JOURNAL OF COMPUTATIONAL PHYSICS **136,** 308–323 (1997)
ARTICLE NO. CP975736

Fully Implicit Kinetic Solution of Collisional Plasmas

V. A. Mousseau

Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, M.S. TWFN-10E46, Washington, D.C. 20555-0001

and

D. A. Knoll

Theoretical Division, Los Alamos National Laboratory, M.S. K717, Los Alamos, New Mexico 87545

Received March 13, 1995; revised February 13, 1997

In this paper we describe a numerical technique, matrix-free Newton–Krylov, for solving a single species, stationary, one spatial dimension, one velocity dimension, Vlasov–Fokker–Planck equation. This method is both deterministic and fully implicit and may not have been a viable option before recent developments in numerical methods. It is demonstrated that efficient steady-state solutions to the nonlinear integro-differential equation, can be achieved, obtaining quadratic convergence, but not incurring the large memory requirements of an integral operator. We present a model problem which simulates ion transport in the edge plasma of a tokamak fusion reactor and use this model problem to demonstrate the performance of the new solution method. We demonstrate that the solution algorithm is compatible with a higher-order, monotone, convective differencing scheme.

1. INTRODUCTION

The need to use kinetic modelling in collisional plasmas for resolution of the distribution functions arises from the desire for an accurate description of reaction rates, transport quantities, and electric field. To accomplish this we need to resolve the structure of the distribution functions of ions and electrons in velocity space. The plasma fluid equations [1] are based on the assumption that the distribution functions are Maxwellian over the length scales of interest, and when this assumption applies, more efficient computations are possible. In this paper we are concerned with developing an advanced solution algorithm for applications where the distribution functions can be non-Maxwellian. A wide variety of plasma applications require some level of kinetic simulation to address the effect of non-Maxwellian distribution functions. Examples of these are edge plasma modeling for magnetically confined fusion [2–4], inertial confinement fusion plasmas [5], plasma processing applications [6], and lasers [7]. In each of these applications there exists the need to resolve length scales on the order of the particle mean free path. Since the particles must travel a few mean free paths before collisions redistribute them to a local Maxwellian distribution, these applications may have significant regions that are non-Maxwellian and thus require kinetic modelling.

In this paper we present a fully implicit solution algorithm for the Vlasov–Fokker–Planck equation and apply it to kinetic modeling of the edge plasma in a tokamak fusion reactor. The algorithm contains no operator splitting or directional splitting, and all integral coupling is evaluated at the current iteration level. Everywhere that the distribution function occurs, including in the moment quantities of temperature, pressure, fluid velocity, or number densities, the distribution function is evaluated at new time values. This couples all distribution functions at a given spatial location (i.e., across all of velocity space). This fully implicit coupling allows large Courant number simulations and thus efficient steady-state solutions. We do this by employing a preconditioned matrix-free Newton–Krylov method [8]. The preconditioner is an incomplete LU factorization (ILU) [9] of a modified Jacobian. For the preconditioner, the Jacobian is modified by evaluating the integral coupling at the previous iteration level, in other words, we are using only the differential operators at new time in the preconditioner.

Our model problem arises from transport in the tokamak edge plasma. In a diverted tokamak, the core plasma and edge plasma are separated by the magnetic separatrix. As the plasma diffuses radially across the tokamak separatrix, it moves from a region of closed magnetic field lines, the core, to a region of open magnetic field lines, the edge. These open magnetic field lines guide the plasma to the divertor plates. For some edge plasma conditions the particle mean free path due to coulomb collisions, which scale as $n/T^{3/2}$, may be on the same order as the problem geometry. This similarity in scale between mean free path and fluid transport length motivates a kinetic treatment [2, 10–14]. Our model problem is not used to study kinetic effects in the edge plasma. It is used only to demonstrate the performance of the solution algorithm.

0021-9991/97 $25.00

The organization of the rest of the paper is as follows. Section 2 presents the mathematical model and Section 3 defines the discrete equations. Section 4 motivates the use of the matrix-free Newton–Krylov method and describes the algorithm with emphasis on Newton's method, the Krylov method, and the use of the matrix-free approximation. Section 5 describes our model problem and provides calculational results. Section 6 summarizes the work completed and decsribes future work to be done.

2. MATHEMATICAL MODEL

Our model equation contains both the Vlasov operator and a simplified Fokker–Planck collision operator. In general form, the time dependent Vlasov–Fokker–Planck equation has seven independent variables, three velocity dimensions (3V), three spatial dimensions (3D), and one temporal dimension. To demonstrate the numerical method we apply it to a simplified two-dimensional, 1D–1V, model equation describing ion transport in a tokamak divertor. This equation is derived from a more complete three-dimensional, 1D–2V, Vlasov–Fokker–Planck equation for transport along a magnetic field line [2, 13], in which the two velocity coordinates are perpendicular and parallel to the magnetic field lines.

For our model problem, we eliminate the perpendicular velocity dimension, $v_\perp$, by assuming it to be Maxwellian [15]. More formally,

$$f(v, v_\perp) = \left(\frac{m}{2\pi kT}\right) \exp \frac{-mv_\perp^2}{2kT} F(v). \tag{1}$$

Here f is the three-dimensional distribution function, F is the two-dimensional distribution function, v is the particle velocity along the magnetic field line, $v_\perp$ is the particle velocity perpendicular to the magnetic field line, m is the particle mass, k is the Boltzmann constant, and T is the fluid temperature. With this assumption, the distribution function has the form $F(x, v)$ where x is the distance along the magnetic field line. Therefore, the two dimensional steady-state ion transport equation is

$$v\frac{\partial F}{\partial x} + \frac{qE_x}{m}\frac{\partial F}{\partial v} = \left.\frac{\partial F}{\partial t}\right|_{\text{col}} + \left.\frac{\partial F}{\partial t}\right|_{\text{neut}} + \left.\frac{\partial F}{\partial t}\right|_{\text{src}}, \tag{2}$$

where q is the particle charge. The first term accounts for the convective flux. The second term represents the acceleration of the charged particles due to the electric field. The first term on the right hand side (RHS) represents the ion–ion collisions. The second term on the RHS accounts for the neutrals that are created at the plate, transport back into the plasma, and are ionized by electrons. The final term on the RHS is the volumetric source term that represents the diffusion of the ''hot'' ions from the core into the edge plasma. Only the ion kinetic equation will be solved. The electron density is obtained from quasi-neutrality and the electron temperature is obtained by assuming thermal equilibrium with the ions. The electron density and temperature will only contribute to the ion computation through the electric field.

Since our motivation is to demonstrate and study the numerical methods, we have chosen a simplified Fokker–Planck collision operator [15–18] which has the form

$$\left.\frac{\partial F}{\partial t}\right|_{\text{col}} = \nu_{\text{col}}\left\{\left[\frac{kT}{m}\right]\frac{\partial^2 F}{\partial v^2} + \frac{\partial[(v-\overline{V})F]}{\partial v}\right\}, \tag{3}$$

where ν_{col} is the collision frequency. Note that this collision operator has the following three properties. First, it conserves the number of particles. Second, it represents both the friction between particles and the diffusion of particles in velocity space. Finally, it has no effect on a Maxwellian distribution.

In a high recycling divertor, a significant fraction of the plasma which strikes the divertor plate is recycled as neutral atoms which diffuse back into the plasma and are ionized through electron impact ionization. In our model problem, the neutrals which are generated at the plate are assumed to have an exponential profile falling off from the plate as

$$n_{\text{n}}(x) = n_{\text{n}}(L) \exp\left(\frac{(x-L)}{\lambda_n}\right). \tag{4}$$

Here n_{n} is the neutral number density, λ_{n} is the neutral mean free path defined by ionization, and L is the length of the region. The exact form of the ionization source term is

$$\left.\frac{\partial F}{\partial t}\right|_{\text{neut}} = \frac{\nu_{\text{ie}}\delta(v - v_{\text{n}})n_{\text{n}}}{\Delta v} \tag{5}$$

where ν_{ie} is the electron impact ionization frequency and Δv is the mesh spacing in the v direction. This assumes that the neutral distribution function is a beam with velocity v_n. The correct distribution for the neutrals is unknown so a beam has been chosen for simplicity. Other neutral particle distributions can be included in the future. The core source term is based on work from Emmert [19] and has the form

$$\left.\frac{\partial F}{\partial t}\right|_{\text{src}} = \left(\frac{1}{\Delta v}\right)\left(\frac{R_{\text{inj}}h(x)vm}{2kT_{\text{inj}}}\right)\exp\left[\frac{m(v-\overline{V})^2}{2kT_{\text{inj}}}\right], \tag{6}$$

where R_{inj} is the particle injection rate and $h(x)$ is the injection shape factor such that

$$\int_0^{L_{\text{inj}}} h(x)\,dx = 1. \tag{7}$$

This source was designed to produce Maxwellian distributions in the absence of an electric field. It may not be the most appropriate source model for collisional plasmas and we will investigate other possibilities in future work. The particle source comes mainly from neutral recycling at the plate and the core source is used primarily for energy, i.e., to define a temperature. The distribution function moments are defined by

$$n = \int_{-\infty}^{\infty} F\,dv \tag{8}$$

$$n\overline{V} = \int_{-\infty}^{\infty} vF\,dv \tag{9}$$

$$nkT = m\int_{-\infty}^{\infty} (v - \overline{V})^2\,F\,dv, \tag{10}$$

where n is the particle density, $\overline{V}$ is the fluid velocity, and T is the fluid temperature. For this single species work, we have used a simplified form of the electric field calculation. If we assume the electrons to be governed by the same form of kinetic equation (ignoring unlike coulomb collisions), then the electron momentum equation is

$$m_{\text{e}}\frac{\partial n_{\text{e}}\overline{V}_{\text{e}}^2}{\partial x} = -\frac{\partial P_{\text{e}}}{\partial x} - en_{\text{e}}E_x. \tag{11}$$

Ignoring electron inertia and assuming that $P_{\text{i}} = P_{\text{e}} = P$ and $\text{n}_{\text{i}} = \text{n}_{\text{e}} = n$, we then have the form of the electric field

$$E_x = -\frac{1}{en}\frac{\partial P}{\partial x}, \tag{12}$$

where e is the charge of an electron. Again we have chosen a simplified model to demonstrate the numerics. As we progress to a coupled ion–electron problem we will use a more correct electric field which combines the mv moment of each kinetic equation and quasi-neutrality [20]. Finally, we close out the equation set with an equation of state

$$P = nkT. \tag{13}$$

The two-dimension x–v space computational domain is shown in Fig. 1. For the model problem, the computational domain extends from 1.5×10^5 m/s to -1.5×10^5 m/s in the velocity direction (v) and from zero to 6 m in the spatial direction (x). The upper left boundary is a symmetry plane in x, which is described by

$$F(0, v) = F(0, -v) : v \in [0, v_{\max}]. \tag{14}$$

The lower left and upper right boundaries are simple outflow boundaries being approximated with a Neumann condition or

$$\frac{\partial F(0, v)}{\partial x} = \frac{\partial F(L, v)}{\partial x} = 0 : v \in [0, v_{\max}]. \tag{15}$$

The lower right boundary is the no flow wall boundary,

$$F(L, v) = 0 : v \in [0, -v_{\max}]. \tag{16}$$

The velocity boundaries are set to conserve mass [21] and they are

$$\begin{aligned}\frac{\partial F(x, v_{\max})}{\partial v} &= \frac{\partial F(x, -v_{\max})}{\partial v} \\ &= F(x, v_{\max}) = F(x, -v_{\max}) = 0 : x \in [0, L].\end{aligned} \tag{17}$$

3. DISCRETE EQUATIONS

The discrete version of the Vlasov–Fokker–Plank equation is

$$\begin{aligned}&v_j\frac{\tilde{F}_{i+1/2,j}^{n+1} - \tilde{F}_{i-1/2,j}^{n+1}}{\Delta x_i} + \frac{q}{m}E_i^{n+1}\frac{\tilde{F}_{i,j+1/2}^{n+1} - \tilde{F}_{i,j-1/2}^{n+1}}{\Delta v_j} \\ &\quad= \frac{\nu_{\text{col}}}{\Delta v_j}\left\{\left[\frac{k}{m}T_i^{n+1}\frac{F_{i,j+1}^{n+1} - F_{i,j}^{n+1}}{\Delta v_{j+1/2}}\right.\right. \\ &\qquad\left.\left. + (v_{j+1/2} - \overline{V}_i^{n+1})\,\tilde{F}_{i,j+1/2}^{n+1}\right] \right. \\ &\qquad\left. - \left[\frac{k}{m}T_i^{n+1}\frac{F_{i,j}^{n+1} - F_{i,j-1}^{n+1}}{\Delta v_{j-1/2}}\right.\right. \\ &\qquad\left.\left. + (v_{j-1/2} - \overline{V}_i^{n+1})\,\tilde{F}_{i,j-1/2}^{n+1}\right]\right\},\end{aligned} \tag{18}$$

where the tilde ($\tilde{F}$) indicates a face value that will be described later. The electric field, equation of state, and

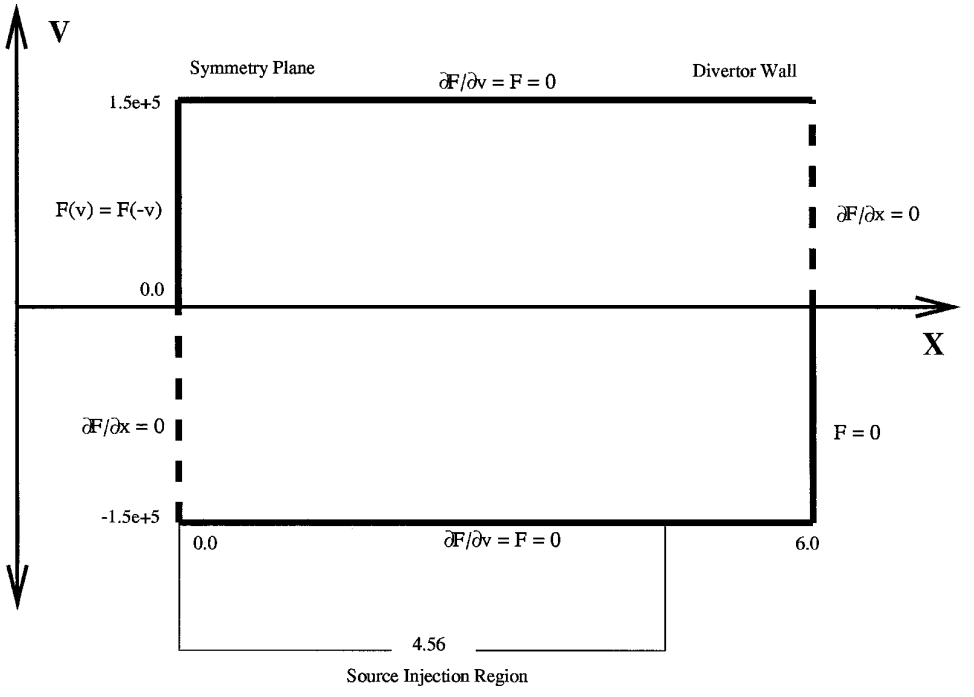

FIG. 1. Model problem geometry.

fluid moment integral quantities are evaluated using the midpoint rule,

$$E_i^{n+1} = -\frac{1}{en_i^{n+1}} \frac{P_{i+1}^{n+1} - P_{i-1}^{n+1}}{2\Delta x_i} \tag{19}$$

$$P_i^{n+1} = n_i^{n+1} k T_i^{n+1} \tag{20}$$

$$T_i^{n+1} = \frac{m}{n_i^{n+1} k} \sum_{j=1}^{ny} (v_j - \overline{V}_i^{n+1})^2 F_{i,j}^{n+1} \Delta v_j \tag{21}$$

$$\overline{V}_i^{n+1} = \frac{1}{n_i^{n+1}} \sum_{j=1}^{ny} v_j F_{i,j}^{n+1} \Delta v_j \tag{22}$$

$$n_i^{n+1} = \sum_{j=1}^{ny} F_{i,j}^{n+1} \Delta v_j. \tag{23}$$

Here the superscript, n, is the nonlinear iteration level, the subscript i is the spatial grid index, and the subscript j is the velocity grid index. The velocity space boundary conditions are

$$\frac{F_{i,ny}^{n+1} - F_{i,ny-1}^{n+1}}{\Delta v_{ny-1/2}} = \tilde{F}_{i,ny-1/2}^{n+1} = 0 : i = 1, nx$$

and

$$\frac{F_{i,1}^{n+1} - F_{i,0}^{n+1}}{\Delta v_{1/2}} = \tilde{F}_{i,1/2}^{n+1} = 0 : i = 1, nx.$$

The lower left boundary is

$$F_{1,j}^{n+1} = F_{2,j}^{n+1} : j = 1, \frac{ny}{2}.$$

The upper right boundary is

$$F_{nx,j}^{n+1} = F_{nx-1,j}^{n+1} : j = \frac{ny}{2} + 1, ny.$$

The upper left symmetry boundary is

$$F_{1,ny+1-j}^{n+1} = F_{1,j}^{n+1} : j = \frac{ny}{2} + 1, ny.$$

The no flow lower left wall boundary is

$$F_{nx,j}^{n+1} = 0 : j = 1, \frac{ny}{2},$$

where nx is the number of discrete points in the x direction and ny is the number of grid points in the v direction.

We will now define the difference form of the cell face quantities. The cell face distribution function values $(\tilde{F}_{i+1/2,j}, \tilde{F}_{i-1/2,j}, \tilde{F}_{i,j+1/2}, \tilde{F}_{i,j-1/2})$ are not direct unknowns in the discrete system so they need to be approximated. For

the base code, the first order operators in the Vlasov equation are upwind differenced. That is,

$$v_j \frac{\tilde{F}^{n+1}_{i+1/2,j} - \tilde{F}^{n+1}_{i-1/2,j}}{\Delta x_i} = \begin{cases} v_j \dfrac{F^{n+1}_{i+1,j} - F^{n+1}_{i,j}}{\Delta x_i} & \text{if } v_j > 0 \\ v_j \dfrac{F^{n+1}_{i,j} - F^{n+1}_{i-1,j}}{\Delta x_i} & \text{if } v_j \le 0, \end{cases} \tag{24}$$

and

$$\frac{q}{m} E^{n+1}_i \frac{\tilde{F}^{n+1}_{i,j+1/2} - \tilde{F}^{n+1}_{i,j-1/2}}{\Delta x_i} = \begin{cases} \dfrac{q}{m} E^{n+1}_i \dfrac{F^{n+1}_{i,j+1} - F^{n+1}_{i,j}}{\Delta v_j} & \text{if } \dfrac{q}{m} E^{n+1}_i > 0 \\ \dfrac{q}{m} E^{n+1}_i \dfrac{F^{n+1}_{i,j} - F^{n+1}_{i,j-1}}{\Delta v_j} & \text{if } \dfrac{q}{m} E^{n+1}_i \le 0. \end{cases} \tag{25}$$

The diffusion operators in the Fokker–Planck term are central differenced as shown in Eq. (18). The cell face distribution function values ($\tilde{F}^{n+1}_{i,j+1/2}$, $\tilde{F}^{n+1}_{i,j-1/2}$) in the collision operator are based on interpolated donor cell differencing [22]. Interpolated donor cell differencing is a linear combination of upwind differencing and central differencing. The upwind component is based on the direction of the frictional force. If the particle velocity is greater than the fluid velocity the frictional force is downward or decelerating. If the particle velocity is less than the fluid velocity the frictional force is upward or accelerating. In equation form this is

$$(v_{j-1/2} - \overline{V}^{n+1}_i) F_{\text{upwind}} = \begin{cases} (v_{j-1/2} - \overline{V}^{n+1}_i) F^{n+1}_{i,j+1} & \text{if } (v_{j-1/2} - \overline{V}^{n+1}_i) > 0 \\ (v_{j-1/2} - \overline{V}^{n+1}_i) F^{n+1}_{i,j} & \text{if } (v_{j-1/2} - \overline{V}^{n+1}_i) \le 0. \end{cases} \tag{26}$$

The central difference contribution for a uniform mesh is

$$(v_{j-1/2} - \overline{V}^{n+1}_i) F_{\text{central}} = (v_{j-1/2} - \overline{V}^{n+1}_i) \left[\frac{F^{n+1}_{i,j+1} + F^{n+1}_{i,j}}{2} \right], \tag{27}$$

so the whole term is

$$(v_{j-1/2} - \overline{V}^{n+1}_i) \tilde{F}^{n+1}_{i,j+1/2} = (v_{j-1/2} - \overline{V}^{n+1}_i) [\alpha F_{\text{upwind}} + (1 - \alpha) F_{\text{central}}]. \tag{28}$$

Here $\alpha = 0.05$ is a weighting term that is chosen to be small, but large enough to keep the differencing nonoscillatory.

For the advanced differencing results presented later, the first order operators in the Vlasov equation are differenced using flux-limited [23] QUICK (Quadratic Upstream Interpolation for Convective Kinematics) [24]. The QUICK method computes the cell face value from the two cell values straddling the cell face location and a third cell value in the ''upwind'' direction. A quadratic equation is fitted to these three cell centered values and the cell face value is computed from the evaluation of the quadratic. For an accelerating convective quantity in velocity space on a uniform mesh this results in

$$\tilde{F}^{n+1}_{i,j+1/2} = \tfrac{1}{8} (3F^{n+1}_{i,j+1} + 6F^{n+1}_{i,j} - F^{n+1}_{i,j-1}). \tag{29}$$

A flux-limiting strategy [23] is employed to ensure that the cell face value is monotone (i.e., the cell face value lies between the two cell center values, $F^{n+1}_{i,j+1} \ge \tilde{F}^{n+1}_{i,j+1/2} \ge F^{n+1}_{i,j}$ or $F^{n+1}_{i,j+1} \le \tilde{F}^{n+1}_{i,j+1/2} \le F^{n+1}_{i,j}$). The details of the flux-limiting and the nonuniform version of QUICK are described in Ref. [18] and will be presented in a future paper.

The collision operator for the advanced differencing examples is computed using the Chang–Cooper method [25]. The Chang–Cooper method is designed to cancel out the numerical error for a Maxwellian distribution. The Chang–Cooper differencing is

$$\tilde{F}^{n+1}_{i,j+1/2} = (1 - \delta) F^{n+1}_{i,j+1} + \delta F^{n+1}_{i,j}, \tag{30}$$

where

$$\delta = \frac{1}{w} - \frac{1}{\exp(w) - 1} \tag{31}$$

and

$$w = \frac{m(v_{j+1/2} - \overline{V}^{n+1}_i) \Delta v_j}{k T^{n+1}_i}. \tag{32}$$

The Chang–Cooper differencing, although it is still a first order method, is an inexpensive technique which helps to minimize discretization error by optimally choosing δ.

4. SOLUTION ALGORITHM

The solution algorithm presented in this paper is fully implicit. Here, all variables are evaluated at new time and solved for simultaneously as is shown in Section 3. Because the equations are nonlinear, Newton's method is used to linearize the problem. The resulting linear systems can be very large and dense (for a 100 velocity by 50 spatial grid there are 7.549×10^5 nonzero elements out of 2.5×10^7 total entries or a 3% fill rate). The density of the matrix results from the integral coupling due to the moment quan-

tities that appear as coefficients of the derivative terms. The storage of this large dense matrix makes most interesting applications prohibitively expensive in terms of memory. The matrix storage problem could be dealt with by simply evaluating the integral quantities at previous iteration values during the iterations. However, as the integral terms become important, this lagging of the integral quantities can significantly affect convergence, as will be demonstrated in Section 5. What is desired is a method that does not require the storage of the large dense matrix, but still has all of its effects during all the iteration steps. The ideal method would approximate the action of the Jacobian matrix without having to build or store the Jacobian matrix. Matrix-free Newton–Krylov is one such method.

We re-emphasize that the proposed solution algorithm does not lag the integral terms in the nonlinear (outer) iteration. The integral terms are lagged in the preconditioner and thus only the linear (inner) iteration is effected.

4.1. *Newton's Method*

The nonlinearity of the problem comes from the electric field and the $(\partial F/\partial t)|_{\text{col}}$ term in Eq. (2). To deal with this nonlinearity Newton's method is used. The main advantage of Newton's method is its quadratic convergence. Its drawbacks, however, are that the initial guess must be within the Newton radius of convergence and it may require many linear solves. One can get an initial guess within the radius of convergence by using a restart from a similar problem. Other methods for increasing the Newton radius of convergence are pseudo-transients, damping strategies, and mesh sequencing [26, 27]. We will discuss these in more detail in the following paragraphs.

Newton's method is a powerful technique for solving systems of nonlinear equations of the form

$$\mathbf{G}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), ..., g_n(\mathbf{x})]^{\mathrm{T}} = 0 \tag{33}$$

where the state vector, $\mathbf{x}$, can be expressed as

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^{\mathrm{T}}. \tag{34}$$

Application of Newton's method requires the solution of the linear system

$$\mathbf{J}^n\, \delta\mathbf{x}^n = -\mathbf{G}(\mathbf{x}^n) \tag{35}$$

where the elements of the Jacobian, $\mathbf{J}$, are defined by

$$\mathbf{J}^n_{i,j} = \frac{\partial g_i}{\partial \mathbf{x}^n_j} \tag{36}$$

and the new solution approximation is obtained from

$$\mathbf{x}^{n+1} = \mathbf{x}^n + d\delta\mathbf{x}^n. \tag{37}$$

The constant, $d \in [0, 1]$, in Eq. (37) is used to damp the Newton updates. The damping strategy is designed to prevent the calculation of nonphysical variable values (i.e., negative distribution functions) and to scale large variable updates when the solution is far from the true solution. This iteration is continued until the norm of $\delta\mathbf{x}$ and/or the norm of $\mathbf{G}(\mathbf{x})$ are below some suitable tolerance level. Construction of the Jacobian matrix can be one of the most difficult tasks in implementing Newton's method. The method which is employed in this study is to approximate the Jacobian with finite differences. This method was chosen because it results in very modular code that is easily modified and maintained.

An inexact Newton's method [28] is used for increased efficiency. This technique adjusts the convergence tolerance for the linear solve of Eq. (35) with the nonlinear iteration,

$$\frac{\|\mathbf{J}^n\, \delta\mathbf{x}^n + \mathbf{G}(x^n)\|_2}{\|\mathbf{G}(\mathbf{x}^n)\|_2} < \gamma, \tag{38}$$

and thus increases efficiency by loosening the convergence tolerance when the nonlinear iteration is ''far'' from the solution where high accuracy does not benefit the global convergence. The selection of the best value for γ is empirical, but we have found values in the range of 10^{-3}–10^{-2} to work well for our problem.

To increase the radius of convergence of Newton's method a pseudo-transient is employed. In this method the Jacobian is modified as follows:

$$\left(\frac{\mathbf{I}}{\Delta t} + \mathbf{J}^n\right) \delta\mathbf{x}^n = -\mathbf{G}(\mathbf{x}^n). \tag{39}$$

By increasing the diagonal of the Jacobian, the size of the Newton update $\delta\mathbf{x}^k$ is lowered and the iteration is damped. Additionally, the increased diagonal will serve to decrease the condition number of the matrix and thus improve the performance of the Krylov method. However, this will most likely come at the cost of increased Newton iterations. The time step, Δt, is allowed to increase as the steady-state residual decreases. This is accomplished by the algorithm

$$\Delta t^n = \Delta t^0 \frac{\|\mathbf{G}(\mathbf{x}^0)\|_\infty}{\|\mathbf{G}(\mathbf{x}^{n-1})\|_\infty}. \tag{40}$$

Here the superscript 0 refers to the initial Newton iteration.

Mesh sequencing is used to get an initial guess inside of the Newton radius of convergence by interpolating a

converged solution from a coarse grid where the computational costs are low to a fine grid. Here the coarse grid distribution function values are linearly interpolated to get the new fine grid distribution function values. More advanced grid interpolation schemes will be investigated in the future. Using this technique places the largest number of Newton iterations on the coarse grids where the cost per iteration is lowest and the radius of convergence is largest.

4.2. *Krylov Method*

Krylov techniques are used as the iterative method for solving Eq. (35). There are many Krylov, conjugate gradient-like, solvers [29, 30], but we have focused our attention on the Generalized Minimal Residual (GMRES) method [31]. This is because GMRES has been shown to be more robust in matrix-free implementations [32, 33] than other Krylov algorithms and our solution method requires a matrix-free implementation. Since the memory requirements of the GMRES algorithm increases with iteration, we want to minimize the number of linear iterations per Newton step. To lower the number of linear iterations we precondition the original matrix system. Preconditioning reduces the spread in eigenvalues (the farthest distance between two eigenvalues) and/or clusters the eigenvalues (reduces the total number of unique eigenvalues) of the matrix, both of which increase the performance of the Krylov method. In this study we use Incomplete Lower–Upper factorization [9], ILU(k), as our preconditioner.

4.3. *Matrix-Free Approximation*

To solve Eq. (35), the GMRES algorithm requires the action of the Jacobian only in the form of matrix vector products, not the acutal Jacobian matrix itself. Because of the large size and density of the Jacobian matrix, caused by the integral coupling, one would like to find a way to approximate the Jacobian's action. This can be done using the so-called matrix-free or reduced storage approximation [8, 34],

$$\mathbf{Jv} \approx \frac{\mathbf{G}(\mathbf{x} + \varepsilon\mathbf{v}) - \mathbf{G}(\mathbf{x})}{\varepsilon}. \tag{41}$$

Here $\mathbf{v}$ is a vector in the Krylov iteration and ε is a small perturbation. The perturbation is calculated as

$$\varepsilon = \frac{\Sigma[(1 \times 10^{-8}) \times F(x, v)]}{N\|\mathbf{v}\|_2}, \tag{42}$$

where N is the total number of unknowns (i.e., $N = nx \times ny$, where nx is the number of discrete volumes in the x direction and ny is the number of discrete volumes in the v direction). It should be noted that the ε used in Eq. (42) represents an average of all of the individual epsilons that would have been used in evaluating a complete numerical Jacobian [33].

By using this approximation in the Krylov iteration, one is not required to form the full Jacobian matrix on each Newton step, but rather just that part which is required for a preconditioner. Thus, the large computational cost of evaluating each element in the Jacobian via Eq. (36) is replaced with one additional right-hand-side evaluation per GMRES iteration. This results in a vast reduction in storage requirements since the Jacobian matrix is never stored.

With preconditioning, Eq. (41) has the form

$$\mathbf{JPv} \approx \frac{\mathbf{G}(\mathbf{x} + \varepsilon\mathbf{Pv}) - \mathbf{G}(\mathbf{x})}{\varepsilon}. \tag{43}$$

The preconditioning matrix $\mathbf{P}$ is formed by evaluating integral terms at the state $\mathbf{x}^n$ (i.e., lagged). Thus, while we employ the matrix-free Newton–Krylov iteration on the equation

$$\begin{aligned} v\frac{\partial F^{n+1}}{\partial x} + \frac{qE_x^{n+1}}{m}\frac{\partial F^{n+1}}{\partial v} = \nu_{\text{col}}\Bigg\{\left[\frac{kT^{n+1}}{m}\right]\frac{\partial^2 F^{n+1}}{\partial v^2} \\ + \frac{\partial[(v - \overline{V}^{n+1})F^{n+1}]}{\partial v}\Bigg\} + \left.\frac{\partial F}{\partial t}\right|_{\text{neut}} + \left.\frac{\partial F}{\partial t}\right|_{\text{src}}, \end{aligned} \tag{44}$$

the preconditioner is constructed from the Jacobian of the same equation with T^n, E_x^n, and $\overline{V}^n$ at the old iteration level. Since the integral quantities are evaluated at the last iteration value, the preconditioner matrix has a five or nine diagonal structure, which results from the five (upwind) or nine (QUICK) point stencil of the difference operators (see Section 3). Therefore, when we use ILU(0) as the preconditioner, the total preconditioner matrix storage is $5 \times N$ or $9 \times N$, where N is the total number of unknowns.

5. MODEL PROBLEM AND RESULTS

To demonstrate the numerical method we have chosen a 1D–1V model problem. The purpose of this model problem is not to study kinetic effects, but rather to study algorithm performance. In fact, the model problem is in a density, temperature regime where collisional effects dominate. The problem is a one-dimensional representation of a high recycling divertor along the magnetic field line.

Ions and energy are injected into the source region, according to Eq. (6) at a temperature of 75 eV, and flow out of the problem at the divertor plate, $x = 6$ m. The injection temperature was used to keep the upstream temperature at approximately 30 eV. The maximum particle velocity is set at 1.5×10^5 m/s, which is approximately

TABLE I

Input Values for Model Problem

nx	100	ny	200	λ_n	0.5 m
L	6.0 m	v_{max}	1.5×10^5 m/s	T_{inj}	75.0 eV
ν_{col}	2.5×10^5 1/s	m	2 AMU	R_{inj}	$3.367 \times 10^{27}\text{m}^{-3}\text{s}^{-1}$
v_n	3.5×10^4 m/s	$n_n(L)$	$6.122 \times 10^{19}\text{m}^{-3}$	L_{inj}	4.56 m

$3 \times v_{th}$, for the conditions at the symmetry plane. For this first problem, a constant collision frequency is used, and $v_{th} = (2kT/m)^{0.5} \approx 5.0 \times 10^4$ m/s for $T = 30$ eV. $n_n(L)$ was chosen to give a high recycling solution, and $\nu_{ie} = v_n/\lambda_n$. Here the neutral model is set to give a source of cold particles that ionize in about one third of the computational region. The rest of the input quantities are defined in Table I.

In Fig. 2 we see the neutral particle profile which results from Eq. (4). These neutrals are ionized and feed into Eqs. (2) through (5). Figure 3 shows the electric field computed by Eq. (12). Here we can see the increase in the electric field in the pre-sheath region near the wall.

In Figs. 4 through 7 we see the moment quantities defined by Eqs. (8) through (10) and Eq. (13), respectively. In these figures we can see the classic high recycling divertor structure of a density bump and a declining temperature [26, 27]. Figure 8 shows constant "x" slices of the distribution function. Here we can see acceleration of the flow as we move across the simulation towards the divertor, shown by the shifting of the distribution function towards higher velocity. We can also see the initial rise in number density, the peak at $x = 4.5$, and the drop as we approach the wall, $x = 6.0$. Figure 8 also demonstrates the non-Maxwellian behavior caused by the ionization source at $x = 5.94$.

Figure 9 shows the convergence, in terms of update size $\delta\mathbf{x}^n$ in Eq. (37), per Newton step for a four mesh sequence of uniform grids. For this run, ILU (3) is the preconditioner, and GMRES(120) is the Krylov solve. This results in a memory usage of 85.12 megabytes for the largest (200×400) grid. Here we can clearly see that mesh sequencing reduces the number of Newton iterations on the finer meshes. The strong oscillations in the coarse grid convergence are a function of the pseudo-transient progression, the damping algorithm, and the time step control.

Figure 10 shows the convergence for the 200×400 mesh as a function of CPU time on an HP 735 workstation. This illustrates the importance of good initial guesses for the fine grids where each Newton iteration takes significantly more time than the coarse grid iterations. Table II shows the Newton iterations, CPU time, and the cumulative CPU time for the entire mesh sequenced run. The cumulative CPU time shows a running total of all of the meshes CPU time combined. From Fig. 10 we can see that as the solution

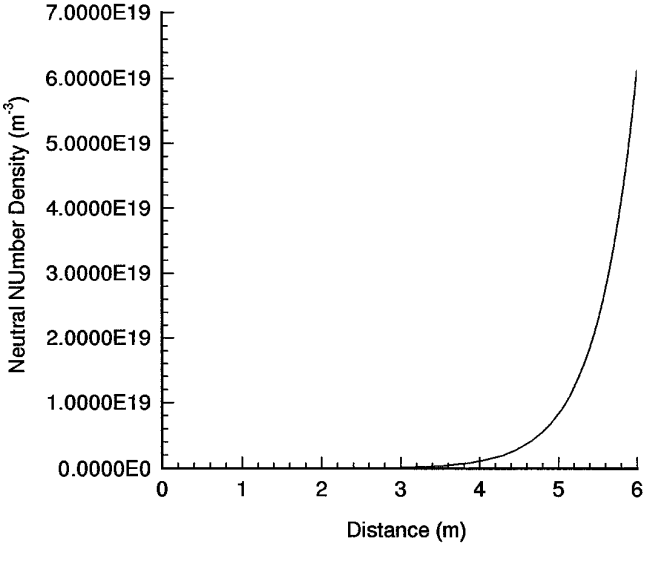

FIG. 2. Neutral number density.

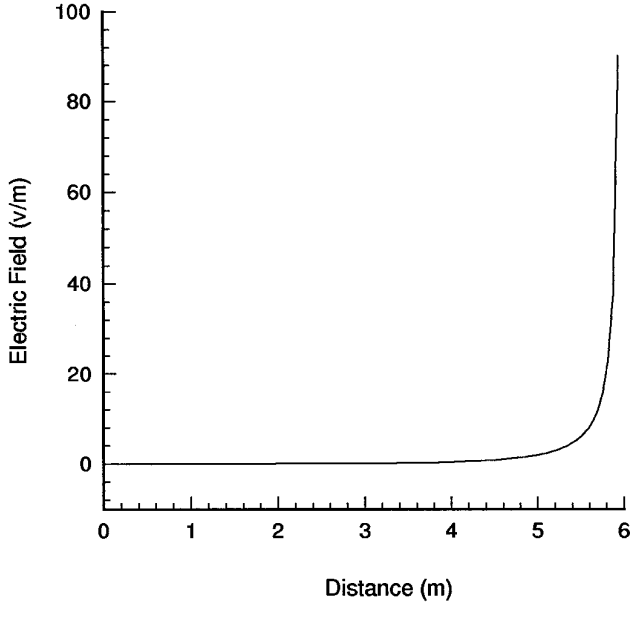

FIG. 3. Electric field.

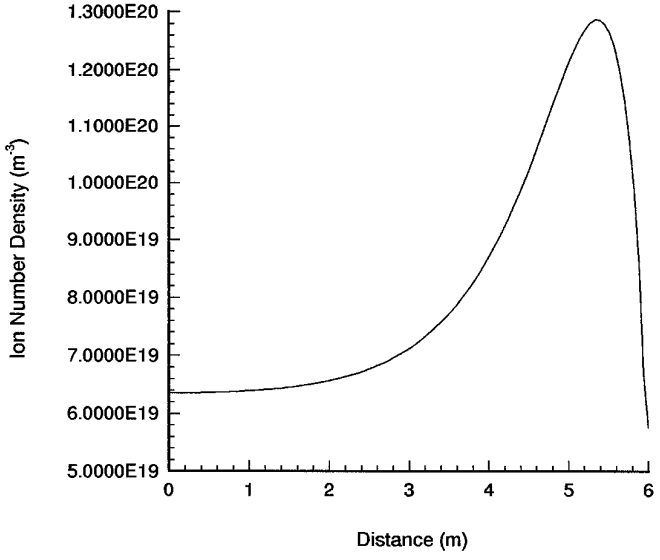

FIG. 4. Ion number density.

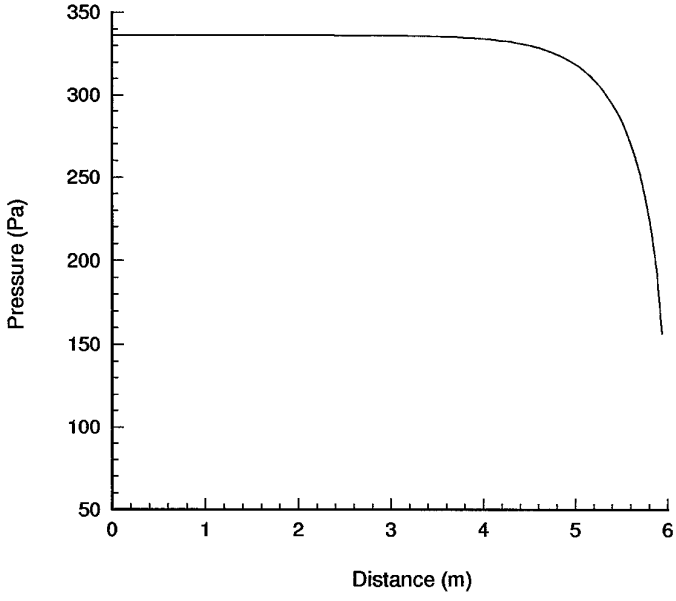

FIG. 6. Fluid pressure.

is approached, the CPU cost per Newton iteration goes up. This is caused by the Krylov convergence tolerance from Eq. (38).

Figure 11 shows the effect of mesh refinement on the fluid pressure. It is clear that the solution is beginning to converge. These results indicate the need for higher-order differencing of the first order operators, which are currently differenced with first-order upwind everywhere except in the $(\partial F/\partial t)|_{\text{col}}$ term where interpolated donor cell differencing is used. Higher-order differencing as well as a nonuniform grid will aid in limiting the required mesh size for accuracy.

Figure 12 shows results similar to those in Fig. 11 for a mesh sequenced set of runs which uses nonuniform mesh spacing in the x direction (i.e., smaller spacing near the wall), flux-limited QUICK for the Vlasov operator, and Chang–Cooper differencing of the Collision operator. It should be noted that the advanced differencing could have included a nonuniform grid spacing in the v direction but optimal refinement of the velocity dimension has not yet been incorporated. However, the nonuniform spacing in the x direction resolves the steep gradient near the wall

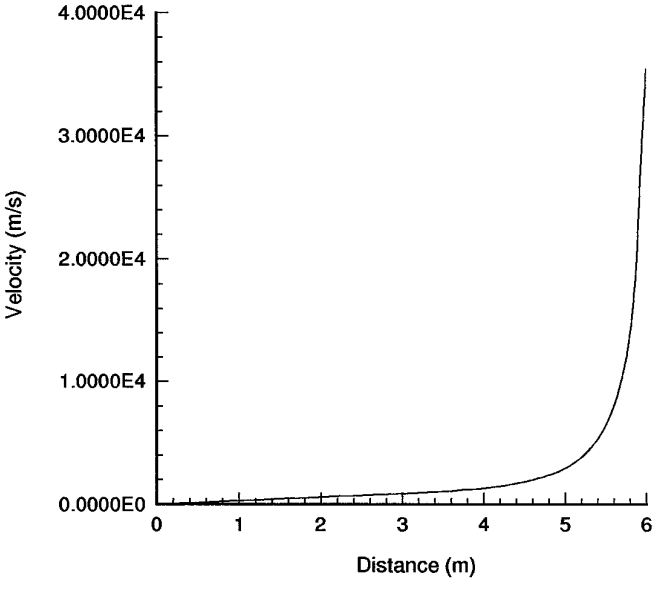

FIG. 5. Fluid velocity.

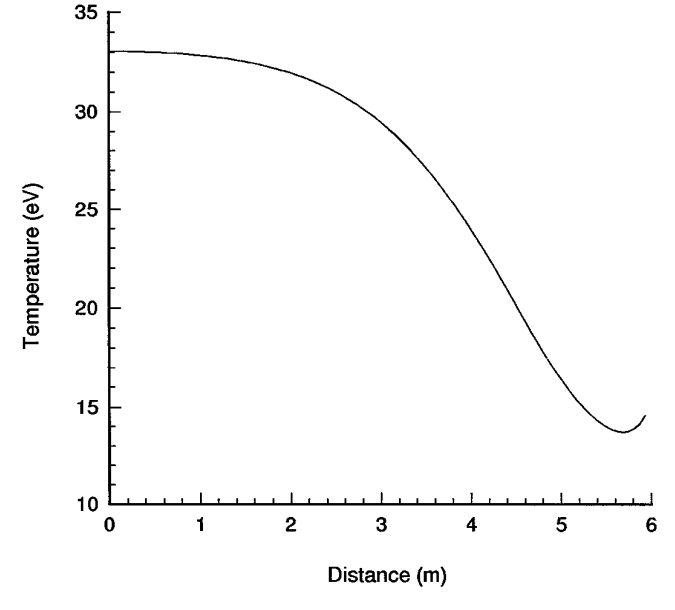

FIG. 7. Fluid temperature.

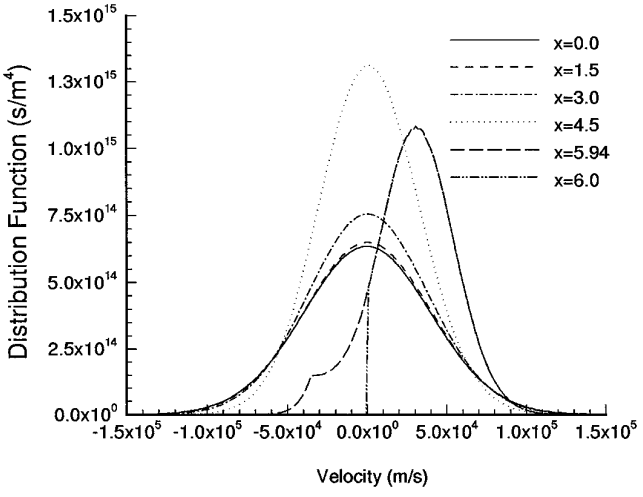

FIG. 8. Distribution function slices.

with a coarse mesh and clearly improves accuracy. Here one can see that with advanced differencing a mesh independent solution is more quickly obtained.

Figure 13 shows the cost that has to be paid for the higher accuracy. This plot is taken from the 25 × 50 run. Clearly, the total number of Newton iterations have increased compared to the 25 × 50 line in Fig. 9. The CPU time has also increased from 24 s for the 25 × 50 run shown in Table II to 260 s. It should be noted however that the advanced differencing work has not been optimized to the level that the upwind and interpolated donor cell differencing solutions have. Also, one can see that an accurate solution is obtained even for the 25 × 50 grid problem

TABLE II

Newton and CPU (HP 735) Performance for a Four Grid Sequence

Mesh	Newton iterations	CPU time (s)	Cumulative CPU time (s)
25 × 50	33	24	24
50 × 100	15	66	90
100 × 200	12	305	395
200 × 400	10	1682	2077

using advanced differencing, Fig. 12, while clearly the 25 × 50 grid solution is unacceptable for the first order differencing, Fig. 11. In fact, the 25 × 50 high-order solution taking 260 s is comparable in accuracy to the 100 × 200 first-order solution taking nearly 400 s.

As the collision frequency ν is increased, the integral coupling, due to the T and $\overline{V}$ in the collision operator, becomes more important. Figures 14 and 15 show the advantage of the matrix-free Newton–Krylov method when the integral coupling becomes more important. From these figures one can see that both the CPU time and the total number of iterations rapidly increase for the lagged matrix solution (i.e., T and $\overline{V}$ at the last iteration value). But the matrix-free approximation makes the computation time almost independent of the collision frequency.

As mentioned previously, the unique capability of this solution algorithm is that it can retain implicit coupling of integral terms without the memory requirements of storing these terms in the Jacobian. Figures 16 and 17 show the

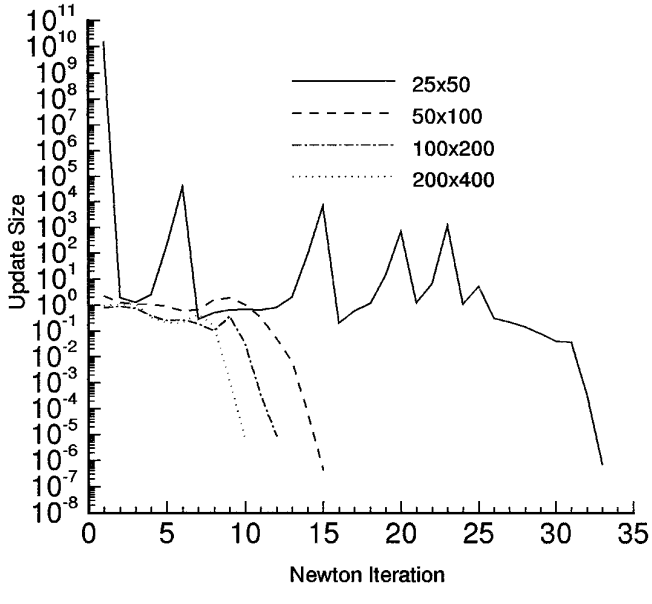

FIG. 9. Mesh sequenced convergence vs Newton iteration.

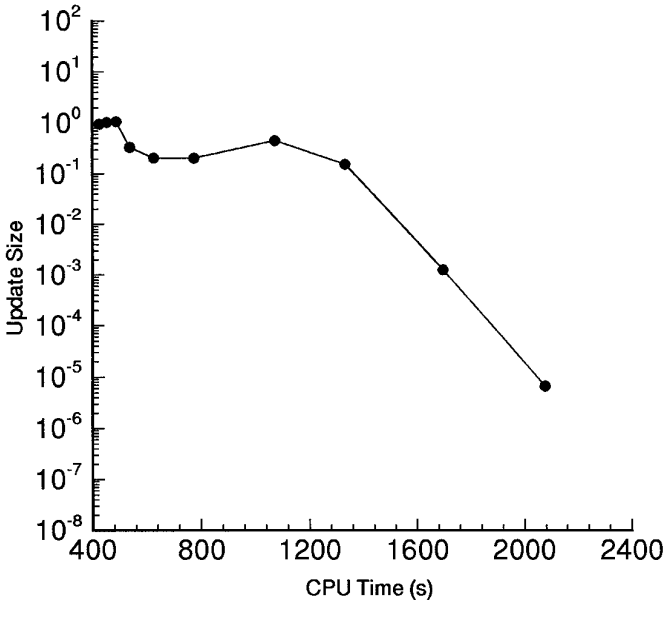

FIG. 10. 200 × 400 convergence vs CPU time (HP 735).

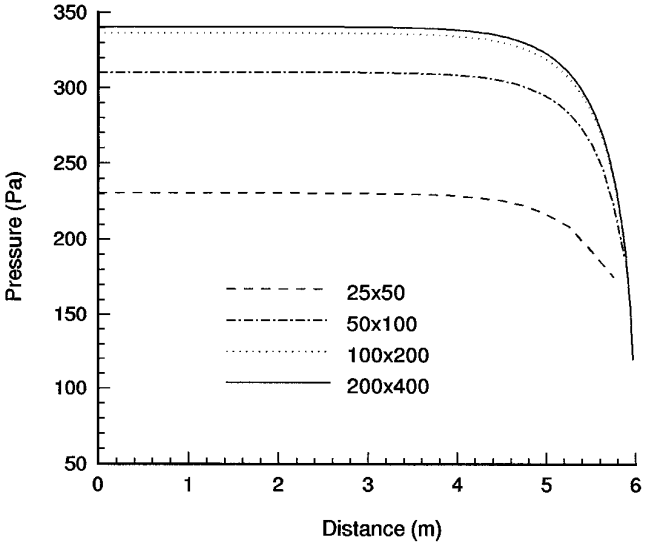

FIG. 11. Mesh sequenced pressure.

importance of keeping the integral quantities implicit for convergence on the 25×50 grid. In the matrix-free option all integral quantities are implicit and it takes only 33 iterations to converge compared to 1975 when the integral quantities are evaluated at the old iteration value. Although it costs more in terms of CPU time per iteration, one can clearly see from Fig. 16 that the matrix free option is more computationally efficient by an order of magnitude. In both cases the initial guess was Maxwellian with a flat temperature and density profile. The severe oscillations during the early iterations of the matrix-free implementation suggest that a hybrid method, were the implicit integral coupling is included only after some initial iterations, may be a good compromise between robustness and efficiency.

For the first problem a constant collision frequency was used in Eq. (3). Next we will increase the nonlinearity and integral coupling by making the collision frequency a local function of T and n ($\nu_{\text{col}} = K_1\, n/T^{3/2}$, where K_1 is a constant). Figure 18 shows both the matrix free and the lagged integral convergence for the x dependent ν_{col} in terms of CPU time. Again we see that the matrix free solution requires significantly less time to obtain the same steady state solution. It should be noted that the collision frequency is lower for the x dependent ν_{col}. This lower collision frequency reduced the importance of the integral coupling which resulted in an easier problem to solve. To get a more accurate collision operator, and a more correct representation of time scales, we also plan to include a dependence on v in ν_{col} in the future (i.e., $\nu_{\text{col}} \sim 1/v^3$).

Finally, we demonstrate performance sensitivity to preconditioning, pseudo-transient implementation, and Krylov convergence tolerance, γ. Figure 19 shows the effect of the level of ILU fill [35] on the average number of Krylov iterations per Newton iteration. The number of nonzero diagonals required is 5, 13, and 45 for ILU(0), ILU(3), and ILU(6) respectively. The simulation is a restart from the base problem, see Table I, with ν_{col} in Eq. (3) perturbed from 2.5×10^5 to 2.4×10^5. This is the same problem used to generate Figs. 20 through Fig. 22. The plot clearly shows how the number of linear iterations increases when ILU(0) is used as a preconditioner as a

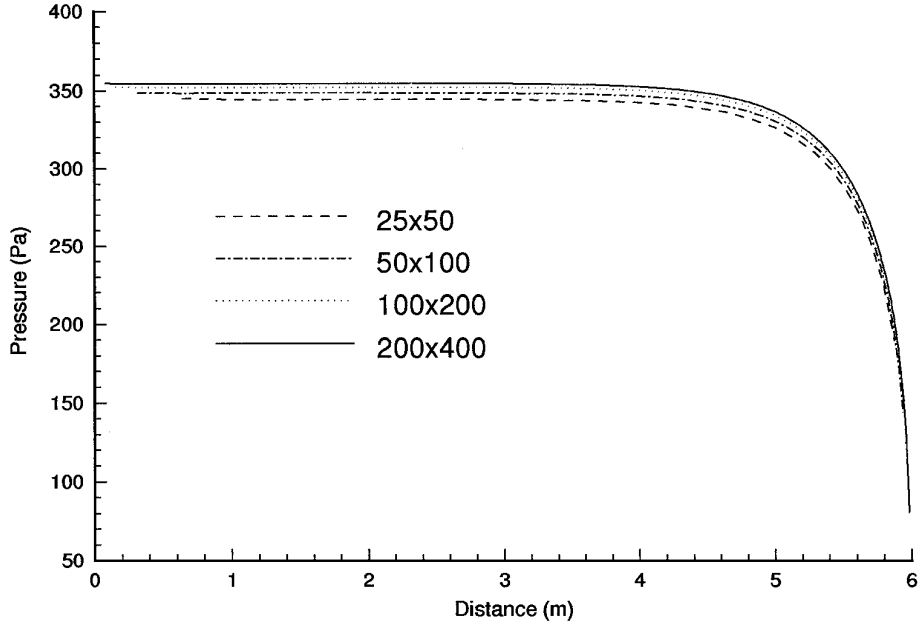

FIG. 12. Advanced discretization mesh sequenced pressure.

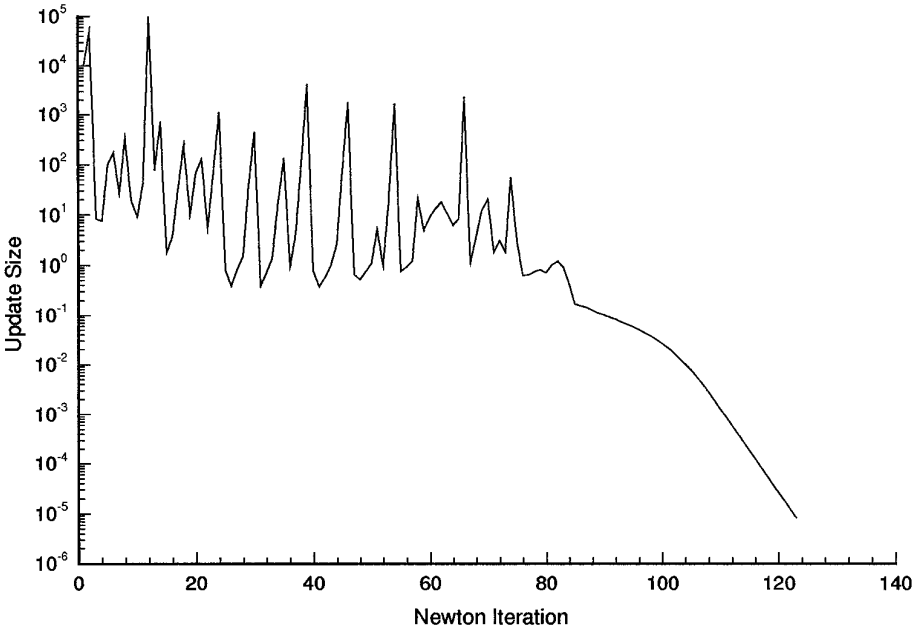

FIG. 13. Advanced discretization update size vs Newton iteration.

function of problem size. The GMRES algorithm has a maximum iteration count built into the solution. When GMRES meets this limit, the last iteration is used as an initial guess and the algorithm is restarted. For this run the maximum iteration limit for GMRES is set at 480. This maximum iteration limit is set high to allow the weakest preconditioner, ILU(0), to converge on the largest problem, 100×200. In the 100×200 ILU(0) problem, this limit is attained in two of the four Newton iterations, but the problem still converges. Figure 19 also demonstrates that higher levels of ILU fill-in, although requiring more memory and CPU time, scale better with the problem size than ILU(0). Figure 20 shows the effect of level of ILU fill-in on the run time (note that the CPU time is on a log

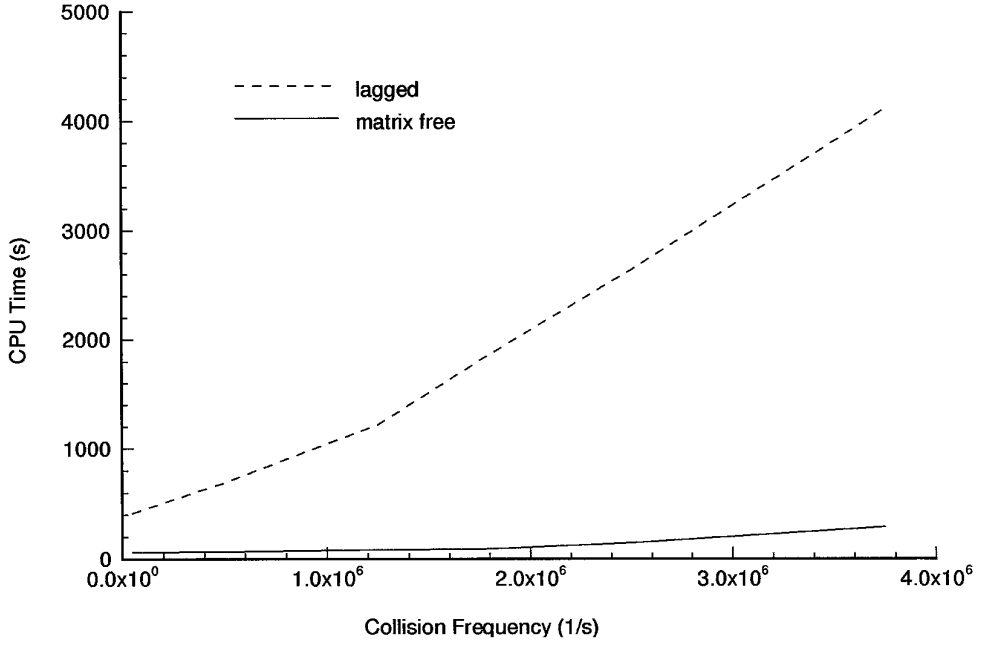

FIG. 14. Matrix free and lagged integral vs collision frequency CPU time (HP 735).

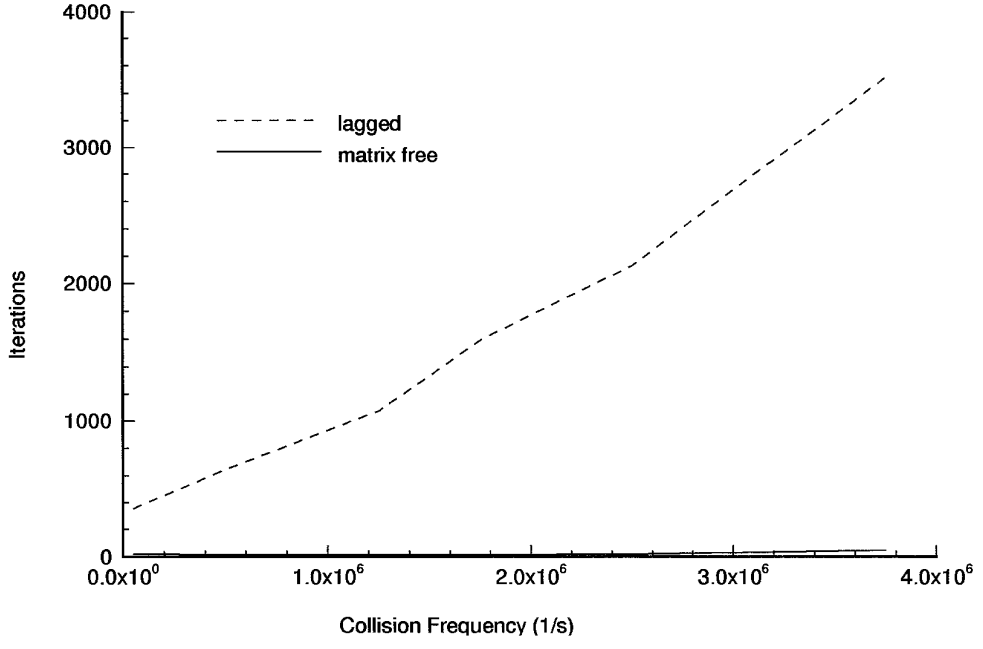

FIG. 15. Matrix free and lagged integral vs collision frequency Newton iterations.

scale). Here we can see that the most powerful preconditioner ILU(6) is also the most effective for the larger grids.

The effect of time step size, Δt from Eq. (39), on the Krylov iteration and the run time is presented in Fig. 21 for a 50×100 grid. Here one can see that as the time step size increases the CPU time decreases, even though the number of iterations per Newton step increases. Recall that since this is a pseudo transient and not a "real" transient it does not take 100 times as many steps to reach steady state for $\Delta t = 1.0 \times 10^{-5}$ as $\Delta t = 1.0 \times 10^{-3}$ (the actual numbers are 7 and 152 or a ratio of about 22).

The effect of the Krylov convergence tolerance, γ from Eq. (38), on the number of Krylov iterations is presented in Fig. 22 on a 50×100 grid. Results are presented for two

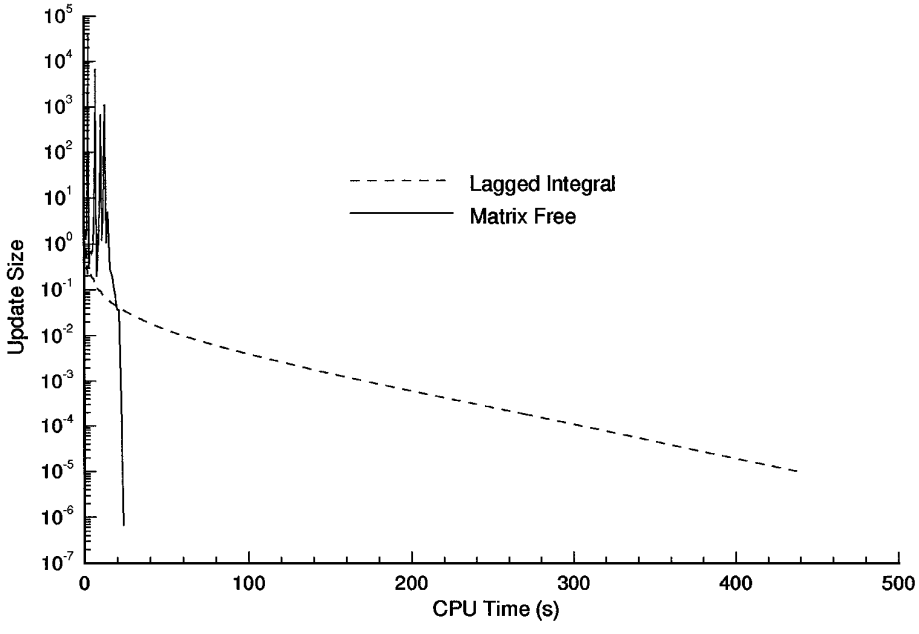

FIG. 16. Matrix free vs lagged integral CPU time (HP 735).

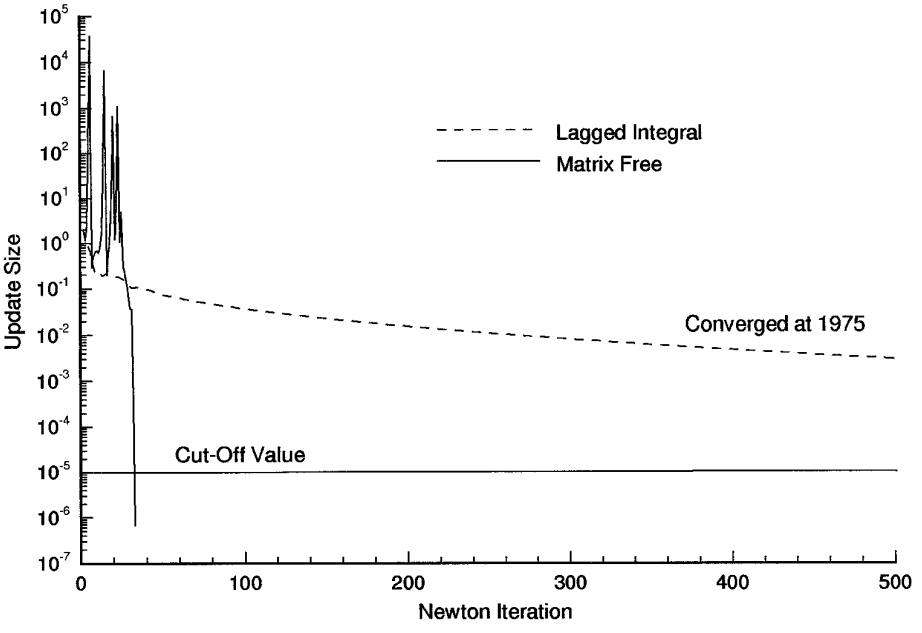

FIG. 17. Matrix free vs lagged integral Newton iterations.

different Krylov solvers, GMRES(40) and GMRES(120), where the maximum dimension of the Krylov space are 40 and 120, respectively. Since the GMRES algorithm's memory usage and work increase with the number of iterations, an upper bound is set (i.e., 40 or 120). If the number of iterations exceeds the preset limit the algorithm is restarted with an initial guess constructed from the existing Krylov vectors [31]. Here we show the effect of restarting the GMRES algorithm. With the maximum Krylov dimension set to 120 there are no restarts of the GMRES algorithm. When the maximum is set to 40 the algorithm has to restart often. When the GMRES algorithm is restarted the convergence rate is slowed and sometimes stalls [32, 33]. Thus, there must be a trade-off between accuracy of the linear solve, preconditioning, and memory requirements for the GMRES algorithm.

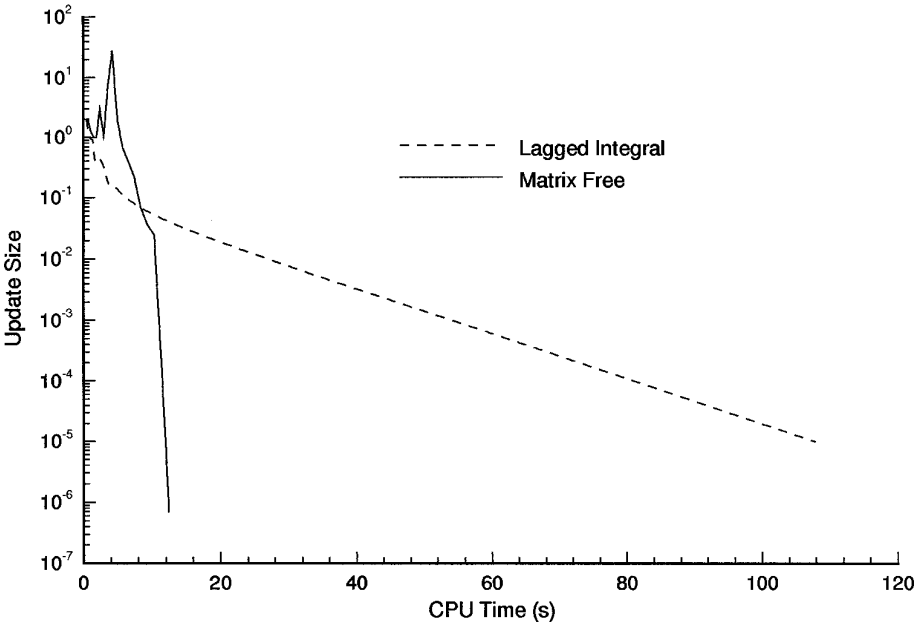

FIG. 18. Matrix free vs lagged integral CPU time (HP 735) for variable collision frequencies.

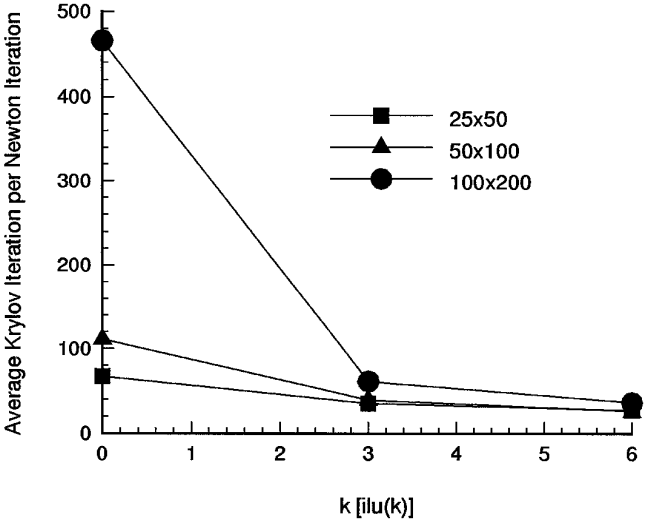

FIG. 19. Average Krylov iteration vs level of ILU fill-in.

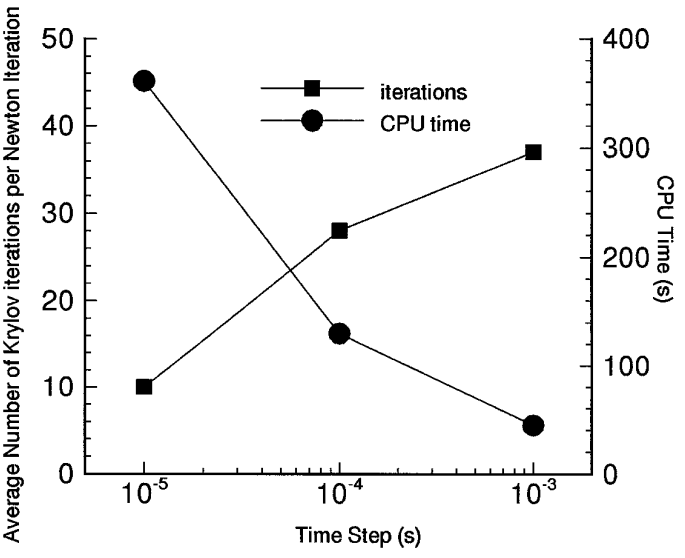

FIG. 21. Average Krylov iteration and run time vs time steps size.

6. SUMMARY AND CONCLUSIONS

We have demonstrated that the matrix-free Newton–Krylov method is a viable option for fully implicitly solving the Vlasov equation with a simplified Fokker–Planck collision operator. The matrix-free option, which eliminates the need for forming the full Jacobian matrix, drastically reduces the storage requirements of a simulation. For example, for a sample problem of 100 velocity cells and 50 spatial cells, we reduce the number of nonzero elements that need to be stored from 7.549×10^5 for the complete Jacobian, to less than 2.5×10^4 for the preconditioner. We have also demonstrated that including the integral terms implicitly significantly accelerates convergence as compared with lagging them at old time values. Because this method is fully implicit it can handle multiple time scales effectively, as was demonstrated by solving the steady-state equation directly. Therefore, there may be problems that can be solved using this technique, such as coupled ion–electron problems, which are very difficult to solve using other numerical integration methods. This method does not have any assumptions which preclude moving to higher dimensions although this work has not yet been initiated. We have demonstrated that the method works with monotone higher-order differencing of the convective operators (flux-limited QUICK). This method has also been applied to coupled ion–electron kinetic solutions with

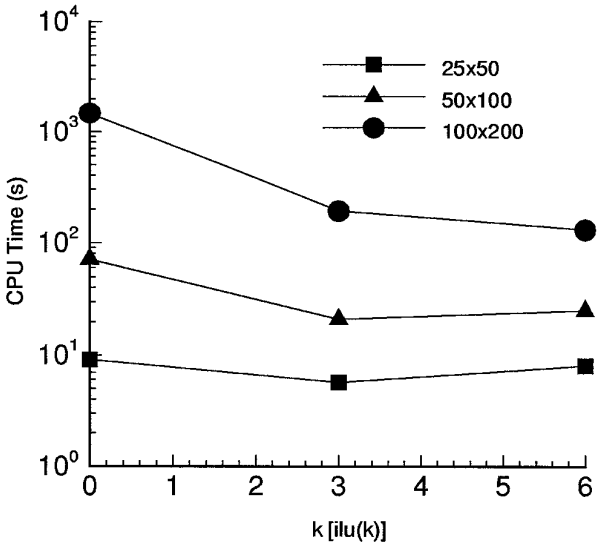

FIG. 20. Run time vs level of ILU fill in for different grids.

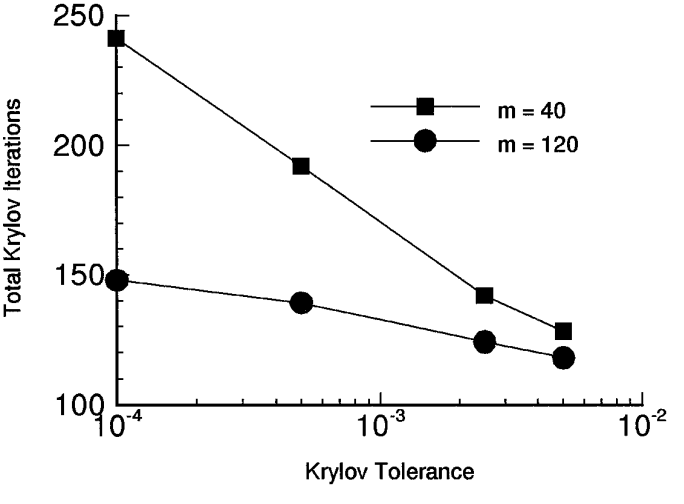

FIG. 22. Total Krylov iteration vs Krylov tolerance, γ, for different maximum Krylov space dimensions.

a self consistent electric field [20]. More information on the advanced differencing and the coupled ion–electron problem will be presented in future work.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Dept. of Energy (DOE), Idaho Operations Office, under DOE Contract No. DE-AC07-94ID13223, and supported by the Idaho National Engineering Laboratory Long Term Research Initiative in Computational Mechanics and the Laboratory Directed Research and Development.

REFERENCES

1. S. I. Braginskii, Transport processes in a plasma, in *Reviews of Plasma Physics* (Consultants Bureau, New York, 1965).
2. L. M. Montierth, R. L. Morse, and W. A. Neuman, Collisional transport treatment of surface plasma structures, *Phys. Fluids B* **1,** 1911 (1989).
3. A. A. Batishcheva *et al.*, Fokker–Planck simulation of electron transport in SOL plasmas with ALLA code, *Contrib. Plasma Phys.* **36,** 235 (1996).
4. A. A. Batishcheva *et al.,* Massively parallel Fokker–Planck code ALLAP, *Contrib. Plasma Phys.* **36,** 414 (1996).
5. A. R. Bell, Non-Spitzer heat flow in a steadily ablating laser-produced plasma, *Phys. Fluids* **28,** 2007 (1985).
6. M. E. Riley, K. E. Greenberg, G. A. Hebner, and P. J. Drallos, Theoretical and experimental study of low temperature capacitively coupled RF helium plasmas, *J. Appl. Phys.* **75,** 2789 (1994).
7. D. Loffhagen and R. Winkler, A new nonstationary Boltzmann solver in self-consistent modelling of discharge pumped plasmas for excimer lasers, *J. Comput. Phys.* **112,** 91 (1994).
8. P. N. Brown and A. C. Hindmarsh, Matrix-free methods for stiff systems of ODE's, *SIAM J. Numer. Anal.* **23,** 610 (1986).
9. J. A. Meijerink and H. A. van der Vorst, Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems, *J. Comput. Phys.* **44,** 134 (1981).
10. R. J. Procassini, C. K. Birdsall, and B. I. Cohen, Particle simulations of collisional transport in a high recycling, diverted tokamak scrape-off layer, *Nucl. Fus.* **30,** 2329 (1990).
11. O. V. Batischev, S. I. Krasheninnikov, D. J. Sigmar, Yu. S. Sigov, and T. K. Soboleva, Influence of kinetic effects on particle and energy flows in the ITER divertor, *Contrib. Plasma Phys.* **34,** 436 (1994).
12. D. P. Coster, *Tokamak Divertor Modeling with Fluid and Kinetic Codes,* Ph.D. thesis (Princeton University, 1993).
13. J. D. McCullen, L. M. Montierth, R. L. Morse, and W. A. Neuman, Surface plasma structures in the kinetic regime, *Phys. Fluids B* **1,** 448 (1989).
14. R. J. Procassini and C. K. Birdsall, Particle simulation model of transport in a bounded, Coulomb collisional plasma, *Phys. Fluids B* **3,** 1876 (1991).
15. P. J. Catto. [Personal communication]
16. L. R. T. Gardner, G. A. Gardner, and S. I. Zaki, Collisional effects in plasmas modelled by a simplified Fokker–Planck equation, *J. Comput. Phys.* **107,** 40 (1993).
17. A. Lenard and I. B. Berstein, Plasma oscillations with diffusion in velocity space, *Phys. Rev.* **112,** 1456 (1958).
18. C. E. Rathmann and J. Denavit, Simulation of collisional effects in plasmas, *J. Comput. Phys.* **18,** 165 (1975).
19. G. A. Emmert, R. M. Wieland, A. T. Mense, and J. N. Davidson, Electric sheath and presheath in a collisionless, finite ion temperature plasma, *Phys. Fluids* **23,** 803 (1980).
20. V. A. Mousseau, *Fully Implicit Kinetic Modelling of Collisional Plasmas,* Ph.D. thesis (University of Idaho, 1996).
21. E. M. Epperlein, Implicit and conservative difference scheme for the Fokker–Planck equation, *J. Comput. Phys.* **112,** 291 (1994).
22. F. H. Harlow and A. A. Amsden, Numerical calculation of multiphase fluid flow, *J. Comput. Phys.* **17,** 19 (1975).
23. P. H. Gaskell and A. K. C. Lau, Curvature-compensated convective transport: SMART, a new boundedness-preserving transport algorithm, *Int. J. Numer. Methods Fluids* **8,** 617 (1988).
24. B. P. Leonard and J. E. Drummond, Why you should not use ''Hybrid,'' ''Power-Law'' or related exponential schemes for convection modelling—There are much better alternatives, *Int. J. Numer. Methods Fluids* **20,** 421 (1995).
25. J. S. Chang and G. Cooper, A practical difference scheme for Fokker–Planck equations, *J. Comput. Phys.* **6,** 1 (1970).
26. Dana A. Knoll, A. K. Prinja, and R. B. Campbell, A direct Newton solver for the two-dimensional tokamak edge plasma fluid equations, *J. Comput. Phys.* **104,** 418 (1993).
27. D. A. Knoll and P. R. McHugh, An inexact Newton algorithm for solving the tokamak edge plasma fluid equations on a multiply-connected domain, *J. Comput. Phys.* **116,** 281 (1995).
28. R. Dembo, S. C. Eisenstat, and T. Steihaug, Inexact Newton methods, *SIAM J. Numer. Anal.* **19,** 400 (1982).
29. R. W. Freund *et al.,* Iterative solution of linear systems, *Acta Numer.,* 57 (1991).
30. P. R. McHugh, *An Investigation of Newton–Krylov Algorithms for Solving Incompressible and Low Mach Number Compressible Fluid Flow and Heat Transfer Problems Using Finite Volume Discretization,* Ph.D. thesis (University of Idaho, 1995).
31. Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems, *SIAM J. Sci. Stat. Comput.* **7,** 856 (1986).
32. D. A. Knoll, P. R. McHugh, and V. A. Mousseau, Newton–Krylov–Schwarz methods applied to the tokamak edge plasma fluid equations, in *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering,* (SIAM, 1995).
33. Paul R. McHugh and Dana A. Knoll, Inexact Newton's method solutions to the incompressible Navier–Stokes and energy equations using standard and matrix-free implementations, *AIAA J.* **34,** 2394 (1994).
34. P. N. Brown and Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, *SIAM J. Sci. Stat. Comput.* **11,** 450 (1990).
35. J. W. Watts, A conjugate gradient-truncated direct method for the iterative solution of the reservoir simulation pressure equation, *Soc. Petr. Eng. J.,* 345 (1981).