storing *all* the previous Krylov vectors found, which may result in large storage requirements unless the iterative procedure converges rapidly. Thus, to avoid large storage requirements, the number of GMRES iterations required for convergence must be minimal, and an effective preconditioning scheme is required.

4.2. *Jacobian-Free Implementation of Krylov Methods: The Newton–Krylov Technique*

As opposed to stationary iterative techniques, no matrix splitting is needed in Krylov iterative algorithms to proceed with the iteration, and all that is required is the product of the matrix of coefficients times a vector **x**, dictated by the iterative algorithm. This property is of particular relevance when the linear system stems from Newton's method, as the matrix-vector product involves a Jacobian matrix, and it can be expressed as

$$\left(\frac{\partial \mathbf{\Lambda}}{\partial \mathbf{y}}\right)_{\mathbf{y}_0} \cdot \mathbf{x} = \lim_{\epsilon \to 0} \frac{\mathbf{\Lambda}(\mathbf{y}_0 + \epsilon \mathbf{x}) - \mathbf{\Lambda}(\mathbf{y}_0)}{\epsilon}. \tag{34}$$

This equation indicates that it is possible to calculate the matrix-vector product without ever forming the Jacobian matrix (hence, the name Jacobian-free). This method is usually referred to as the Newton–Krylov Jacobian-free technique.

In this particular implementation, the limit in Eq. (34) can be calculated theoretically (Eq. (39) in Subsection 4.4). It is important to bear in mind, however, that in cases where this limit can only be calculated numerically (using a small—but non-zero—ϵ), the accuracy of the algorithm is limited by the error introduced in the numerical evaluation, whose leading term is proportional to $(\epsilon |\mathbf{x}|^2)$. Hence, a Krylov algorithm that renders orthonormal (not just orthogonal) Krylov vectors (such as GMRES) is essential to preserve accuracy and ensure convergence.

4.3. *The Preconditioning Step*

The preconditioning step can be conceptually viewed as acting on the matrix of coefficients A with an operator P^{-1} (the preconditioner) such that either $[P^{-1}A]$ (left preconditioning) or $[AP^{-1}]$ (right preconditioning) is sufficiently close to the identity matrix. In Jacobian-free applications, right preconditioning is preferred because it can be naturally incorporated into the Jacobian-free product as

$$\left(\frac{\partial \mathbf{\Lambda}}{\partial \mathbf{y}}\right)_{\mathbf{y}_0} \cdot \underbrace{P^{-1} \cdot \mathbf{x}}_{\mathbf{z}} = \lim_{\epsilon \to 0} \frac{\mathbf{\Lambda}(\mathbf{y}_0 + \epsilon\, \overbrace{P^{-1} \cdot \mathbf{x})}^{\mathbf{z}} - \mathbf{\Lambda}(\mathbf{y}_0)}{\epsilon}. \tag{35}$$

As Krylov techniques only require the product of the system matrix times a vector to proceed, the preconditioning step can be implemented in the algorithm very straightforwardly and usually boils down to solving $P \cdot \mathbf{z} = \mathbf{x}$ [17], where **z** is the unknown, and **x** is the Krylov vector dictated by the algorithm. Preconditioned GMRES is also able to return orthonormal Krylov vectors, because the Krylov vectors in this case are obtained form the Krylov subspace $\{\mathbf{r}_0, A_p\, \mathbf{r}_0, A_p^2 \mathbf{r}_0, \ldots, A_p^{k-1}\mathbf{r}_0\}$, where $A_p = AP^{-1}$ for right preconditioning, and $A_p = P^{-1}$ A for left preconditioning.

The preconditioning matrix P has to contain significant information about the eigenvalue spectra of the original system matrix for the preconditioning step to be effective. This means that P may carry some of the ill-conditioned features of the original matrix—if any—and that the difficulty in inverting the original matrix will be present in the inversion of P

as well, thus defeating the original purpose of preconditioning, namely, accelerating the convergence in an inexpensive way. Fortunately, in practice, a reasonable approximation to **z** will suffice for this purpose. Thus, preconditioning schemes [8] are based on "simple" ways of obtaining this approximation, such as stationary iterative techniques (Jacobi, Gauss–Seidel, SOR), incomplete Cholesky decompositions, or approximate multigrid methods, to name some. The disadvantage of using such simple techniques is that their potential of acceleration is limited by the same boundaries that limit their use as stand-alone solvers. This is particularly important in the case of preconditioners based on stationary iterative techniques, in which the number of iterations presents a power scaling with the number of mesh points that renders them very inefficient when high resolution is required. For the 2D Laplacian operator, this scaling is $O(N)$ for Jacobi or Gauss–Seidel, and $O(N^{1/2})$ for the optimized SOR method. This power scaling is a direct consequence of the fact that stationary iterative techniques are successful in damping oscillatory harmonics of the initial residual, $\mathbf{e}_r = \mathbf{x} - P \cdot \mathbf{z}_0$, but are very inefficient in removing the smooth modes, a task to which the iterative method devotes most of its effort. For this reason, stationary iterative techniques are also called "smoothers."

Multigrid preconditioners [10, 11] use the principles of MG techniques to deal with this limitation. These techniques combine the smoothing property of stationary iterative techniques with a suitable grid coarsening ("restriction") algorithm, on the grounds that smooth modes look oscillatory in a coarsened mesh. Thus, successive restriction steps, followed by smoothing steps (to remove the oscillatory modes in the coarsened mesh), will be very effective in removing the smooth modes from the initial residual $\mathbf{e}_r$. This procedure is performed recursively to a point where a direct solution is efficient, and then this exact solution is extrapolated back ("prolongation") through the mesh ladder, with a smoother step between successive extrapolations, up to the original mesh. This is the simplest cycle (so-called "V-cycle"), and it is not difficult to envision more complex cycles stemming from the combination of partial or complete V-cycles, intermixing restriction and prolongation steps as desired (thus resulting in W-cycles, and so on [18]).

In stand-alone MG solvers, the adequate engineering of the restriction and prolongation algorithms is essential to preserve the accuracy and efficiency of the solution. However, as preconditioners (i.e., accelerators of convergence of an already accurate solver), a decent approximation to the actual solution is often enough to speed convergence. Thus, restriction and prolongation operators constructed from simple interpolation algorithms will suffice to greatly improve the Krylov rate of convergence. Moreover, since MG solvers damp all the modes at similar rates, as preconditioners they will render an almost constant number of iterations of the solver (here, CG and GMRES) with the mesh refinement.

As mentioned earlier, it is impractical to use MG methods as stand-alone solvers to deal with the full algebraic system, because they cannot be implemented Jacobian-free. However, as preconditioners, they only have to deal with a sparse approximation of the exact system matrix, which can be formed and stored easily. In this paper, the preconditioning matrix is obtained from the sparse representation of the Fokker–Planck equation outlined in Eq. (14). Results on the performance of this preconditioner are presented in Subsection 5.1.

4.4. *Implementation Details of the Energy-Conservative Solver*

As stated earlier, the development of an energy-conservative solver involves dealing with three different problems, namely, symmetric sparse, non-symmetric sparse, and non-symmetric dense. In the following sections, the implementation of Krylov methods for each of these problems is discussed in detail.

To focus the discussion that follows, a 2D cylindrical velocity space with angular symmetry is adopted. This space is spanned by (v_r, v_p), where v_r is the cylindrical z-axis, and v_p is the cylindrical r-axis (Fig. 1), and $v_r \in [0, v_{limit}]$; $v_p \in [0, v_{limit}]$. Here, v_{limit} is typically set to several times the characteristic velocity of the problem, v_0. The domain is discretized with an integer mesh and a half mesh [4].

4.4.1. Fokker–Planck non-linear system. Once the Fokker–Planck collision operator is discretized in a two-dimensional velocity space (i.e., $L(f)$ is transformed into $\Lambda_{i,j}(f_{l,m})$, according to techniques presented in Ref. [4]), the energy-conservative time discretization scheme in Eq. (17) becomes the following algebraic equation,

$$\left[\frac{\delta_{i,j,l,m}}{\eta} - \left(\frac{\partial \Lambda_{i,j}}{\partial f_{l,m}}\right)_k\right] f_{l,m}^{k+1} = \frac{f_{i,j}^n}{\eta} - \Lambda_{i,j}\left(f_{l,m}^k\right), \tag{36}$$

where η is the (corrected) time step, n is the time level index, and k in the non-linear Newton iteration level. Here, $\delta_{i,j,l,m}$ represents the unitary tensor (Krönecker delta). Define $q = i + N_r(j-1)$, $s = l + N_r(m-1)$ (thus allocating all the mesh points of a two-dimensional mesh in a single vector) to cast Eq. (36) in the standard form of a linear system of equations, as

$$\left[\frac{\delta_{q,s}}{\eta} - \left(\frac{\partial \Lambda_q}{\partial f_s}\right)_k\right] f_s^{k+1} = \frac{f_q^n}{\eta} - \Lambda_q\left(f_s^k\right). \tag{37}$$

In this equation, the matrix-vector product required to solve the system is

$$\left[\frac{\delta_{q,s}}{\eta} - \left(\frac{\partial \Lambda_q}{\partial f_s}\right)\right] x_s = \left[\frac{\overline{I}}{\eta} - \left(\frac{\partial \mathbf{\Lambda}}{\partial \mathbf{f}}\right)\right] \cdot \mathbf{x} = \frac{\mathbf{x}}{\eta} - \left(\frac{\partial \mathbf{\Lambda}}{\partial \mathbf{f}}\right) \cdot \mathbf{x}, \tag{38}$$

where $\overline{I}$ is the identity dyadic, and $\partial\mathbf{\Lambda}/\partial\mathbf{f}$ is a Jacobian matrix, which is non-symmetric and dense (because both the friction and diffusion coefficients are integral expressions of f). Then, it is possible to calculate the matrix-vector product in Eq. (38) using the Jacobian-free techniques introduced in Subsection 4.2. This is in fact the crucial element that allows the development of a competitive energy-conservative Fokker–Planck solver, since forming and storing the Jacobian matrix would be prohibitive.

As the Fokker–Planck collision operator is bilinear on f, the derivative in Eq. (34) can be calculated theoretically (Eq. (19)), as

$$\left.\frac{\partial \Lambda_q}{\partial f_s}\right|_k x_s = \Gamma\left\{\frac{\partial}{\partial \mathbf{v}} \cdot \frac{\partial}{\partial \mathbf{v}} \cdot \left[2\overline{T}\left[H\left(f_s^k\right), \delta H(x_s)\right] + \frac{1}{2}\frac{\partial^2 G\left(f_s^k\right)}{\partial \mathbf{v}\partial \mathbf{v}} x_s \right.\right.$$
$$\left.\left. + \frac{1}{2}\frac{\partial^2[\delta G(x_s)]}{\partial \mathbf{v}\partial \mathbf{v}} f_s^k\right]\right\}_{q=i+N_r(j-1)}. \tag{39}$$

The operator within curly brackets is to be discretized at the (i, j) node in exactly the same way as the Fokker–Planck operator itself, using an energy-conservative difference scheme [4]. A disadvantage of the Jacobian-free product in Eq. (39) is that it requires solving for $\delta H(x_s)$ and $\delta G(x_s)$ (i.e., two more algebraic problems) in every GMRES iteration. (Alternatively, one could use the outer Newton–GMRES iteration to solve the couled problem

$\{f, G, H\}$; this would simplify each GMRES iteration at the cost of increasing the dimension of each GMRES vector by a factor of 3.) Therefore, limiting the number of iterations is crucial, not only to alleviate the storage requirements of GMRES, but also to preserve the efficiency of the algorithm.

The preconditioning matrix P used for this problem is the 9-banded sparse matrix stemming from a second-order discretization in velocity space of the Fokker–Planck collision operator in the iterative formulation presented in Eq. (14), with the Rosenbluth potentials lagged to the previous time level. This matrix is stored in diagonal sparse format. The true Jacobian matrix is never formed.

4.4.2. Linear Poisson systems. The energy-conservative solver requires that both the G Rosenbluth potential and δG be determined from the following partial differential equation (Eqs. (12) and (23), respectively),

$$\nabla^2 Y = \frac{1}{v_p}(v_p Y_p)_p + Y_{rr} = S, \tag{40}$$

where $Y = \{G, \delta G\}$ and $S = \{H, \delta H\}$, respectively. Here, the subscript p indicates $\partial/\partial v_p$, and the subscript r indicates $\partial/\partial v_r$. This PDE is discretized using second order (centered) finite differences in a 5-point stencil, as

$$\frac{1}{v_p}(v_p Y_p)_p = \frac{v_{p,j+\frac{1}{2}}\left((Y_{i,j+1} - Y_{i,j})/\Delta v_{p,j+\frac{1}{2}}\right) - v_{p,j-\frac{1}{2}}\left((Y_{i,j} - Y_{i,j-1})/\Delta v_{p,j-\frac{1}{2}}\right)}{v_{p,j\Delta v_{p,j}}}$$

$$Y_{rr} = \frac{\left((Y_{i+1,j} - Y_{i,j})/\Delta v_{r,i+\frac{1}{2}}\right) - \left((Y_{i,j} - Y_{i-1,j})/\Delta v_{r,i-\frac{1}{2}}\right)}{\Delta v_{r,i}}.$$

Velocity increments are defined in Ref. [4]. As $v_p \to 0$, the term $(1/v_p)(v_p Y_p)_p \to 2Y_{pp}$ by L'hospital's rule, and is discretized with symmetric boundary conditions (i.e., $Y_{i,j+1} = Y_{i,j-1}$ at $j = 1$) as

$$Y_{pp i,j=1} = 2\frac{Y_{i,j+1} - Y_{i,j}}{\Delta v_{p,j}\Delta v_{p,j+\frac{1}{2}}}\bigg|_{i,j=1}.$$

Such discretization transforms the differential operator in Eq. (40) in a $N \times N$ non-symmetric 5-banded diagonal sparse matrix (with $N = N_r \times N_p$, where N_r and N_p are the number of mesh points in the v_r and v_p directions, respectively).

However, Krylov methods are faster and more efficient for symmetric matrices (CG can be used instead of GMRES). The non-symmetry in the previous matrix is due to the $1/v_p$ coefficient, which can be removed by multiplying both sides of the discretized form of Eq. (40) by the volume element in velocity space (given by $\Delta\Omega = 2\pi v_{p,j}\Delta v_{p,j}\Delta v_{r,i}$ for a general mesh point; specialized expressions are needed for the boundaries and corner points [4]). The result of this operation reads

$$\Delta v_{r,i}\left[v_{p,j+\frac{1}{2}}\frac{Y_{i,j+1} - Y_{i,j}}{\Delta v_{p,j+\frac{1}{2}}} - v_{p,j-\frac{1}{2}}\frac{Y_{i,j} - Y_{i,j-1}}{\Delta v_{p,j-\frac{1}{2}}}\right]$$
$$+ v_{p,j}\Delta v_{p,j}\left[\frac{Y_{i+1,j} - Y_{i,j}}{\Delta v_{r,i+\frac{1}{2}}} - \frac{Y_{i,j} - Y_{i-1,j}}{\Delta v_{r,i-\frac{1}{2}}}\right] = v_{p,j}\Delta v_{p,j}\Delta v_{r,i}S_{i,j}.$$

Clearly, the $1/v_p$ coefficient has disappeared, and the resulting 5-banded sparse matrix is rendered symmetric positive definite (SPD). Hence, CG is used as the solver driver. This matrix is stored in diagonal sparse format. The preconditioning matrix is the same as the system matrix.

4.4.3. Non-linear Poisson system. The problem for the H Rosenbluth potential is more involved, since the following non-linear PDE [4] has to be solved numerically:

$$Q[H,H] = \frac{1}{16\pi}\left[(H_r H)_r - H_r^2 + 2\frac{H}{v_p}(v_p H_p)_p + H_p^2 - \frac{1}{v_p}(v_p H_p H)_p\right] = -\frac{fH}{2}.$$

This equation has to be solved iteratively by Newton's method, applied to the following non-linear functional (recall Eq. (24)):

$$\kappa[H(f)] = Q[H,H] + \frac{fH}{2}.$$

The root of this function solves the non-linear Poisson problem above. Here, H is the only unknown (f is known). Then, the Newton iterative method reads

$$\left.\frac{\partial\kappa}{\partial H}\right|_{H^k} \Delta H = -\kappa(H^k), \tag{41}$$

where $\Delta H = H^{k+1} - H^k$, and k represents the non-linear iteration level. Hence, the solution of the H problem requires a Krylov solve (to invert the linear algebraic system that stems from the discretization of Eq. (41)) within each Newton iteration.

Since the linear algebraic system stems from Newton's method, the matrix-vector product in the Krylov iteration of Eq. (41) can be performed using Jacobian-free techniques, and yields, for a generic function g,

$$\begin{aligned}\left.\frac{\partial\kappa}{\partial H}\right|_{H^k} g &= Q[H^k, g] + Q[g, H^k] + \frac{fg}{2}\\ &= \frac{1}{16\pi}\left[\left(H_r^k g\right)_r - H_r^k g_r + 2\frac{H^k}{v_p}(v_p g_p)_p + H_p^k g_p - \frac{1}{v_p}\left(v_p H_p^k g\right)_p\right]\\ &\quad + \frac{1}{16\pi}\left[\left(g_r H^k\right)_r - H_r^k g_r + 2\frac{g}{v_p}\left(v_p H_p^k\right)_p + H_p^k g_p - \frac{1}{v_p}\left(v_p g_p H^k\right)_p\right] + \frac{fg}{2}.\end{aligned} \tag{42}$$

This equation is to be discretized with second order (centered) finite differences. As opposed to the linear problem, this non-linear problem cannot be symmetrized. Hence, GMRES has to be used as the Krylov driver. The exact Jacobian matrix (which is a 5-banded diagonal matrix) is used to precondition the system.

Minimizing the number of iterations in this non-linear problem is crucial for the efficiency of the solver, in terms of both memory and CPU. Since inverting a linear, symmetric problem is preferable (because CG is used), the linear H problem is solved first (in the same way as the G problem) to provide an accurate initial guess for the Newton iteration (the solutions of the non-linear H problem and the linear H problem differ only by truncation errors [4]). With this initial guess, Newton's algorithm typically converges in less than five iterations despite stringent convergence tolerances ($\|\boldsymbol{\kappa}(H^k)\|_2 < 10^{-9}$, where $\boldsymbol{\kappa}$ is the vector of residuals in the kth Newton iteration, Eq. (41)).

4.4.4. Constitutive equation for δH. The constitutive relation for δH is given in Eq. (28), reproduced as

$$Q[H(f), \delta H(g)] + Q[\delta H(g), H(f)] = -\frac{H(f)g + \delta H(g) f}{2},$$

were $\delta H(g) = (\frac{\partial H}{\partial f})_f g$. Using Eq. (26), this equation can be rearranged as

$$\left.\frac{\partial \kappa}{\partial H}\right|_f \delta H(g) = -\frac{H(f)g}{2}. \tag{43}$$

This equation is formally identical to the Newton iteration step in Eq. (41). Hence, the same techniques are used for its inversion.

5. RESULTS

The motivation of this work is the development of an efficient, robust, energy-conservative Fokker–Planck solver. The previous discussion shows that this is theoretically possible and implies that it is numerically tractable if MG-preconditioned Krylov techniques are employed. This section discusses the actual performance and limitations of the solver.

In order to test this formulation under extreme conditions, two distinct initial distribution functions will be considered for the numerical experiments:

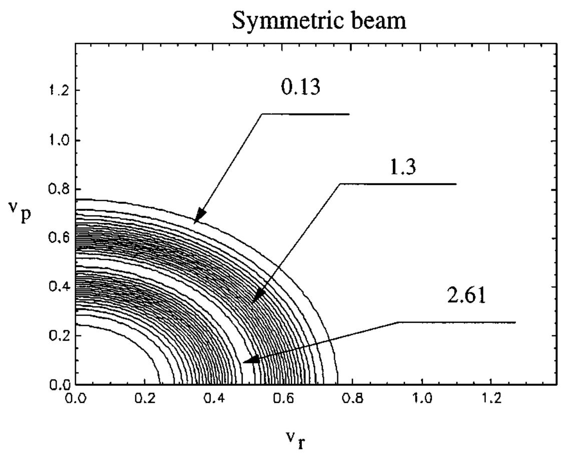
1. A *radial beam,* characterized by a distribution function $f(v_r, v_p)$ with strong angular dependence. The beam is centered on $v_r = 0.5$ and $v_p = 0$, with beam temperature $T_b = 8.89 \cdot 10^{-3}$ and average energy $\langle E\rangle = 0.138$.
2. A *symmetric beam,* with no angular dependence [i.e., $f(v_r, v_p) = f(v)$]. The beam is centered on $v = \sqrt{v_r^2 + v_p^2} = 0.5$, with temperature $T_b = 8.98 \cdot 10^{-3}$ and average energy $\langle E\rangle = 0.147$.

Velocities are in units of an arbitrary reference velocity, v_0; energies are in units of v_0^2. Both beams are localized in the uniformly discretized velocity subdomain and are depicted in Fig. 3.

5.1. *Effectiveness of the MG Preconditioner*

The efficiency of the solver largely depends on the efficiency of the iterative inversion technique. Table I shows that unpreconditioned Krylov techniques are already as efficient as the most efficient of alternative methods for multidimensional problems. Preconditioning improves performance further, but the degree of improvement very much depends on the particular choice of the preconditioning technique.

MG preconditioning is chosen for this application and, with the exception of the smoother, we employ the simple MG method developed in [11]. Piecewise constant interpolation is used for the restriction and prolongation steps. The mesh coarsening factor is 2. Each preconditioning call performs two consecutive V-cycles of $(r-2)$ levels. Not more than $(r-2)$ levels in a $2^r \times 2^r$ mesh may be considered for the restriction step, to ensure that a nine-point stencil (3×3) required to discretize the Fokker–Planck collision operator is contained in the coarsest mesh. Instead of solving the problem exactly at the coarsest level, an approximate solution is found with the smoother. The smoother consists of five passes of the symmetric Gauss–Seidel (SGS) iterative scheme, which achieves a symmetric iteration matrix by performing a forward and a backward pass per iteration step. This is



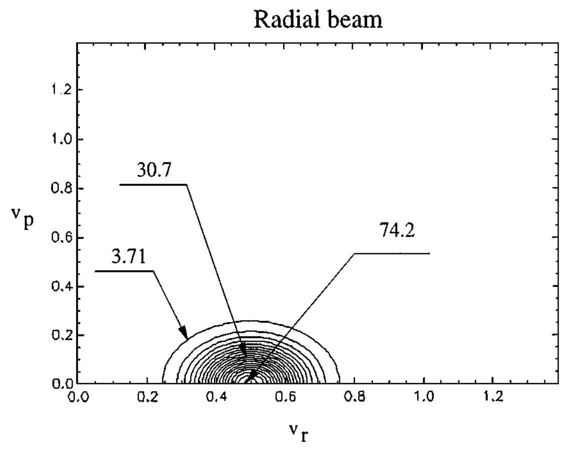


FIG. 3. Initial distribution functions employed in the assessment of the properties of the solver. Velocity units are arbitrary. Some reference values of the distribution function at selected contours are indicated; contours are equally spaced.

beneficial for non-symmetric systems (since it tends to symmetrize the Gauss–Seidel iteration, hence improving convergence [17]) and is crucial to grant a SPD preconditioner in CG. The SGS technique will also be implemented as a stand-alone preconditioner (consisting of ten SGS iterations), to gauge the effectiveness of the MG preconditioner.

The numerical experiments are performed in a velocity domain limited by $v_r, v_p \in [0, \sqrt{2}]$, uniformly discretized with a $2^r \times 2^r$ mesh. Two different time steps, $\Delta t = 0.01\tau$ and $\Delta t = \tau$, are considered to address the impact of large time steps in the effectiveness of the preconditioner. Here, τ is the collisional time scale of the problem, given by $\tau = 4\pi(Ze)^4 n\lambda/m^2 v_0^3$, where Ze is the charge of the species under consideration, m is the mass, n is the density of the plasma, λ is the Coulomb logarithm, and v_0 is the characteristic speed in the system. The Fokker–Planck collision operator is linearized [3] to prevent Newton's method and the time-adaptive scheme from obscuring the effectiveness of the preconditioner for the large time step case. The initial distribution function for the numerical experiments is chosen so that no unphysical results are obtained with the linearized Fokker–Planck solver for the large time step (Subsection 5.2). The symmetric beam in Fig. 3 satisfies this requirement.

Three different quantities are monitored as the mesh size r is varied: the number of Fokker–Planck iterations per time step, the total number of Poisson iterations per Fokker–Planck iteration (including the linear and non-linear systems), and the cumulative CPU time in an HP9000/735 workstation over the first three time steps, in seconds. The latter is the most important figure of merit to measure the effectiveness of MG. Results are depicted in Fig. 4 and suggest the following observations:

• The number of iterations in the Fokker–Planck system is strongly dependent on the magnitude of the time step. Thus, although MG and SGS perform similarly for small time

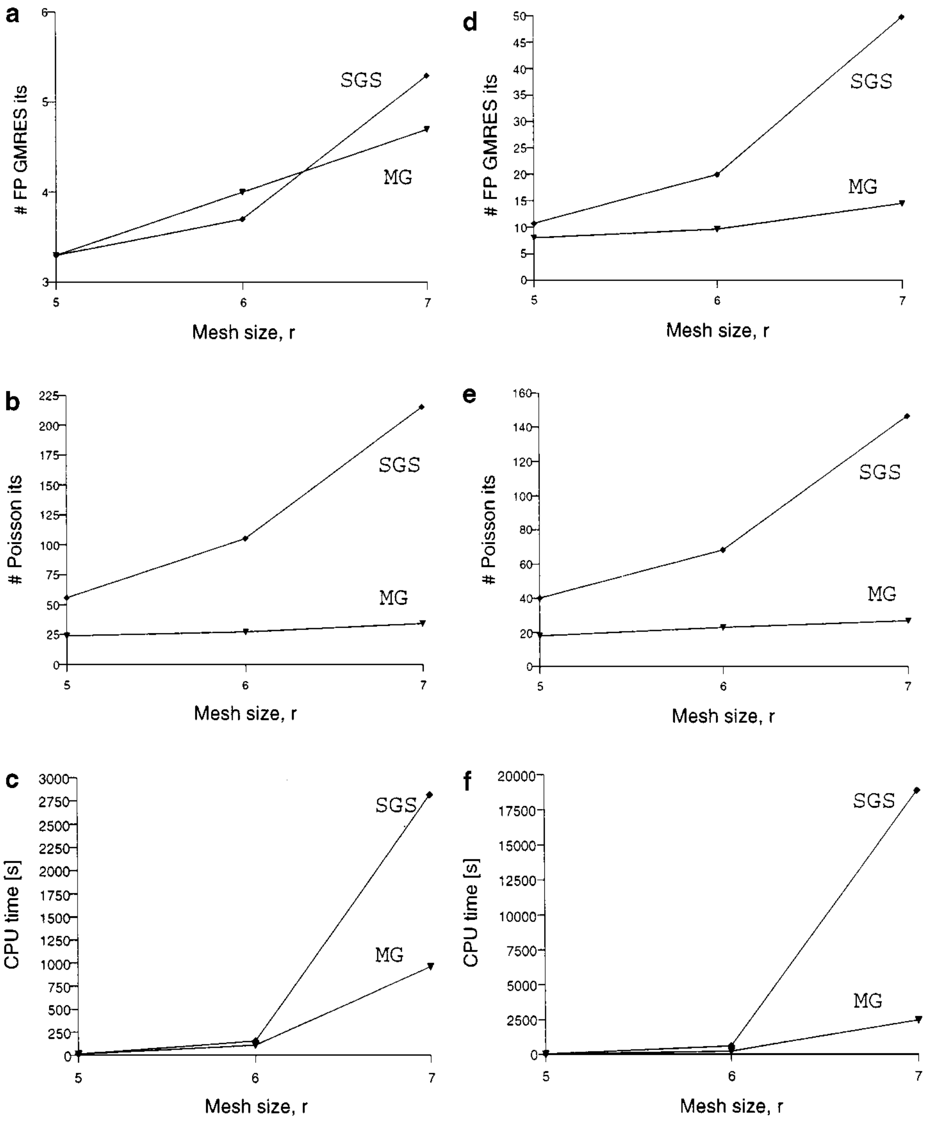


FIG. 4. Comparison of the performance of the SGS preconditioner vs the MG preconditioner in terms of the mesh refinement (given by $r = 5, 6, 7$ in a $2^r \times 2^r$ mesh) and magnitude of the time step [$\Delta t = 0.01\tau$ (left column) and $\Delta t = \tau$ (right column)]. The comparison is done for the number of Fokker–Planck iterations per time step (a),(d); the number of Poisson iterations per Fokker–Planck iteration (b),(e); and the cumulative CPU time in an HP9000/735 workstation in seconds (c),(f) in the first three time steps.

steps (Fig. 4a), MG clearly outperforms SGS for large time steps (Fig. 4d). Note that MG renders the number of Fokker–Planck iterations almost constant with the mesh refinement (as expected), and that its performance is only weakly dependent on the time step size.

- As for the number of Poisson iterations per Fokker–Planck iteration, the superiority of MG is evident for both the small and large time steps (Figs. 4b, 4e). Again, MG keeps the number of iterations almost constant with the mesh size and the time step.
- The results for the CPU time provide crucial insight about the effectiveness of MG preconditioning techniques. According to Figs. 4c and 4f, although MG is generally faster than SGS, it only outperforms SGS significantly in fine meshes, and the improvement is greater for large time steps (almost an order of magnitude for the $2^7 \times 2^7$ mesh and $\Delta t = \tau$). This occurs in spite of the significant improvement that MG introduces in the number of Fokker–Planck and Poisson iterations for all mesh refinements, particularly for $\Delta t = \tau$. The reason is that SGS is much cheaper (CPU-wise) per preconditioning call than MG, and this somewhat offsets the effect of the reduction of the number of iterations in the CPU time.

These results indicate that, although the MG preconditioner is effective for any mesh refinement and any time step, it is particularly suited for fine meshes and large time steps. The results also confirm the ability of Krylov techniques to deal with the variety of algebraic systems present (and particularly with the dense Fokker–Planck system), as indicated by the relatively small number of iterations and short CPU times in all cases.

5.2. *Effectiveness of the Time Adaptive Scheme*

A robust solver gives meaningful answers for extreme choices of mesh refinements and time step sizes. Although numerical instabilities are not an issue here due to the implicitness of the time integration and the nature of the problem at hand, convergence problems may arise in the Newton–Raphson non-linear algorithm for the time integration. Large time steps are the most critical, since Newton's radius of convergence varies inversely with the time step.

An adaptive time step scheme has been implemented (Subsection 3.2) to avoid divergence of Newton's method in these situations. Its effectiveness is analyzed here by monitoring the performance of the solver for the particular case of the radial beam in Fig. 3. The velocity domain is the same as in the previous section, uniformly discretized with a 32×32 mesh. It is of interest to find the solution of the full *non-linear* Fokker–Planck collision operator at $t = 1.5\tau$ with a single time step of size $\Delta t = 1.5\tau$, with and without the aid of the time adaptive scheme.

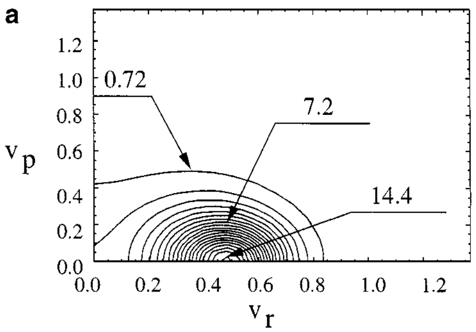
The solution of the problem *with* the time adaptive scheme is shown in Fig. 5a. The different time steps that the time adaptive scheme has selected along the Newton iteration, together with the magnitude of the Newton residual, are shown in Table II. The time adaptive scheme succeeds in finding an initial time step that places the initial distribution function within the Newton radius of convergence $(\Delta t = 0.13\tau)$. Subsequent change of the time step in each iteration does not preclude convergence, as indicated by the decreasing trend of the Newton residual. It does preclude, however, the quadratic convergence rate characteristic of Newton's method, which only appears when the time step remains fixed (i.e., after the target time step $\Delta t = 1.5\tau$ is reached).

In the case of solving *without* the time adaptive scheme, the Newton iteration does not converge, as shown in Table III. The reason for this divergence can be found by looking

TABLE II

Evolution of the Newton Residual of the Fokker–Planck Nonlinear System for a Single Time Step $\Delta t = 1.5\tau$ (where τ Is the Collision Time Scale), Using the Time-Adaptive Scheme to Avoid Unphysical Solutions

Newton it.	Adaptive time step	Magnitude of residual $\|\mathbf{e}_r\|_2$
1	0.13τ	634.16
2	0.26τ	119.2
3	0.39τ	73.48
4	0.52τ	49.56
5	0.65τ	36.34
6	0.78τ	27.94
7	0.91τ	22.15
8	1.03τ	18.1
9	1.16τ	15.0
10	1.5τ	29.52
11	1.5τ	4.61
12	1.5τ	$3.32 \cdot 10^{-2}$
13	1.5τ	$8.6 \cdot 10^{-5}$



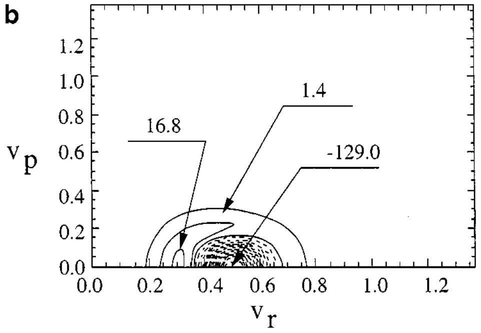


FIG. 5. Results of the EC solver for the radial test case at $t = 1.5\tau$, obtained by (a) solving the full non-linear Fokker–Planck equation with the adaptive time-step scheme and $\Delta t = 1.5\tau$, (b) solving only for the first Newton iteration with $\Delta t = 1.5\tau$. Some reference values of the distribution function at selected contours are indicated; contours are equally spaced.

TABLE III

Evolution of the Newton Residual of the Fokker–Planck Non-linear System for a Single Time Step $\Delta t = 1.5\tau$ (where τ Is the Collision Time Scale), Without Using the Time-Adaptive Scheme

Newton it.	Magnitude of residual $\|\mathbf{e}_r\|_2$
1	634.2
2	501544.0
3	No convergence in GMRES

at the distribution function after the first Newton iteration, depicted in Fig. 5b. This plot shows that the resulting distribution function is negative, which indicates that the original time step $\Delta t = 1.5\tau$ places this initial distribution function out of the radius of convergence of the Newton–Raphson algorithm. Note that, ultimately, the failure of the algorithm does not come from a divergent Newton algorithm, but from the negativity of the distribution function, which results in an ill-conditioned linear system and GMRES fails to converge.

These results indicate that the time adaptive technique is successful in ensuring convergence for exceptionally large time steps, thus improving the robustness of the solver.

5.3. *Energy Conservation in the Solver*

The issue of energy conservation is fundamental and has been the driver for the whole development presented herein. As shown in Subsection 3.1, the non-linear time integration is energy conservative provided that the discretization of the Fokker–Planck operator in velocity space assures the cancellation of the energy moment. Although it is possible to

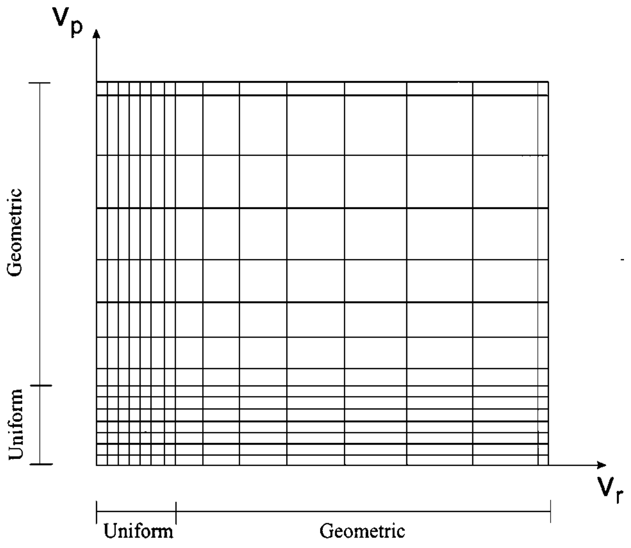


FIG. 6. Sketch of a combined uniform-geometric discretization mesh, with additional accuracy provided at the outer boundaries.

develop a difference scheme that dramatically improves this cancellation [4], exact numerical cancellation of the energy moment is not possible due to errors at the boundaries of a finite velocity domain. Thus, it is crucial to ascertain how this error propagates along the time integration, whether the energy conservative solver provides better results than other implicit iterative solvers, and what is the cost—in terms of CPU time—of the energy conservative approach vs. other implementations.

In order to gauge the performance of the energy-conservative (EC) scheme, a particle conservative, non-energy-conservative (NEC) implicit iterative solver has been implemented. The NEC solver employs the iterative approach outlined in Eq. (14), with the collision operator discretized in velocity space with a centered finite difference scheme [4]. To ensure a fair comparison, MG-preconditioned Krylov techniques are also used in the NEC (MG-preconditioned GMRES applied to Eq. (14) is by itself a new and useful contribution).

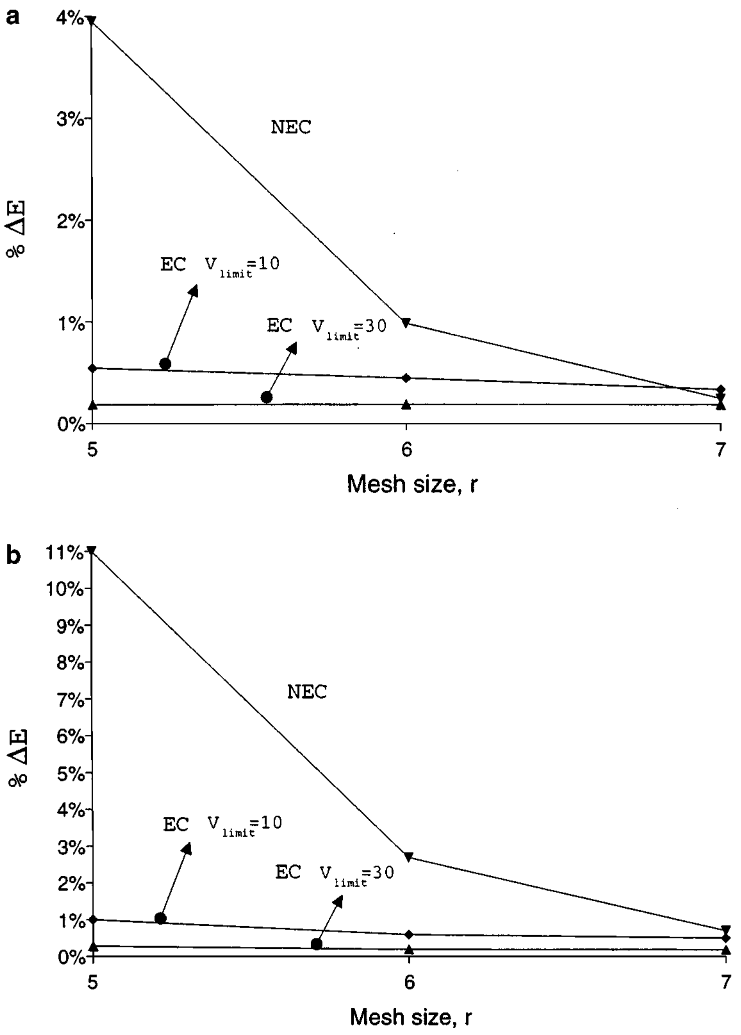


FIG. 7. Global energy error of the energy-conservative (EC) solver and the non-conservative solver (NEC) in terms of the mesh refinement (given by $r = 5, 6, 7$ in a $2^r \times 2^r$ mesh) in a period of 5τ, for both (a) the symmetric and (b) the radial initial distribution functions.

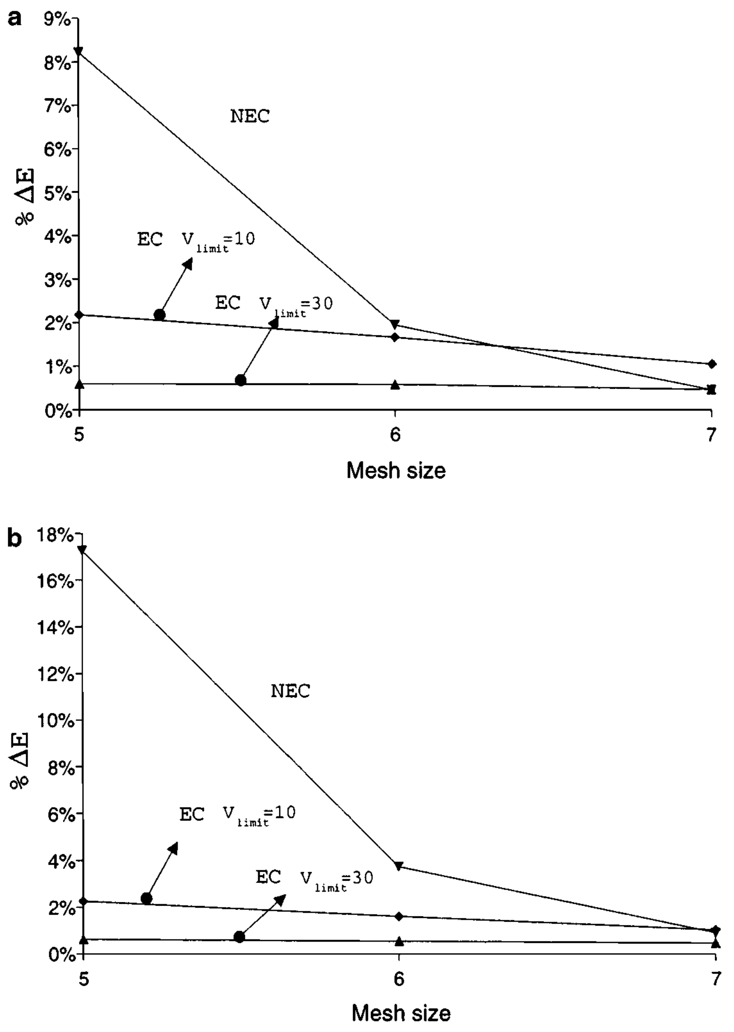


FIG. 8. Global energy error of the energy-conservative (EC) solver and the non-conservative solver (NEC) in terms of the mesh refinement (given by $r = 5, 6, 7$ in a $2^r \times 2^r$ mesh) in a period of 10τ, for both (a) the symmetric and (6) the radial initial distribution functions.

For adequate energy conservation, the 2D cylindrical velocity domain is discretized [4] with a combined uniform-geometric mesh (Fig. 6) with 2^r points per direction, split in a $\frac{1}{2}/\frac{1}{2}$ proportion between the uniform and geometric regions. The uniform mesh region is limited by $v_r, v_p \in [0, \sqrt{2}]$; the geometric mesh region is limited by $v_r, v_p \in [\sqrt{2}, v_{limit}]$.

Simulations are performed for both the symmetric and the radial initial distribution functions (Fig. 3). Two magnitudes are monitored, namely, the cumulative energy error in time periods of 5τ (the time required for the system to reach LTE) and 10τ, and the CPU time spent in the simulation. The EC approach solves the linearized Fokker–Planck operator (first Newton iteration); the NEC approach performs ten iterations on the Rosenbluth potentials per time step. The time step is $\Delta t = 0.2\tau$ in the radial beam case (to prevent unphysical results), and $\Delta t = \tau$ in the symmetric beam case. The velocity domain limit is taken as $v_{limit} = 10$ unless otherwise specified. Convergence tolerances in the iterative solvers are set to $\|\mathbf{e}_r\|_2 < 10^{-7}$, where $\mathbf{e}_r$ is the vector of residuals. The MG preconditioner in these

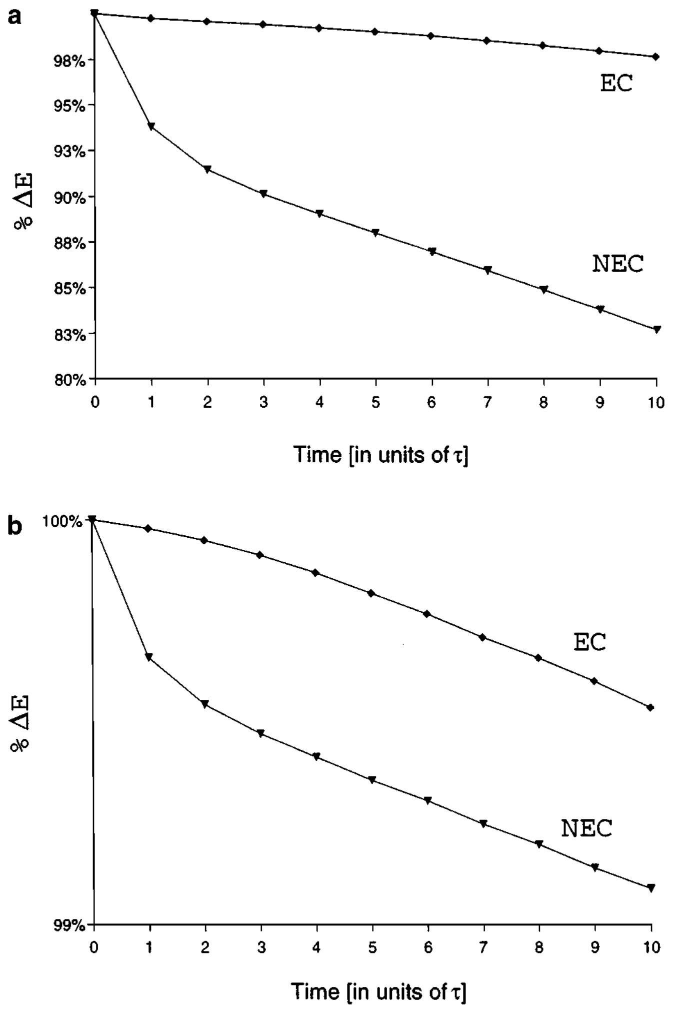


FIG. 9. Time histories of the cumulative energy error $\Delta E(\%)$ in the EC and NEC solvers, for the radial initial distribution function with $v_{limit} = 10$ in (a) a 32×32 mesh, and (b) a 128×128 mesh.

particular simulations employs three consecutive V-cycles with two SGS passes per smoother call.

The results for the cumulative energy error (defined as $\Delta E(\%) = 100 \times |(E_f - E_0)/E_0|$, where E_0, E_f are the initial and final energies, respectively) of the EC and NEC solvers with both the radial and the symmetric initial distribution functions are depicted in Fig. 7 (for $t = 5\tau$) and Fig. 8 (for $t = 10\tau$). Both figures present the same patterns, although magnitudes of relative errors are different. The scaling of the energy error is of particular interest. The energy error in the NEC solver shows a $1/N$ scaling (where $N = 2^r \times 2^r$ is the number of mesh points), consistent with a second order accurate difference scheme. On the contrary, the energy error from the EC solver presents virtually no scaling with the mesh refinement (in contradiction with the $\frac{1}{N}$ scaling found in Ref. [4] for the error in the cancellation of the energy moment). The discrepancy originates in the fact that the boundary terms of the integral in Eq. (29) are different from the boundary terms of the energy moment integral in Eq. (9) and do not follow the same scaling laws. Consequently, energy conservation in the

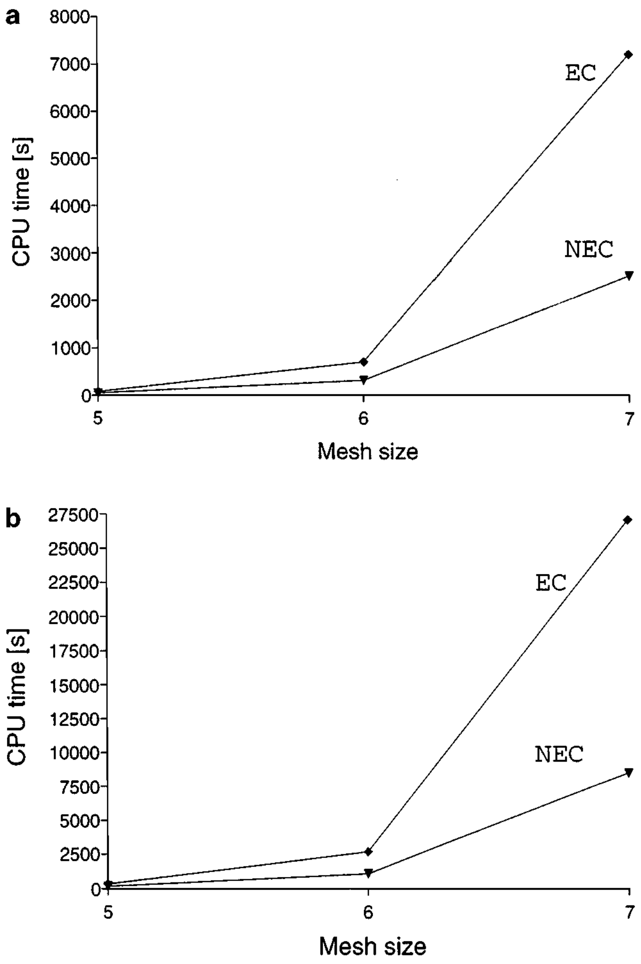


FIG. 10. The CPU times of the EC and NEC solvers corresponding to a $t = 10\tau$ run, for both (a) the symmetric and (b) the radial initial distribution functions, and for different mesh refinements $N = 2^r \times 2^r$ ($r = 5, 6, 7$).

EC solver is better than in the NEC solver in coarse meshes (by an order of magnitude for the $2^5 \times 2^5$ mesh), but the advantage is lost in fine meshes.

The energy error in the EC solver does show a $1/v_{limit}$ scaling (as indicated by the results for $v_{limit} = 10, 30$). However, v_{limit} is effectively bound by efficiency considerations (MG preconditioning works best with uniform or nearly uniform meshes) as well as accuracy considerations (the $1/v_{limit}$ scaling is lost for v_{limit} sufficiently large [4]).

Time histories of the cumulative energy error $\Delta E(\%)$ of the radial initial distribution function with both the EC and NEC solvers are presented in Fig. 9. The cumulative energy error is monitored up to $t = 10\tau$. Results are plotted for a 32×32 mesh (Fig. 9a—where the energy discrepancy between the EC and NEC solvers is large—and for a 128×128 mesh (Fig. 9b—where the energy discrepancy is small. In all cases, $v_{limit} = 10$. These figures show that, while the energy change with the EC solver evolves linearly with time at all times, the energy change with the NEC solver behaves non-linearly during the first two collision times (precisely when the distribution function changes more drastically towards the Maxwellian distribution) and evolves linearly after that.

The CPU times resulting from the $t = 10\tau$ simulations with both EC and NEC are presented in Fig. 10. Clearly, although both solvers are within the same order of magnitude, the NEC solver outperforms the EC solver for the particular set of parameters chosen. This is to be expected because:

1. The EC solver is effectively inverting a dense system. The fact that the EC CPU time is within the same order of magnitude as the NEC CPU time indicates that the Jacobian-free GMRES algorithm is successful in dealing with the dense algebraic system.
2. The NEC method also employs the same powerful iterative matrix-inversion techniques, namely, MG-preconditioned GMRES. In fact, these results prove the effectiveness of these methods in ”traditional” approaches of dealing with the implicit integration of the Fokker–Planck transport equation.

It is of interest to note that the CPU time in both the EC solver and NEC solver scales as $O(N^{3/2})$. This observation is consistent with profiling results of the code that show that the CPU time is dominated, for large meshes, by the calculation of the far-field boundary conditions of the Rosenbluth potentials (procedure that scales as $O(N^{3/2})$, as discussed in Ref. [4]).

6. CONCLUSIONS

In this paper, the development of an energy-conservative solver for the multidimensional Fokker–Planck equation has been undertaken. The solver uses the energy-conservative difference scheme developed in Ref. [4] and is based on the coupling of the implicit time integration with Newton’s method. Such formulation requires the inversion of dense algebraic systems, efficiently performed by Jacobian-free Newton–Krylov iterative methods. These techniques are more efficient than standard techniques and avoid forming and storing the dense matrix. The efficiency of Krylov methods can be boosted further if adequate preconditioning schemes are employed. Here, multigrid preconditioning (MG) is used. Results indicate that MG is effective for any mesh refinement and any time step and is particularly suited for fine meshes and large time steps.

A direct use of Newton’s method in the implicit time integration does not guarantee convergence unless the initial condition is within Newton’s radius of convergence. This, in turn, strongly depends on the time step (large time steps correspond to smaller radii). To ensure convergence, a time adaptive scheme has been implemented. Simulations indicate that the scheme is successful for exceptionally large time steps.

Conservation of energy is not exact due to errors at the boundaries of finite velocity domains. The propagation of the energy error in the energy-conservative solver (EC) has been monitored and compared against that of a non-energy-conservative solver (NEC) for different initial conditions. The comparison shows that EC outperforms NEC by as much as an order of magnitude in coarser meshes, but that the advantage is lost in fine meshes.

A comparison of the CPU times for the EC and NEC solvers shows that the NEC solver is more efficient in fine meshes. Hence, as it stands, the EC solver is the best alternative to deal with problems in coarse meshes, while the NEC solver is more effective for fine meshes. However, the fact that the CPU times of both the EC and NEC solvers are within the same order of magnitude indicates that MG-preconditioned Jacobian-free Newton–Krylov techniques are dealing successfully with the dense algebraic system. Hence, future development of energy-conservative Fokker–Planck solvers in multidimensional geometries may only be concerned with the design of better energy-conservative difference schemes.

The development of faster algorithms to calculate the far-field boundary conditions of the Rosenbluth potentials (to provide $O(N)$ scaling), as well as the improvement of the cancellation of the boundary terms in the difference scheme [4] to provide better energy conservation values, are still pending issues.

ACKNOWLEDGMENTS

The Krylov linear solver package PCG (W. Joubert, G. F. Carey, Copyright 1992–1995, University of California and University of Texas at Austin) has been used in this work. This research has been developed under OFES funding at Los Alamos National Laboratory. L.C. has been partially funded by a Fellowship of the Spanish Ministry of Education and Culture, and this research is part of his Ph.D. dissertation at the University of Illinois.

REFERENCES

1. F. L. Hinton, Collisional transport in plasma, in *Handbook of Plasma Physics*, edited by M. N. Rosenbluth and R. Z. Sagdeev (North-Holland, Amsterdam, 1984), Vol. 1, p. 147.
2. J. S. Chang and G. Cooper, A practical difference scheme for Fokker–Planck equations, *J Comput. Phys.* **6**, 1 (1970).
3. E. M. Epperlein, Implicit and conservative difference scheme for the Fokker–Planck equation, *J. Comput. Phys.* **112**, 291 (1994).
4. L. Chacón, D. C. Barnes, D. A. Knoll, and G. H. Miley, An implicit energy-conservative 2D Fokker–Planck algorithm. I. Difference scheme. *J. Comput. Phys.* **157**, 618 (2000).
5. Y. Saad, *Iterative Methods for Sparse Linear Systems* (PWS, Boston, 1996).
6. P. N. Brown and Y. Saad, Hybrid krylov methods for nonlinear systems of equations, *SIAM J. Sci. Stat. Comput.* **11**, 450 (1990).
7. P. McHugh and D. Knoll, Inexact Newton's method solution to the incompressible Navier–Stokes and energy equations using standard and matrix-free implementations, *AIAA J.* **32**(12), 2394 (1994).
8. Y. Saad, Preconditioning techniques for indefinite and non-symmetric linear systems, *J. Comput. Appl. Math.* **24**, 89 (1988).
9. V. Mousseau and D. Knoll, Fully implicit kinetic solution of collisional plasmas, *J. Comput. Phys.* **136**, 308 (1997).
10. D. Knoll and W. Rider, A multigrid preconditioned Newton–Krylov method, *SIAM J. Sci. Comput.* **21**, 691 (1999).
11. D. Knoll, G. Lapenta, and J. Brackbill, A multilevel iterative field solver for implicit, kinetic, plasma simulation, *J. Comput. Phys.* **149**, 1 (1999).
12. J. K. Reid, On the method of Conjugate Gradients for the solution of large sparse systems of linear equations, in *Large Sparse Sets of Linear Equations* edited by J. K. Reid (Academic Press, New York, 1971), p. 231.
13. Y. Saad and M. Schultz, GMRES: A generalized minimal residual algorithm for solving non-symetric linear system, *SIAM J. Sci. Stat. Comput.* **7**, 856 (1986).
14. M. N. Rosenbluth, W. M. Macdonald, and D. L. Judd, Fokker–Planck equation for an inverse-square force, *Phys. Rev.* **107**, 1 (1957).
15. J. D. Jackson, *Classical Electrodynamics*, 2nd ed. (Wiley, New York, 1975), p. 239.
16. Y. Saad, Krylov subspace methods for solving large unsymmetric linear systems, *Math. Comp.* **37**, 105 (1981).
17. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins Univ. Press, Baltimore, 1996).
18. W. L. Briggs, *A Multigrid Tutorial* (SIAM, Philadelphia, 1987).
19. J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations* (Chapman & Hall, London/ New York, 1989).