Journal of Computational Physics **161,** 589–604 (2000)
doi:10.1006/jcph.2000.6518, available online at http://www.idealibrary.com on IDEAL®

Massively Parallel Three-Dimensional Toroidal Gyrokinetic Flux-Tube Turbulence Simulation

Charlson C. Kim and Scott E. Parker

Center for Integrated Plasma Studies, Department of Physics, University of Colorado, Boulder, Colorado 80309

Received December 27, 1999; revised March 27, 2000

A massively parallel three-dimensional nonlinear gyrokinetic flux-tube simulation model is discussed. This simulation is used to study turbulent heat transport in core tokamak fusion plasmas. This model allows for high resolution simulations of ion-temperature-gradient-driven turbulence using realistic plasma parameters assuming locality of the turbulent fluctuations. The simulation model, computational techniques, and parallel algorithms are discussed. The use of field-aligned coordinates allows for a natural domain decomposition in the direction along the magnetic field with good parallel performance. Digital filtering along the field line maintains proper toroidal and poloidal periodicity. A new approach to parallelization, "domain cloning," is presented. Domain cloning is another layer of parallelization. It is an alternative to a two-dimensional domain decomposition and may be useful for clustered symmetric-multiprocessor machines. Performance results are presented for two high-performance massively parallel computers. © 2000 Academic Press

1. INTRODUCTION

There has recently been significant progress in kinetic simulation of turbulent heat transport in tokamak plasmas [1–4]. There are two common approaches or computational models. First, global simulations, which model the whole tokamak cross section or a large annular region, are able to simulate plasmas with a minor radius as large a $200\rho_i$ [3–5] on current generation massively parallel supercomputers. Second, flux-tube simulations are able to simulate arbitrarily large tokamak plasmas with much higher resolution but with assumptions of locality of the turbulence [2, 6–10]. Here, we will discuss the numerical implementation, parallelization, and performance of a flux-tube simulation. This type of model greatly reduces the volume of the simulation domain, and hence, the computational requirements. It does, however, require imposing more assumptions, which will be discussed in Section II. The simulation uses a particle-in-cell (PIC) method where the particles represent full gyrokinetic ion physics using the δf method [11–13]. The electrons are treated as adiabatic, i.e., Boltzmann with $e\phi/T_e \ll 1$. This adiabatic electron model is commonly used for studying ion-temperature-gradient-driven (ITG) turbulence, which is the physics problem that will be discussed throughout this paper. The flux-tube-reduced domain, using field-line-following (FLF) coordinates, was developed by Cowley, Beer, Waltz, Hammett and co-workers in gyrofluid simulations [7, 9, 14]. A similar approach, called "quasi-ballooning coordinates," has been utilized by Dimits and co-workers for gyrokinetic turbulence simulations [2, 6]. Quasi-ballooning coordinates are not exactly field-aligned which allows the grid to connect in the (almost) field line direction. Beer's work also discussed issues specific to FLF PIC simulation without an actual implementation [9]. We follow Beer's FLF approach in this paper and address further computational issues specific to δf PIC methods, including filtering along the field line, gyroaveraging, and parallel algorithms.

The flux-tube reduced domain using field-line-following (FLF) coordinates takes advantage of the field-aligned fluctuation spectra, i.e., $k_\| \ll k_\perp$. Instead of simulating the entire plasma volume, the flux-tube computational domain is a long thin tube representative of a sample of the turbulence within the core. The tube follows a set of magnetic field lines, typically spanning one poloidal circuit. The radial size of the box is taken to be small compared to the minor radius of the tokamak. We assume local values for equilibrium gradients and periodic boundary conditions in the perpendicular directions. In the parallel direction, boundary conditions are modified by the shearing of the tube associated with the shearing magnetic field lines. This is taken into account in the simulation using a shift of the particle's perpendicular position as it passes across the domain's boundaries along the field line following Beer's prescription [9]. A similar shift is also used for the parallel grid boundary conditions. The boundary conditions will be discussed in more detail in Section III.

Discrete particle noise is reduced by filtering out large k components of the grid quantities (in this case, the guiding center ion density). In the perpendicular direction this is easily performed in Fourier space as part of the fast Fourier transform (FFT) solver of Poisson's equation. However, in the direction of the magnetic field, a Fourier filter is not possible because of the non-periodicity. An approximate Fourier filter could be applied by mapping a few poloidal passes of the field line (a few $\pi q R$) then using a parallel FFT, but would involve a relatively large amount of interprocessor communication. Instead a straightforward multi-step, multi-weight digital filter is utilized. This scheme is very compatible with the one-dimensional domain decomposition used here.

Domain decomposition in the context of conventional PIC simulation has been shown to be scalable [15–18]. Here we discuss the implementation of a domain decomposition using non-orthogonal curvilinear field-line-following coordinates. The domain decomposition involves parceling out equal sections of the domain and their respective particles to each processing element (PE). Each PE works only with its own subdomain. As particles leave the subdomain, their information must be passed to the appropriate PE that controls the subdomain in which the particle now resides. This is achieved by calls to a parallel particle sorting algorithm first developed by Decyk as part of the "PLib" parallel PIC simulation library [15], and further developed by Tran. The gyrokinetic field solve involves only local operations in the direction along the field line. This makes a one-dimensional domain decomposition in this direction natural. Because the domain size is fixed, a domain decomposition must pack the subdomains more and more tightly as the number of PEs increases. This reduces the distance that a particle can travel before it passes to the next subdomain and must be communicated to another PE. As a result, particle sort and communications times become a larger fraction of the computing time as the number of PEs is increased.

To circumvent this packing problem, a new parallelization scheme that supplements the domain decomposition, dubbed domain cloning, is introduced. Instead of packing more and more subdomains together, domain cloning clones copies of the grid and loads each clone with its own set of particles. At all time steps, the clones are updated so that each clone has identical grid information and each clone pushes its own set of particles. This reduced density of PEs exhibits reduced particle sorting time and outperforms a simple 1D domain decomposition.

The paper is organized as follows. In Section II, a brief description of the physical model is given. Section III discusses the numerical algorithms. In Section IV we discuss the domain decomposition and in Section V the domain cloning algorithm. Timing results and parallel performance issues are also discussed in Sections IV and V. Section VI shows representative simulation results and particle convergence properties.

2. PHYSICAL MODEL

The simulation solves gyrokinetic equations for the ions [19–21]. We take the electrostatic limit and assume adiabatic electrons. In addition, the $E_\parallel$ nonlinearity is neglected as in Refs. [11, 19] due to small $k_\parallel/k_\perp$ ordering arguments. Tests with and without this term indicate that it has very little effect on the nonlinear saturation level or heat flux. In general, gyrokinetics is a coordinate transformation of the Vlasov–Poisson system from particle coordinates $(\vec{\mathbf{x}}, \vec{\mathbf{v}})$ to guiding center coordinates $(\vec{\mathbf{R}}, \mu, \mathbf{v}_\parallel, \phi)$, where $\vec{\mathbf{R}}$ is the position of the guiding center of the particle, $\mu = v_\perp^2/2B$, ϕ is the gyrophase angle, and $\mathbf{v}_\parallel$ and $v_\perp$ are velocity components parallel and perpendicular to the magnetic field. These new coordinates follow the gyrocenter of the particle orbit. The transform in coordinate space is straightforward, $\vec{\mathbf{R}} = \vec{\mathbf{x}} - \vec{\rho}$, where $\vec{\rho}$ is the gyroradius. We employ the ordering of $\omega/\Omega \sim \rho/L \sim k_\parallel/k_\perp \sim \epsilon$ where ω is a typical frequency of interest, $\Omega = eB/mc$ is the gyrofrequency, L is a typical length scale, and $k_\parallel$ and $k_\perp$ are wave numbers parallel and perpendicular to the magnetic field. With this, the transformed equation is averaged over the gyrophase angle of the particles, eliminating the fast time scale of the gyromotion. By taking the magnetic moment, μ, as invariant ($\dot{\mu} = 0$), the phase space variables are reduced by one dimension, with μ reduced to a particle parameter (e.g., charge or mass). The gyrokinetic Vlasov equation can be written as

$$\partial_t f = \dot{\mathbf{z}} \cdot \nabla_{\mathbf{z}} f = 0, \tag{1}$$

where f is the gyrophase independent distribution function in phase space and $\mathbf{z}$ is a gyrophase independent phase space variable in guiding center coordinates, $\mathbf{z} = (\vec{R}, v_\parallel, \mu)$ [22]. The guiding center equations of motion are given in Ref. [22] and will be discussed below. The quasi-neutrality condition is [20]

$$-\left(\frac{\rho_s}{\rho_i}\right)^2 \lambda_D^{-2}[1 - \Gamma_0(b)]\phi = -4\pi e(\delta\bar{n}_i - \delta n_e), \tag{2}$$

where $\delta n = (n - n_0)/n_0$ is the perturbed part of the density, $\delta\bar{n}_i$ is the perturbed gyrophase averaged ion density, $b = (k_\perp \rho_i)^2$, $\rho_s^2 \equiv (T_e/T_s)\rho_i^2$, $\lambda_D \equiv \sqrt{T_e/(4\pi n_0 e^2)}$ is the Debye length, and Γ_0 is the gyroaveraged Bessel function J_0. The appearance of the Γ_0 accounts for finite gyroradius effects. For small b, the gyrokinetic Poisson equation reduces to $(\rho_s/\lambda_D)^2\nabla_\perp^2\phi = -4\pi e(\delta\bar{n}_i - \delta n_e)$. Note that the $1-\Gamma_0$ depends only on $k_\perp\rho_i$ which is crucial in the implementation of the field solve and choice of domain decomposition along the direction of the B-field.

To properly treat the adiabatic electron response, we must take the perturbed electron density to be of the form

$$\delta n_e = e(\phi - \langle\phi\rangle)/T_e, \tag{3}$$

where T_e is the electron temperature and $\langle\phi\rangle$ denotes a flux surface average of the electrostatic potential [9, 23, 24]. The proper adiabatic electron response has a strong effect on the nonlinear evolution. Without this proper treatment, nonlinear $\mathbf{E}\times\mathbf{B}$ shear flows are suppressed resulting in much higher heat fluxes [23, 25]. The flux-surface-average of the electrostatic potential requires a global sum in the direction along B, which is an important consideration for parallelization.

The simulation domain used is a flux tube [9]. This is a long narrow "tube" that follows a set of magnetic field lines in the tokamak. The assumption of the reduced flux tube domain relies on the fact that the turbulence is observed to be typically elongated along the magnetic field line and has short perpendicular variations, i.e., $k_\perp \gg k_\parallel$, where k is a typical wave number. The flux tube used is the minimal volume necessary to resolve the micro-instabilities associated with turbulent transport. The formulation assumes L_x and L_y, the two perpendicular dimensions of the flux tube, to be much smaller than the minor radius of the tokamak. We take local values of gradient and equilibrium quantities, e.g., $q' = q'|_{r=r_0}$, $T' = T'|_{r=r_0}$, $R = R|_{r=r_0}$, etc. Dimensions along the B-field are determined by external parameters such as q, the safety factor, and R, the major radius. Typically, the simulation domain will span one poloidal circuit, i.e., $\theta \in [-\pi, \pi]$.

Field-line following coordinates are used in this simulation. This coordinate system follows the twisting and sheared magnetic field lines of a tokamak and maps them to rectilinear coordinates [9, 14]

$$x = r - r_0, \qquad y = \frac{r_0}{q_0}(q(r)\theta - \psi_0), \qquad z = q_0 R_0 \theta, \tag{4}$$

where r is the minor radius, r_0 is the location of the center of the flux tube, θ is the poloidal angle, ψ_0 is the toroidal angle, $q(r)$ is the safety factor as a function of the minor radius, $q_0 = q(r_0)$, and R_0 is the major radius to the center of the tokamak. It must be noted that these are not simple Cartesian rectilinear coordinates, in particular, the Jacobian is not equal to 1. Assuming small r_0/R and L_x, the FLF coordinates simplify the problem by providing simple differential operators

$$\hat{\mathbf{b}}\cdot\nabla f = \frac{\partial f}{\partial z}, \tag{5}$$

$$(\hat{\mathbf{b}}\times\nabla\phi)\cdot\nabla f = \frac{\partial\phi}{\partial y}\frac{\partial f}{\partial x} - \frac{\partial\phi}{\partial x}\frac{\partial f}{\partial y}, \tag{6}$$

$$\nabla_\perp^2\phi = \frac{\partial^2\phi}{\partial x^2} + (1+s^2z^2)\frac{\partial^2\phi}{\partial y^2} + 2sz\frac{\partial^2\phi}{\partial x\partial y}, \tag{7}$$

where $s = 1/L_s = r_0 q'/Rq^2$ is the shear scale length assumed to be constant throughout the box. As with the other gradient quantities, s assumes the local value at r_0.

The δf method [11–13] assumes a time independent equilibrium phase space distribution with a small perturbation that is time dependent having the form

$$f(\mathbf{z}, t) = f_0(\mathbf{z}) + \delta f(\mathbf{z}, t). \tag{8}$$

This distribution function is inserted into the gyrokinetic Vlasov equation with the result

$$\partial_t \delta f + \dot{\mathbf{z}} \cdot \nabla_{\mathbf{z}} \delta f = -\dot{\mathbf{z}}^1 \cdot \nabla_{\mathbf{z}} f_0, \tag{9}$$

where we have used the fact $\dot{\mathbf{z}}^0 \cdot \partial_{\mathbf{z}} f_0(\mathbf{z}) = 0$ and $\dot{\mathbf{z}} = \dot{\mathbf{z}}^0 + \dot{\mathbf{z}}^1$. In the simulation we use a Maxwellian equilibrium distribution in velocity and take the spatial distribution to be uniform. The δf method reduces the noise associated with the discrete particle representation of f. Here, we keep only the $\mathbf{E} \times \mathbf{B}$ nonlinearity as in Ref. [11].

We evolve Eq. (9) by defining a weight as [12]

$$w_i = \left.\frac{\delta f}{f}\right|_{\mathbf{z}=\mathbf{z}_i, t}, \tag{10}$$

where

$$\delta f = \sum_i w_i \delta(\mathbf{z} - \mathbf{z}_i), \tag{11}$$

and finite size particles are used to approximate the δ-function in the spatial dimensions as in Ref. [12]. This eliminates the need to track volume elements associated with each discrete particle [13]. The weights are evolved in time along with the particle's phase space variables according to

$$\dot{w}_i = -(1 - w_i) \left[\dot{\mathbf{z}}^1 \frac{\nabla_{\mathbf{z}} f_0}{f_0} \right]_{\mathbf{z}=\mathbf{z}_i, t}, \tag{12}$$

which is then deposited onto the grid to solve for the fields, similar to the deposision of a unit of charge in conventional PIC simulations.

We follow the characteristics of Eq. (9). In field line following coordinates, the equations of motion are [26]

$$\dot{x} = -\frac{1}{R_0}\left(\mu + \frac{\rho_\parallel^2}{B}\right)\sin\theta - \frac{\partial\phi}{\partial y}, \tag{13}$$

$$\dot{y} = -\frac{1}{R_0}\left(\mu + \frac{\rho_\parallel^2}{B}\right)\left(\cos\theta + \frac{r_0 q'\theta}{q}\sin\theta\right) + \frac{\partial\phi}{\partial x}, \tag{14}$$

$$\dot{z} = \rho_\parallel B^2, \tag{15}$$

$$\dot{\rho}_\parallel = -\left(\mu + \frac{\rho_\parallel^2}{B}\right)\frac{r_0}{q_0 R_0^2}\sin\theta, \tag{16}$$

$$\dot{\mu} = 0, \tag{17}$$

and the evolution equation for the particle weight is

$$\dot{w} = (\kappa E_y + \dot{\vec{x}} \cdot \vec{E})(1 - w), \tag{18}$$

to lowest nontrivial order in r_0/R, where $B = B_0(1 - (r_0/R_0)\cos\theta)$, $\rho_\parallel = v_\parallel/B$, and $\kappa = \kappa_n - (3/2 - (1/2)(v^2/v_{th}^2))\kappa_T$ is related to the scale lengths of the density and temperature profiles ($\kappa_n = -\frac{1}{n}\frac{\partial n}{\partial x}$), $\kappa_T = -\frac{1}{T}\frac{\partial T}{\partial x}$, and $v_{th} = (T/m)^{1/2}$ is the thermal velocity). Note that to lowest order, μ is an invariant of the motion. These equations of motion are for the guiding center and not the actual particle position and finite-gyroradius effects are taken into account by four-point gyroaveraging [27], as explained in the following section.

3. ALGORITHMS AND NUMERICAL ISSUES

The particle equations of motion and the evolution equation for the particle weight can be written in the form

$$\dot{\mathbf{Z}} = \mathbf{F}(\mathbf{Z}), \tag{19}$$

where $\mathbf{Z} = (\mathbf{R}, v_\parallel, \mu, w, t)$. The evaluation of $\mathbf{F}$ is a complicated and slow numerical operation because it involves the deposition of the particles on the grid (scatter), the field solve, and the evaluation of the field at the particle location (gather). Equation (19) is integrated using a conventional predictor-corrector scheme [28–30] known as the modified Euler method,

$$\tilde{\mathbf{Z}}^{n+1} = \mathbf{Z}^{n-1} + 2\Delta t\mathbf{F}(\mathbf{Z}^n), \tag{20}$$

$$\mathbf{Z}^{n+1} = \mathbf{Z}^n + \frac{\Delta t}{2}(\mathbf{F}(\mathbf{Z}^n) + \mathbf{F}(\tilde{\mathbf{Z}}^{n+1})). \tag{21}$$

The conventional leapfrog methods [31] cannot be used because the equations for the drift motion are first order in time. The modified Euler scheme has the property that the predictor step is time centered. It also minimizes the number of evaluations of **F** by storing the two old values of **Z**. This gives a significant advantage over multiple evaluations of **F** that would be required in higher order Runge–Kutta schemes. However, the predictor-corrector scheme does require more memory, but this is a minor drawback since memory is typically not a limitation.

For completeness, we define the (one-dimensional) local truncation error as $e^{n+1} = z^{n+1} - z(t^{n+1})$, where $z(t^{n+1})$ is the exact solution and z^{n+1} is the numerical solution predicted by Eqs. (20) and (21) using the exact values for z^n and z^{n-1}. Using Taylor series one can obtain the following local truncation error $e^{n+1} = -\frac{5}{12}z'''(t)\Delta t^3 + O(\Delta t^4)$ [30]. Hence, this scheme is second order in time. In addition, this modified Euler method has a relatively weak damping rate for a second-order scheme with $\omega\Delta t \leq 1$, $\gamma \sim O(\omega\Delta t)^3$ for harmonic oscillations where ω is the natural frequency and gamma is the numerical damping rate [32].

The particles are initialized with a Maxwellian distribution. This is done using a bit-reversed quiet start [31] in all five phase-space variables. Alternately, data can be read from a restart file of specified positions, velocities, and weights. This allows check-pointing, running the simulation beyond the runtime limit, or facilitates a nontrivial initial condition. The bit-reversed quiet start reduces discrete particle noise at early times and more uniformly fills the phase space for better resolution (or sampling).

We use a four-point gyroaveraging technique [27], which properly weights the deposition of charge and the calculation of the electric field over the gyroorbit and is accurate up to to $k_\perp\rho_i \sim 1$.

In the field-line following coordinates, care must be taken to deposit a circular ring of charge on the twisted non-orthogonal grid, as well as proper gyroaveraging of the electric field. The twist of the coordinates due to magnetic shear modifies the the four-point gyroaverage in the following way

$$x = x_{gc} + \delta_x^i \rho_i,$$
$$y = y_{gc} + \delta_y^i \rho_i + sz\delta_x^i \rho_i,$$

where (x, y) is the position of the particle at the four points. (x_{gc}, y_{gc}) is the guiding center position, ρ_i is the gyroradius of the particle, s is the shear as defined for Eq. (7), z is the position along the field line, and the (δ_x^i, δ_y^i) are the ring weight delta functions that take on the value $(\delta_x^i, \delta_y^i)_{i\in[1,2,3,4]} = [(1, 0), (-1, 0), (0, 1), (0, -1)]$ for a 4-point gyroaveraging. The extra term involving δ_x^i in y results from the r dependence of $q(r)$ in the definition of the FLF y (see Eq. (4)) and accounts for the shearing of the coordinate system. It is possible to reduce the number of ring-points to 2 or even 1, or increase it to 8 or more, but a 4-point ring is most commonly used and gives reasonable accuracy [27].

The gyrokinetic quasi-neutrality condition is solved using a two-dimensional FFT in the perpendicular direction. Assuming the size of the box in the perpendicular directions is not too large and $r_0/R \ll 1$, we take the gyroradius to be a function of z only, $\rho_i(z(\theta)) = \rho_{i0}(1 - (r_0/R)\cos(\theta))$, where ρ_{i0} is the value at $z = 0$. Additionally, $k_\perp$ is a function of z, as discussed in Ref. [14]. The parameter b in Eq. (2) then becomes

$$b(z) = k_\perp^2(z)\rho_i^2(z) = \left[1 + 2\frac{r_0}{R}\cos\left(\frac{z}{q_0 R}\right)\right]\left(k_x^2 + (1 + s^2z^2)k_y^2 + 2szk_x^2k_y^2\right). \quad (22)$$

This form of b in Eq. (2) allows for simple evaluation in Fourier space.

Filtering in the perpendicular direction is also done in Fourier space to smooth out high $k_\perp$ noise with wavelengths comparable to the grid spacing. The Fourier filter used is of the form $e^{-(k_\perp \rho_i)^a}$ where a is typically 2–4. An inverse FFT is then done and further filtering is done along the direction of the B-field in real space.

As will be discussed below, enforcing toroidal boundary conditions requires a shift at the end of the box in the z-direction. Hence, this direction is not periodic in z, and therefore, a Fourier filter is not appropriate. In addition, the domain is decomposed in the z-direction, making FFTs in this direction cumbersome. Instead, we use a two-step, three-point digital filter [31]. Both steps are essentially weighted boxcar averages of the form

$$\phi_i = \frac{W\phi_{i-1} + \phi_i + W\phi_{i+1}}{1 + 2W}, \quad (23)$$

where $W = 1/2$ in the first pass and $W = -1/6$ for the second pass. The first step is the initial smoothing. The second step corrects for over attenuation. The ion density is passed through the filter several times, typically three times. This reduces large $k_\parallel$ noise. As shown in Fig. 2, small $k_\parallel$ values ($<$0.5) are well preserved. At the resolution of the grid spacing, there is minimal attenuation of $\sim$85%, with a fair drop off for larger k. This is a marked improvement over a single weight of $W = 1/2$ where there is attenuation at all values. This method fits well with the one-dimensional domain decomposition and the non-periodic boundary conditions in z. Communication is minimized to nearest neighbors. In the actual implementation, the digital filter is first performed on the ion density, then the Fourier space filtering is applied in conjunction with the FFT Poisson solver. This is done to maintain the volume-averaged ϕ equal to zero. We found that digital filtering after the Fourier filter introduced spurious components from adjacent grid points and produced a non-zero volume-averaged ϕ.

The smoothed electrostatic potential is now locally available on each processor. The E-field is then calculated from the electrostatic potential using a centered finite-difference (CFD) without any communications because nearest grid cells in z are held in buffers. We utilize double buffering to eliminated the need for communications during the CFD. Double buffering provides a buffer cell on both ends of the subdomain. This will be explained in more detail in Section IV. With the E-fields for the subdomain calculated, particles can be pushed locally without any communications. The next necessary communication is to update the particles that have moved beyond the subdomain of the processor. Using the predicted position, the ion density, electrostatic potential, and E-field are calculated and the corrector step is performed (see Eqs. (20) and (21)).

Because of the non-orthogonality of the FLF coordinates and reduced domain of the flux tube, the proper boundary conditions are not trivial. In the radial and poloidal direction we assume periodic boundary conditions. This is based on the assumption that the correlation length of the turbulence in these directions is small compared to the size of the box in that direction [9].

The direction along the magnetic field line (z) must be treated carefully since both the linear eigenmodes and the turbulence are well-aligned with the magnetic field. One end of the box in the z-direction does not line up with the other end; see Fig. 1. In the z-direction, we enforce periodicity in the toroidal (ϕ) and poloidal (θ) directions at the ends of the box in z. We follow the prescription of Beer [9] and do a shift in the y direction following from Eq. (4)

$$\delta y = \pm\left(q(r)\frac{r_0}{q_0}2\pi\right) \qquad \mathrm{mod}\, L_y, \tag{24}$$

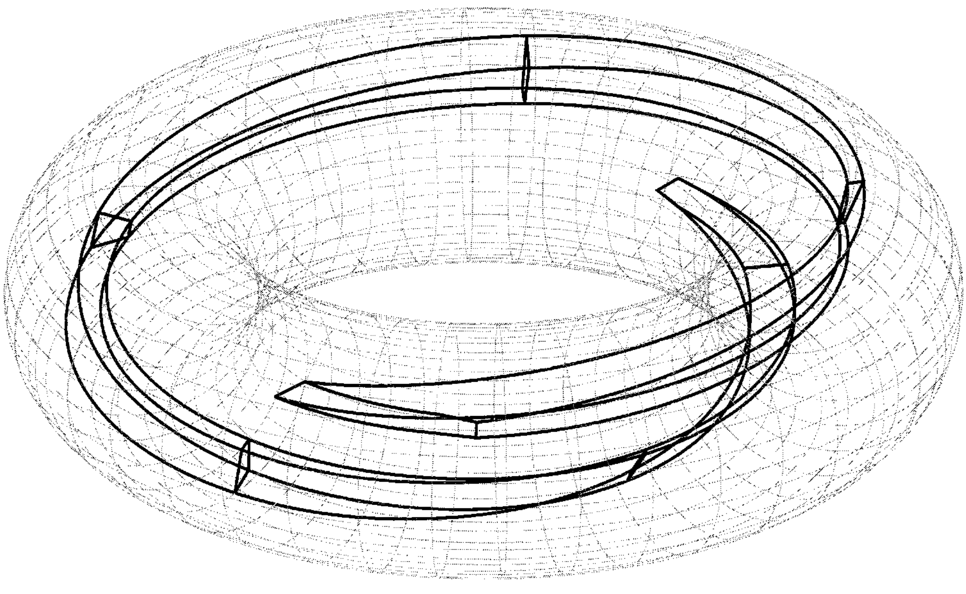

FIG. 1. Flux-tube computational domain for a present-day tokamak.

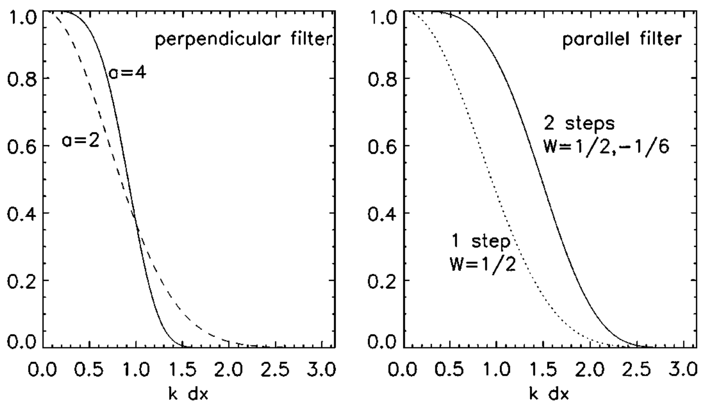

FIG. 2. Filter function plotted versus $k\Delta x$.

where the $\pm$ accounts for either end of the box and the mod L_y ensures poloidal periodicity (L_y is the size of the box in the y-direction). This must be done for both particles and field quantities. Alternatively, one could assume statistical periodicity, as we did in the perpendicular direction, but this would require that the flux-tube domain be extended to several poloidal circuits to ensure that the ends of the box are decorrelated. This would require additional gridpoints as well as particles to fill the larger domain space, and in turn, more computing resources. Hence, this shift (Eq. (24)) is a more efficient alternative.

4. PARALLEL ISSUES AND PERFORMANCE

Initially, a very simple parallelization scheme, we termed "a poor man's domain decomposition," was adopted. A copy of the grid quantities is passed to each of the PEs. The particles however were divided among the PEs. After each charge deposition, a global sum and broadcast were performed to update the density array on each of the PEs. The grid calculations are replicated on each of the PEs. It was observed that for a small number of PEs, this scaled fairly well with particle number. Unfortunately, as the number of PEs increased beyond 4, the global communication time became prohibitive. Despite its shortcomings, this initial parallelization scheme motivates the domain cloning scheme which will be discussed in Section VI.

The current implementation uses a one-dimensional domain decomposition in the z-direction. This takes the z grid spanning from $1:km$ and decomposes it into N subdomains of $1:mykm$ where $mykm = km/N$ and N is the number of PEs. Each subdomain is assigned to one of the PEs and, in effect, the subdomain becomes the simulation domain of that particular PE. The PEs perform only those grid calculations (deposition of the ion density, Poisson Solver, CFD) relevant to their subdomain. Each PE pushes only those particles that lie in its particular subdomain at that particular time step. Load balancing is not an issue because the particle density is fairly uniform along the magnetic field line. As particles pass beyond the subdomain of a particular PE, the particle and its information must be passed to the appropriate PE that is assigned to that region of space. Particle sorting is performed at the end of each particle update for both the predictor and corrector steps (Eqs. (20) and (21)).

The particle sorting routine involves an initialization routine and move routine. The initialization routine allocates an array for the particles that will be passed to other subdomains (**send**), an array to index the particles to be passed (**isend**), and an array to index the holes in the particle array (**ihole**). The move routine determines which particles do not belong in the subdomain (and simultaneously which subdomain they do belong in) and records their index in **isend** and **ihole**, then fills and sorts (by subdomain) the **send** array. The appropriate portion of the **send** array is passed to the appropriate subdomain. The particle array is then filled, first filling in the holes using **ihole** then appending to the end of the array.

In the gather/scatter operations, the CFD, and the digital filter, grid information from the neighboring subdomain is needed. This could, potentially, be a prohibitive communications cost. The grid communication is minimized by the use of buffers. A buffer is an identical copy of a neighboring grid of the adjacent subdomain. This additional memory cost of storing a buffer reduces the communications cost that would otherwise be needed. Thus, a particle that sits just beyond the last gridpoint of the subdomain (but not in the next subdomain) will have all the necessary grid information available without the need of making a communications call to neighboring PEs. Here, memory is wasted in exchange for reduced communications cost. The electrostatic potential uses a double buffer, a buffer at both ends of the subdomain. This is necessary for the digital filtering and for the CFD.

The flux tube code has been run on a number of platforms. Here we report performance on two massively parallel computers: (1) the Origin 2000 (O2K) at the Advanced Computer Laboratory, Los Alamos National Laboratory; and (2) the T3E at the National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory. The use of the Message Passing Interface (MPI) message-passing library makes the code portable.

Figure 3 shows the performance on the two massively parallel supercomputers. This scaling study was performed by increasing the domain size with the number of PEs. Note in Table I that the number of grid points in the z direction corresponds to the number of PEs. This means that each PE holds the minimum subdomain size possible. For a given domain size, three runs were performed for 2, 4, 8 particles per grid point. This gives a variety of configurations, showing performance versus problem size. Figure 3 shows the node seconds per timestep versus the number of particles. The simulation achieves performance times of fractions of μs per particle per timestep. For the larger runs, the O2K shows twice the

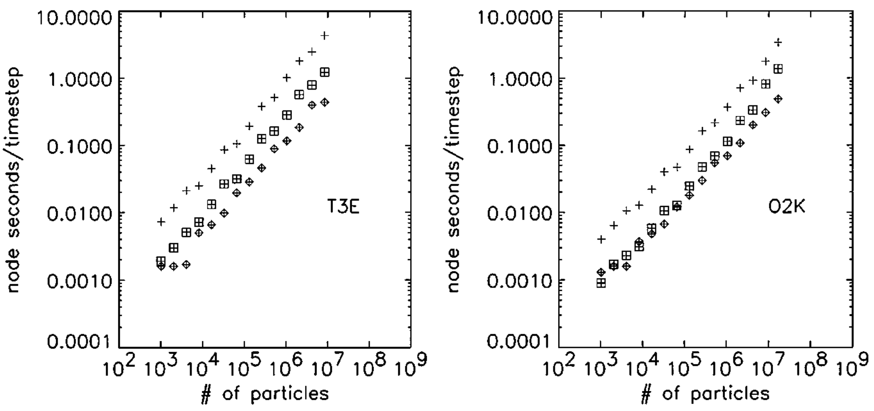

FIG. 3. Comparison of the performance on the T3E and O2K. Open symbols are total wall-clock time. Square symbols are particle sort times. Diamond symbols are Poisson solver/filter times.

TABLE I
The T3E and O2K Timings for Different Numbers of Processors and Problem Sizes

No. part.	Grid size	Part./cell	No. PE	Wall-clock T3E	Wall-clock O2K	Speed T3E	Speed O2K
2^{10}	$8 \times 8 \times 4$	4	4	0.007	0.004	27.3	15.6
2^{11}	$8 \times 8 \times 4$	8	4	0.012	0.006	23.4	11.7
2^{12}	$8 \times 8 \times 4$	16	4	0.021	0.011	20.5	10.7
2^{13}	$16 \times 16 \times 8$	4	8	0.025	0.013	24.4	12.7
2^{14}	$16 \times 16 \times 8$	8	8	0.045	0.022	22.0	10.7
2^{15}	$16 \times 16 \times 8$	16	8	0.086	0.040	21.0	9.77
2^{16}	$32 \times 32 \times 16$	4	16	0.105	0.047	25.6	11.5
2^{17}	$32 \times 32 \times 16$	8	16	0.194	0.086	23.7	10.5
2^{18}	$32 \times 32 \times 16$	16	16	0.381	0.162	23.3	9.89
2^{19}	$64 \times 64 \times 32$	4	32	0.516	0.215	31.5	13.1
2^{20}	$64 \times 64 \times 32$	8	32	1.022	0.369	31.2	11.3
2^{21}	$64 \times 64 \times 32$	16	32	1.806	0.711	27.6	10.8
2^{22}	$128 \times 128 \times 64$	4	64	2.466	0.919	37.6	14.0
2^{23}	$128 \times 128 \times 64$	8	64	4.324	1.770	33.0	13.5
2^{24}	$128 \times 128 \times 64$	16	64	9.315	3.377	35.5	12.9

Note. Wall-clock time in seconds/step, speed in μs-PE/particle-step.

performance of the T3E, despite having a slower clock speed. Figure 3 shows excellent linear scaling with problem size.

Figure 3 also shows the wall clock times of the Poisson solver and particle sort times. As mentioned above, the sort time consumes a large fraction of the total wall clock time. To some extent, domain cloning improves this in a simple way. Figure 4 shows a simple scaling study where we fix the size of the domain ($64 \times 64 \times 64$) and the number of particles (2^{21}) and vary the number of processors. Figure 4 shows a log-log plot of seconds per particle per timestep versus the number of PEs. A straight line fit is made to the timing data. By this, we obtain the scaling parameter A for $\tau_{ts} \propto N^A$ where τ_{ts} is the wall clock time per particle per timestep and N is the number of PEs. One can see that the performance of the

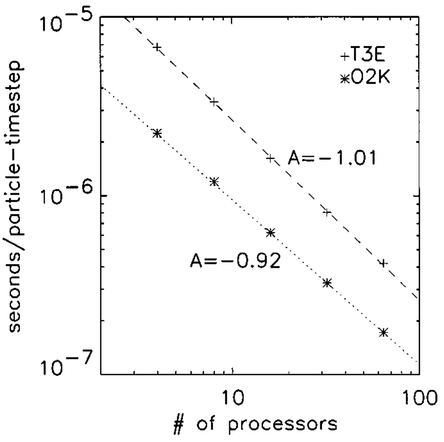

FIG. 4. Performance versus number of processors. A denotes the slope of the line.

simulation is in the fractions of microseconds per particle per timestep for both machines. Despite its slower performance, the T3E is seen to scale better than the O2K, with a fitted value of $A = -1.01$, whereas the O2K has an $A = -.92$.

5. DOMAIN CLONING

As the number of PEs increases, the subdomains become more closely packed resulting in more subdomain boundaries. This increases the time spent in the particle sorting routine as particles encounter subdomain boundaries more frequently. In addition, the higher grid resolution gained is unnecessary due to the elongated nature of the typical modes present in the simulation.

Domain cloning presents a simple solution to this over packing issue that sidesteps the complications associated with a higher order domain decomposition, e.g., parallelization of FFT. As its nomenclature suggests, the simulation domain is cloned, i.e., multiple copies of the same domain are made. The PEs are divided into groups equaling the number of clones. Each group of PEs is assigned to one of the domain clones. Each domain clone is loaded with its own set of particles and a one-dimensional domain decomposition is performed such that each PE has a corresponding subdomain clone. This is easily achieved in MPI with two calls to **MPI_COMM_SPLIT**, the first defining each cloned domain **TUBE_COMM** and the second defining the subdomain clones **GRID-COMM**. The members of **TUBE_COMM** comprise a single domain clone. The corresponding members of **GRID_COMM** hold identical copies of the grid information but different particles. In the push phase of the simulation, each clone pushes its own set of particles as described in the previous section. These particles reside only in their particular domain clone and are never communicated to other domain clones. Upon depositing the particles onto the grid, the subdomain clones perform a global sum among themselves (**GRID_COMM**) to obtain the total ion density in each subdomain (Fig. 5).

The minor additions made to the original algorithm are the allocation of two additional communicators and a global sum across clones. Communcations calls must be modified to remain local among the cloned domains (**TUBE_COMM**) by changing the communicator parameter. All other aspects of the program remain unchanged.

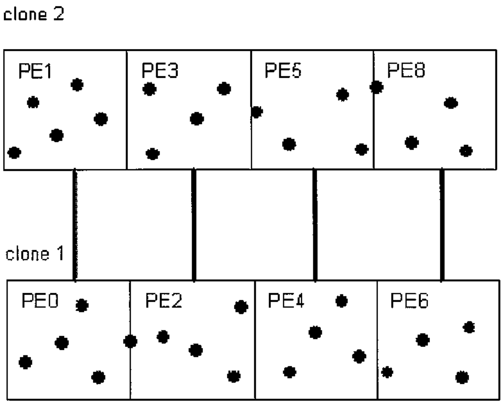

FIG. 5. Schematic of a cloned domain. Thick lines connecting processors denote subdomain clone **GRID-COMM** and black dots denote particles.

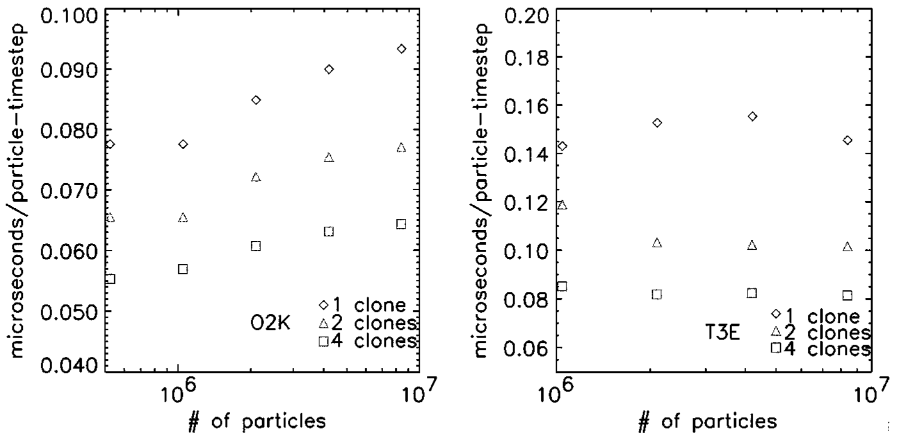

FIG. 6. Particle sort time for the O2K and T3E.

There is a redundant calculation of the electrostatic potential, filtering, and E-field, but for a small number of clones this inefficiency is small. Each domain clone now has identical grid information, but retains their own set of particles. In sorting the particles, each cloned domain has fewer subdomain boundaries and each subdomain spans a larger portion of the domain, reducing the particle sorting time. Figure 6 shows the sort time per particle per timestep for both the O2K and T3E. The timing test holds the number of PEs fixed (32 for the O2K and 64 for the T3E) as well as the perpendicular grid size (64×64). The parallel grid size is reduced for increasing numbers of clones, e.g., 32 for 1 clone on the O2K, 16 for 2 clones, etc. The timing is done for increasing numbers of particles. There is a marked improvement in sorting time with the increasing number of clones, thereby reducing total wall clock time. For example, with 64 processors and 4 clones there is an over all speedup of 50%. As shown in Fig. 3, the particle sort time can be a significant fraction of total time. For large particle runs, requiring very large numbers of PEs, the packing problem results in sort times that are on the order of 30–40% of the run time. Domain cloning provides a simple solution.

This simple scheme of domain cloning can be used alone or in conjunction with domain decomposition as another layer of parallelization. For the case of no domain decomposition, this reduces to our initial parallelization scheme discussed at the beginning of Section IV.

6. REPRESENTATIVE SIMULATION RESULTS

Finally, we briefly discuss representative simulation results and demonstrate numerical convergence with respect to particle number. Further application of this model to turbulence and transport simulation results are given in Refs. [33–36]. This simulation has been used extensively to study ion heat transport. Figure 7 shows a scan varying the temperature gradient scale length, L_T, which is the linear drive of the ion-temperature gradient instability. Plotted is the ion heat diffusivity (χ_i) and linear growth rate (γ). The linear growth rate was measured from the linear growth phase of $|\phi|$. Figure 7 shows an important parameterization of how the nonlinear heat flux scales with the drive of the linear instability and was used to study the physics basis for various reduced transport models in the “Cyclone Project” [36]. The physical parameters are $R/L_T = 6.9$, $R/L_n = 2.2$, $T_e/T_i = 1$, $r/R = 0.18$, $q = 1.4$, and

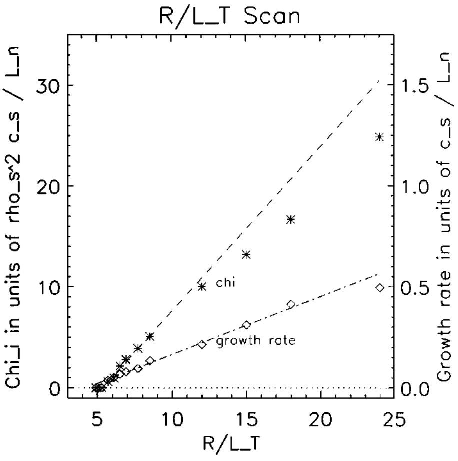

FIG. 7. Ion heat diffusivity and linear growth rate versus R/L_T, the primary linear drive of the linear instability.

$\hat{s} \equiv \frac{r}{q}\frac{dq}{dr} = 0.78$. These are the “Cyclone Team DIII-D base case” parameters which are representative of a typical H-mode plasma (DIII-D Shot 81499), but assumes a circular zero-beta large aspect ratio MHD equilibrium, no impurities, adiabatic electrons, and no fast ions; see Refs. [33, 36] and references within, for further details. The numerical parameters are 8 particles per cell, $64 \times 64 \times 32$ grid, with $\Delta x = \Delta y = \rho_s$, and a timestep of $\Delta t c_s / L_T = 0.0345$. The code used here has been benchmarked with the other gyrokinetic flux-tube code of Dimits and co-workers [35, 36].

Figures 8 and 9 show convergence with respect to particle number for both linear interpolation and nearest grid point interpolation schemes. Except for particle number, all parameters are the same as given above and are well converged with respect to timestep and grid size. The number of particles ranges from 1 particle per cell to 16 particles per cell for linear interpolation and 1 particle per cell to 32 particles per cell for nearest grid point interpolation. Figure 8 shows the ion heat diffusivity χ_i (proportional to the ion heat flux since the gradient is held fixed). Even 1 particle per cell is enough for convergence of the steady-state average χ_i in either case. However, the electrostatic field energy does not converge until there are at least 4 particles per cell. Careful observation of Fig. 8 indicates

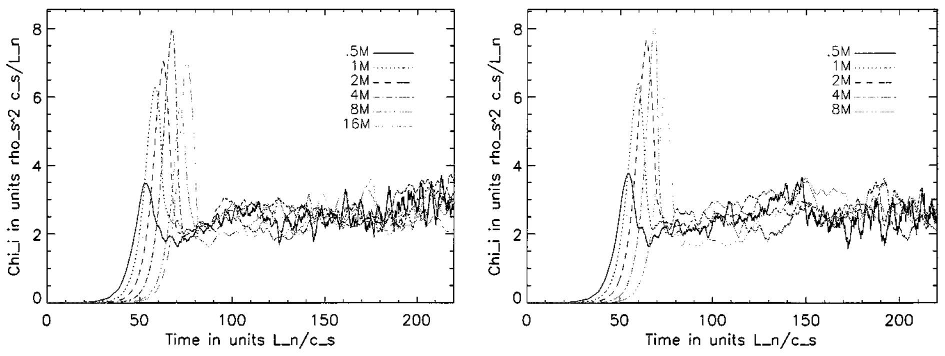

FIG. 8. Particle number convergence of the ion heat diffusivity using both nearest grid point and linear interpolation.

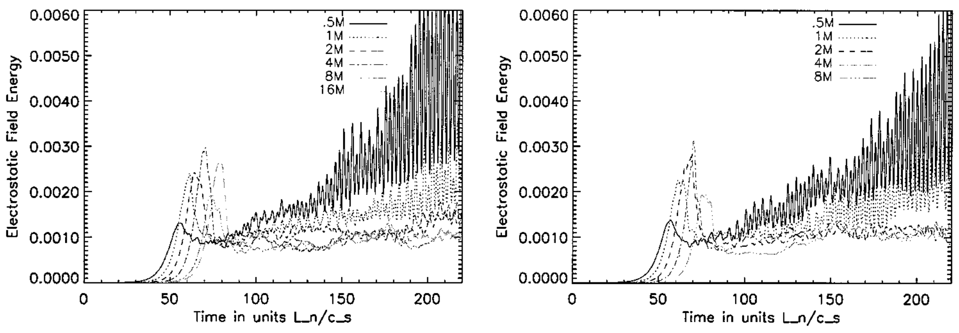

FIG. 9. Particle number convergence for the electrostatic field energy using both nearest grid point and linear interpolation.

that there may be late time growth for the nearest grid point method even for 8 particles per cell. The large oscillations in the field energy in Fig. 9 (as the particle number is decreased) are due to the growth of the rms value of the weights and excitation of geodesic-acoustic oscillations [36, 37]. We use either nearest-grid-point or linear interpolation for particle deposition and calculation of the particle's electric field. Linear interpolation is obviously more accurate and smoother; however, it requires 44 more operations for every gather and scatter operation during the gyroaverage. This results in the nearest grid-point being approximately twice as fast as linear interpolation.

7. SUMMARY

A description of the computational issues relating to a massively parallel three-dimensional toroidal nonlinear gyrokinetic flux-tube simulation has been discussed. This model allows for high resolution simulations of ion-temperature-gradient-driven turbulence in large tokamak plasmas assuming locality of the turbulence. We have presented techniques for filtering, gyroaveraging, solving the quasi-neutrality condition (field solve), and toroidal boundary conditions. The field-line coordinates and associated flux-tube computational domain are naturally suited for a one-dimensional decomposition in the direction along the magnetic field line. The field solve is local in this direction allowing for spectral solution in the other two perpendicular directions. Timing and scaling results were presented on two massively parallel supercomputers, the T3E, and O2K. The O2K has approximately twice the per processor performance of the T3E, and the simulation shows near perfect parallel scalability in both cases.

We have also presented a new parallelization scheme, domain cloning, which is another layer of parallelism and a simple alternative to two-dimensional domain decomposition. Domain cloning is a supplement to domain decomposition that circumvents the problem associated with the over packing of PEs for a given physical domain size. It was shown to improve the performance of the particle sorter and reduce the wall clock time.

ACKNOWLEDGMENTS

We thank Dr. Yang Chen for many useful discussions and help with δf formulation of the equations of motion. We also thank Drs. V. Decyk and M. Tran for their help with parallelization issues and use of their particle sorting routines. We also thank Drs. M. Beer and A. Dimits for useful discussion on flux-tube simulations. This research is supported by U.S. Department of Energy (DOE) Grant DE-FG03-97ER5442(4/6). This work is an active part of the Numerical Tokamak Turbulence Project DOE Grand Challenge. Computing resources where generously provided by the Advanced Computing Laboratory at Los Alamos National Laboratory and the National Energy Research Scientific Computing Center at Lawrence Berkeley National Laboratory.

REFERENCES

1. S. Parker, W. Lee, and R. Santoro, *Phys. Rev. Lett.* **71**, 2042 (1993).
2. A. Dimits, T. Williams, J. Byers, and B. Cohen, *Phys. Rev. Lett.* **77**, 71 (1996).
3. R. Sydora, V. Decyk, and J. Dawson, *Plasma Phys. Controlled Fusion A* **38**, 281 (1996).
4. Z. Lin, T. Hahm, W. Lee, W. Tang, and R. White, *Science* **281**, 1835 (1998).
5. S. Parker, H. Mynick, M. Artun, V. Decyk, J. Kepner, W. Lee, and W. Tang, *Phys. Plasmas* **3**, 1461 (1996).
6. A. Dimits, *Phys. Rev. E* **48**, 4070 (1993).
7. R. Waltz, G. Kerbel, and J. Milovich, *Phys. Plasmas* **1**, 2229 (1994).
8. S. Parker, W. Dorland, R. Santoro, M. Beer, Q. Liu, W. Lee, and G. Hammett, *Phys. Plasmas* **1**, 1461 (1994).
9. M. Beer, S. Cowley, and G. Hammett, *Phys. Plasmas* **2**, 2686 (1995).
10. S. Parker, M. Artun, V. Decyk, J. Kepner, W. Lee, H. Mynick, and W. Tang, in *Advanced Series in Nonlinear Dynamics*, edited by S. Benkadda, F. Doveil, and Y. Elskens (World Scientific, Singapore, 1996), Vol. 9.
11. A. Dimits and W. Lee, *J. Comput. Phys.* **107**, 309 (1993).
12. S. Parker and W. Lee, *Phys. Fluids B* **5**, 77 (1993).
13. G. Hu and J. Krommes, *Phys. Plasmas* **1**, 863 (1994).
14. S. Cowley, R. Kulsrud, and R. Sudan, *Phys. Fluids B* **13**, 2767 (1991).
15. V. Decyk, *Comput. Phys. Comm.* **87**, 87 (1995).
16. T. Williams, in *Proceedings, High Performance Computing: Grand Challenges for Compiter Simulation, Arlington, VA, March 29, 1995*.
17. A. Mankofsky *et al.*, *Comput. Phys. Comm.* **48**, 155 (1988).
18. J. Kepner, S. Parker, and V. Decyk, *SIAM News* **30**, 1 (1997).
19. E. Frieman and L. Chen, *Phys. Fluids* **25**, 502 (1982).
20. W. Lee, *Phys. Fluids* **26**, 556 (1983).
21. D. Dubin, J. Krommes, C. Oberman, and W. Lee, *Phys. Fluids* **26**, 3524 (1983).
22. T. S. Hahm, *Phys. Fluids* **31**, 2670 (1988).
23. W. Dorland, Ph.D. thesis, Princeton University, 1993.
24. G. Hammett, M. Beer, W. Dorland, S. Cowley, and S. Smith, *Plasma Phys. Controlled Fusion* **35**, 973 (1993).
25. M. Beer, Ph.D. thesis, Princeton University, 1996.
26. R. B. White and M. S. Chance, *Phys. Fluids* **27**, 2454 (1984).
27. W. W. Lee, *J. Comput. Phys.* **72**, 243 (1987).
28. C. Z. Cheng and H. Okuda, *J. Comput. Phys.* **25**, 133 (1977).
29. W. W. Lee and H. Okuda, *J. Comput. Phys.* **26**, 139 (1978).
30. R. W. Hamming, *Introduction to Applied Numerical Analysis* (McGraw–Hill, New York, 1971).
31. C. Birdsall and A. Langdon, *Plasma Physics via Computer Simulation* (McGraw–Hill, New York, 1985).
32. R. A. Santoro, Ph.D. thesis, Princeton University, 1994.
33. S. Parker, C. Kim, and Y. Chen, *Phys. Plasmas* **6**, 1709 (1999).
34. S. Parker, Y. Chen, and C. Kim, *Comput. Phys. Com.* **127**, 59 (2000).
35. A. Dimits, B. Cohen, N. Mattor, W. Nevins, D. Shumaker, S. Parker, and C. Kim, *Nuclear Fusion* **40**, 661 (2000).
36. A. Dimits *et al.*, *Phys. Plasmas* **7**, 969 (2000).
37. N. Winsor, J. Johnson, and J. Dawson, *Phys. Fluids* **11**, 2448 (1968).