SpectralPlasmaSolver: a Spectral Code for Multiscale Simulations of Collisionless, Magnetized Plasmas

Juris Vencels\textsuperscript{1}, Gian Luca Delzanno\textsuperscript{1}, Gianmarco Manzini\textsuperscript{1}, Stefano Markidis\textsuperscript{2}, Ivy Bo Peng\textsuperscript{2}, Vadim Roytershteyn\textsuperscript{3}

\textsuperscript{1}Los Alamos National Laboratory, Los Alamos, NM, USA
\textsuperscript{2}KTH Royal Institute of Technology, Stockholm, Sweden
\textsuperscript{3}Space Science Institute, Boulder, CO, USA

E-mail: \{vencels,delzanno,manzini\}@lanl.gov, \{markidis, bopeng\}@kth.se, vroytershteyn@spacescience.org

Abstract. We present the design and implementation of a spectral code, called SpectralPlasmaSolver (SPS), for the solution of the multi-dimensional Vlasov-Maxwell equations. The method is based on a Hermite-Fourier decomposition of the particle distribution function. The code is written in Fortran and uses the PETSc library for solving the non-linear equations and preconditioning and the FFTW library for the convolutions. SPS is parallelized for shared-memory machines using OpenMP. As a verification example, we discuss simulations of the two-dimensional Orszag-Tang vortex problem and successfully compare them against a fully kinetic Particle-In-Cell simulation. An assessment of the performance of the code is presented, showing a significant improvement in the code running-time achieved by preconditioning, while strong scaling tests show a factor of 10 speed-up using 16 threads.

1. Introduction

Collisionless, magnetized plasmas are ubiquitous in a variety of laboratory, space and astrophysical environments. They are described by kinetic theory and are characterized by high-dimensionality, since the particle distribution function is defined in the six-dimensional phase space obtained by the three spatial and three velocity coordinates, and strong anisotropy, since transport properties can be very different along and across the magnetic field \cite{1}. In addition, collisionless plasmas are characterized by extreme separation of spatial and temporal scales. This occurs already at the kinetic level due to the large mass difference between electrons and ions. However, the scale separation widens even further if one compares the kinetic scales with typical macroscopic scales of the system: several orders of magnitude in spatial and temporal scale separation are common for instance in magnetic fusion energy experiments or in the Earth’s magnetosphere. Even with the most advanced supercomputers, fully-kinetic simulations of these systems are unfeasible today and in the foreseeable future.

On the other hand, fluid models treat the plasma as a macroscopic fluid and are therefore suitable for system-scale simulations. They are obtained from kinetic theory by taking moments of the distribution function. A kinetic closure is necessary to close the moment equations but it can be rigorously justified only in very restrictive limits \cite{2}, which are often not met in practice. Indeed, performing system-scale simulations of collisionless (or weakly collisional) plasmas that properly include the kinetic/microscopic physics is a major challenge of computational plasma
physics. One of the recent approaches to this problem is to embed a kinetic solver in a small part of the computational domain of a fluid simulation \[3, 4, 5\].

The most common approaches for the numerical solution of the kinetic equations handle the discretization of phase space by either introducing macro-particles (as in the Particle-In-Cell (PIC) approach \[6, 7\]) or a computational grid (Eulerian-Vlasov \[8, 9\]) and therefore treat the full distribution function everywhere in the computational domain. Adaptive mesh refinement techniques can be used effectively to reduce the number of degrees of freedom for these kinetic solvers \[10, 11, 12\]. A third approach is spectral, namely the velocity part of the distribution function is expanded in basis functions (typically Hermite or Fourier) \[13\]. In certain cases, such as for a proper Hermite basis \[14, 15\] or the Legendre basis \[16\], it can be shown that the low-order moments of the expansion correspond to the fluid moments and the kinetic physics is recovered by simply adding more moments to the expansion. Thus, for a suitable spectral basis the fluid-kinetic coupling is built-in.

In recent years there has been a renewed interest on spectral methods for the kinetic equations \[15, 17, 18, 19, 20, 21\]. A comparison between the one-dimensional, electrostatic, Hermite spectral method and PIC on standard kinetic theory problems such as Landau damping, two-stream instabilities and ion acoustic waves was presented in Ref. \[19\], where it was concluded that, at least on these problems, the spectral method was orders of magnitude more accurate or faster than PIC. Vencels et al. \[15\] used the same method and performed kinetic simulations where the number of Hermite modes could be changed dynamically at run-time, thus providing the transition between fluid and kinetic regimes. An extension of the spectral method to the multi-dimensional, fully electromagnetic case is presented in \[20\], where exact conservation laws in the discrete are derived and some preconditioning techniques are presented to speed-up the convergence of the Newton-Krylov solver adopted for the nonlinear equations. These previous works also point out that the optimization of the Hermite basis (in order to reduce the number of modes and improve the computation time and robustness of the method), parallelization and efficient preconditioning techniques are critical needs for these methods.

In this paper we present the Hermite-based spectral solver, called SpectralPlasmaSolver (SPS), for the multi-dimensional Vlasov-Maxwell equations and discuss the design and implementation of the code. SPS is written in Fortran and parallelized with OpenMP for shared-memory systems. It uses open-source libraries such as PETSc \[22\] for nonlinear solvers and preconditioning and FFTW \[27\] for the convolutions.

The paper is organized as follows. Section 2 presents the governing equations and the Hermite-Fourier method. Section 3 discusses the design and implementation of the SPS code. Section 4 focuses on the application of the method to a two-dimensional Orszag-Tang vortex problem, often used as a paradigm to study plasma turbulence. Scaling tests on a shared-memory system are presented in Section 5, while conclusions are drawn in Section 6.

2. The Hermite-Fourier Spectral Method

We model a collisionless, magnetized plasma by solving the Vlasov-Maxwell equations. We consider a Cartesian system of coordinates with the spatial domain defined by \([0, L_x] \times [0, L_y] \times [0, L_z] \) (\(L_x, L_y, L_z\) are the domain lengths in each spatial direction) and assume periodic boundary conditions. We use the following normalization: time is normalized to the electron plasma frequency \(\omega_{pe} = \sqrt{e^2 n_0 / \varepsilon_0 m_e} \) (where \(e\) is the elementary charge, \(m_e\) is the electron mass, \(\varepsilon_0\) is vacuum permittivity, and \(n_0\) is a reference density), velocity to the speed of light \(c\), the spatial coordinate to the electron inertial length \(d_e = c / \omega_{pe}\), the magnetic field to a reference magnetic field \(B_0\), and the electric field to \(cB_0\). The distribution function \(f\) is normalized as \(fc^3/n_0\).
Accordingly, the dimensionless Vlasov equation reads:
\[ \frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla_x f_s + \frac{q_s}{e \omega_{pc}} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_v f_s = 0, \tag{1} \]
where \( s \) labels the plasma species \( (s = e, i \text{ for electrons and ions}) \), the cyclotron frequency is \( \omega_{cs} = q_s B_0/m_s \) and \( \mathbf{E} (\mathbf{B}) \) is the electric (magnetic) field. Equation (1) is coupled to Maxwell’s equations via the electromagnetic field:
\[ \frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}, \quad \frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{B} - \frac{\omega_{pe}}{\omega_{ce}} \left( \int_{-\infty}^{+\infty} f_i dv - \int_{-\infty}^{+\infty} f_e dv \right), \tag{2} \]
\[ \nabla \cdot \mathbf{E} = \frac{\omega_{pe}}{\omega_{ce}} \left( \int_{-\infty}^{+\infty} f_i dv - \int_{-\infty}^{+\infty} f_e dv \right), \quad \nabla \cdot \mathbf{B} = 0. \tag{3} \]

We expand the particle distribution function in Hermite basis functions,
\[ f_s = \sum_{n=0}^{N_n-1} \sum_{m=0}^{N_m-1} \sum_{p=0}^{N_p-1} C_{n,m,p}^{s}(x,t) \Psi_n(\xi_x^s) \Psi_m(\xi_y^s) \Psi_p(\xi_z^s) \tag{4} \]
(where \( N_n, N_m \) and \( N_p \) are the total number of Hermite modes in the \( v_x, v_y \) and \( v_z \) directions, respectively), to obtain a set of non-linear partial differential equations (PDEs) for the coefficients of the expansion. We consider the asymmetrically-weighted (AW) basis function, \( \Psi_n(x) = (\pi 2^n n!)^{-1/2} H_n(x) e^{-x^2} \), where \( H_n \) is the Hermite polynomial of the \( n \)-th order [23]. The argument of the basis functions is defined as \( \xi_x^s = (v_x - u_x^s) / \alpha_y^s \), where \( u_x^s \) and \( \alpha_x^s \) are (constant) shift and scaling parameters that have to be supplied by the user and \( \beta = x, y, z \). We further decompose the spatial part of the particle distribution function into Fourier basis functions,
\[ C_{n,m,p}^{s}(x,t) = \sum_{k_x=-N_x/2}^{N_x/2} \sum_{k_y=-N_y/2}^{N_y/2} \sum_{k_z=-N_z/2}^{N_z/2} C_{n,m,p}^{k_x,k_y,k_z,s}(t) \exp \left[ 2\pi i \left( \frac{k_x x}{L_x} + \frac{k_y y}{L_y} + \frac{k_z z}{L_z} \right) \right], \tag{5} \]
where \( N_x + 1, N_y, N_z \) is the total number of Fourier modes in each spatial direction.

Using the orthogonality properties of the Hermite and Fourier basis functions, the Vlasov-Maxwell equations can be rewritten as a set of ordinary differential equations (ODEs) [20]:
\[ \frac{dC_{n,m,p}^{k_x,k_y,k_z,s}}{dt} = -\frac{2\pi i k_x}{L_x} \alpha_x^s \left( \sqrt{\frac{m+1}{2}} C_{n,m+1,p}^{k_x,k_y,k_z,s} + \sqrt{\frac{m}{2}} C_{n,m-1,p}^{k_x,k_y,k_z,s} + \frac{u_y^s}{\alpha_y^s} C_{n,m,p}^{k_x,k_y,k_z,s} \right) \]
\[ + \frac{2\pi i k_y}{L_y} \alpha_y^s \left( \sqrt{\frac{m}{2}} C_{n,m+1,p}^{k_x,k_y,k_z,s} + \sqrt{\frac{m+1}{2}} C_{n,m-1,p}^{k_x,k_y,k_z,s} + \frac{u_y^s}{\alpha_y^s} C_{n,m,p}^{k_x,k_y,k_z,s} \right) \]
\[ + \frac{2\pi i k_z}{L_z} \alpha_z^s \left( \sqrt{\frac{p}{2}} C_{n,m,p+1}^{k_x,k_y,k_z,s} + \frac{p}{2} C_{n,m,p-1}^{k_x,k_y,k_z,s} + \frac{u_z^s}{\alpha_z^s} C_{n,m,p}^{k_x,k_y,k_z,s} \right) \]
\[ - \frac{q_s}{e \omega_{pc}} \sqrt{\frac{2m}{\alpha_x^s}} [E_x * C_{n,m-1,p}]_{k_x,k_y,k_z} + \frac{\sqrt{2m}}{\alpha_y^s} [E_y * C_{n,m-1,p}]_{k_x,k_y,k_z} + \frac{\sqrt{2p}}{\alpha_z^s} [E_z * C_{n,m,p-1}]_{k_x,k_y,k_z} \]
\[ - \sqrt{(m+1)p} \left( \alpha_y^s \frac{u_y^s}{\alpha_y^s} - \alpha_x^s \right) C_{n,m,p-1}^{s} + \sqrt{m(p+1)} \frac{u_y^s}{\alpha_y^s} C_{n,m-1,p+1}^{s} \]
they are satisfied at all times. In matrix form, Eqs. (6) and (7) become

$$\begin{align*}
\frac{q_s \omega_{cs}}{e\omega_{pe}} B_y & \star \left\{ \sqrt{\pi} \left( \frac{\alpha_s^y}{\alpha_2^y} - \frac{\alpha_s^z}{\alpha_2^z} \right) C_{n-1,m,p-1}^s + \sqrt{(n + 1)p} \frac{\alpha_s^y}{\alpha_2^y} C_{n+1,m,p-1}^s \right\} \\
\sqrt{n(p + 1)} \frac{\alpha_s^y}{\alpha_2^y} C_{n-1,m,p+1}^s + \sqrt{2p} \frac{u_s^y}{\alpha_s^y} C_{n,m,p-1}^s - \sqrt{2n} \frac{u_s^y}{\alpha_s^y} C_{n-1,m,p}^s \right\}_{k_x,k_y,k_z} \\
\sqrt{(n + 1)m} \frac{\alpha_s^x}{\alpha_2^x} C_{n+1,m-1,p}^s + \sqrt{2m} \frac{u_s^x}{\alpha_s^x} C_{n,m-1,p}^s - \sqrt{2m} \frac{u_s^x}{\alpha_s^x} C_{n,m-1,p}^s \right\}_{k_x,k_y,k_z},
\end{align*}$$

for the Vlasov equation, and

$$\begin{align*}
\frac{dB_{\delta}^{k_x,k_y,k_z}}{dt} & = -2\pi i\epsilon_{\beta_{\gamma}d} \frac{k_\beta}{L_\beta} E_{k_x,k_y,k_z}^\gamma, \\
\frac{dE_{\delta}^{k_x,k_y,k_z}}{dt} & = 2\pi i\epsilon_{\beta_{\gamma}d} \frac{k_\beta}{L_\beta} B_{k_x,k_y,k_z}^\gamma \sum_{s=x,t} \frac{q_s}{e} \alpha_s^e \alpha_s^e \left( \frac{\alpha_s^e}{\sqrt{2}} \right)^a_{b,c} + u_s^f C_{0,0,0}^{k_x,k_y,k_z,s} \right).
\end{align*}$$

for Maxwell’s equations (written in index notation with summation over repeated indices implied). Here \(\epsilon_{\beta_{\gamma}d}\) is the Levi-Civita tensor and \(\{a,b,c\} = (1,0,0), (0,1,0), (0,0,1)\) for \(\delta = x, y, z\), respectively. In Eq. (6), the convolution operator is defined as

$$[H \ast C_{n,m,p}]_{k_x,k_y,k_z} = \sum_{N_x/2}^{N_x/2} \sum_{N_y/2}^{N_y/2} \sum_{N_z/2}^{N_z/2} H_{k_x-k_x',k_y-k_y',k_z-k_z'} C_{n,m,p}^{k_x',k_y',k_z',s},$$

where \(H\) is an arbitrary function and the system is truncated by imposing \(C_{n,m,p}^s = 0\) for \(n \geq N_n, m \geq N_m, p \geq N_p\). It is worth noting that the Fourier spatial discretization preserves the divergence constrains (3) of Maxwell’s equations: if these constraints are satisfied at \(t = 0\), they are satisfied at all times. In frames, Eqs. (6) and (7) become

$$\begin{align*}
\frac{dC}{dt} & = \mathbb{L}_1 C + \mathcal{N}(C, F), \\
\frac{dF}{dt} & = \mathbb{L}_2 C + \mathbb{L}_3 F,
\end{align*}$$

with \(C\) representing the coefficients of the Hermite expansion and \(F\) the electromagnetic field. The matrix \(\mathbb{L}_1\) corresponds to the linear advection operator on the right hand side of Eq. (6), while \(\mathcal{N}\) is the non-linear operator associated with the convolutions. Finally, the matrices \(\mathbb{L}_2\) and \(\mathbb{L}_3\) are defined by the right hand side of Eqs. (7).

A common aspect of collisionless plasmas is the filamentation of phase space [24], i.e. the development of increasingly smaller phase-space structures. This implies that any numerical method would eventually run out of resolution. In the context of the Hermite basis, lack of resolution can lead to the well-known recurrence effect [25], and this is typically addressed by adding a collisional term to the right hand side of Eq. (6). We choose

$$C^{[k_x,k_y,k_z,s]} = -\nu \left[ \frac{n(n - 1)(n - 2)}{(N_n - 1)(N_n - 2)(N_n - 3)} + \frac{m(m - 1)(m - 2)}{(N_m - 1)(N_m - 2)(N_m - 3)} + \frac{p(p - 1)(p - 2)}{(N_p - 1)(N_p - 2)(N_p - 3)} \right] C_{n,m,p}^{k_x,k_y,k_z,s},$$

with

$$\nu = \frac{\pi}{(\lambda_n - 1)(\lambda_m - 1)(\lambda_p - 1)}.$$
essentially to damp the high-order modes of the Hermite series. Note that the operator in Eq. (10) does not act on the first three Hermite modes so that the conservation of total mass, momentum and energy remains valid in the discrete, as shown in Ref. [20].

Equations (6) and (7) are discretized in time by a fully-implicit, second-order accurate Crank-Nicolson scheme. In residual form, the resulting set of equations becomes

$$
\begin{align*}
\begin{cases}
R_1(C^{\theta+1}, F^{\theta+1}) = C^{\theta+1} - C^\theta - \Delta t \left[L_1 C^{\theta+1/2} + N(C^{\theta+1/2}, F^{\theta+1/2})\right] = 0 & \iff R(X) = 0,
R_2(C^{\theta+1}, F^{\theta+1}) = F^{\theta+1} - F^\theta - \Delta t L_2 C^{\theta+1/2} - \Delta t L_3 F^{\theta+1/2} = 0
\end{cases}
\end{align*}
$$

where we have introduced the time step $\Delta t$, used superscript $\theta$ to label time $C(\theta \Delta t) = C^\theta$, $C^{\theta+1/2} = (C^{\theta+1} + C^\theta)/2$ (and similarly for $F$), $X = \begin{bmatrix} C \\ F \end{bmatrix}$ and $R = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$.

2.1. Jacobian-Free Newton-Krylov solver

The most important part of the algorithm is the solution of the nonlinear Eqs. (11). For this, we use a Jacobian-Free Newton-Krylov (JFNK) method [26]. At each time step the initial guess $X^0$ is iteratively updated with the Newton method, $X^{n+1} = X^n + \Delta X^n$, where $n$ is the nonlinear iteration number and $\Delta X^n$ is the solution update, which is obtained by solving the linear system

$$
J(X^n) \Delta X^n = -R(X^n)
$$

with the Jacobian matrix $J$ defined as $J_{ij} = \frac{\partial R_i(X)}{\partial X_j}$, where $R_i$ labels the i-th row of $R$.

The linear system (12) is solved with the Generalized Minimal RESidual (GMRES) iterative method. It approximates the exact solution by minimizing the Euclidean norm of the residual $||J(X^n) \Delta X^n + R(X^n)||$. The method requires the computation of matrix-vector products $J \Delta X^n$, which can be approximated by a directional derivative with a small step increment without actually storing $J$ and computing the product.

2.2. Preconditioning

It is well known that the convergence of the linear part of the JFNK solver strongly depends on the problem’s eigenvalue spectrum, and that GMRES typically does not scale well as the problem’s size increases. Hence, applying a suitable preconditioner can significantly accelerate the convergence rate and reduce the simulation time.

By default PETSc uses left preconditioning, i.e. the preconditioner matrix $P_L^{-1}$ is applied to the left of system (12):

$$
P_L^{-1} J(X^n) \Delta X^n = -P_L^{-1} R(X^n).
$$

An efficient preconditioner can effectively reduce the number of linear iterations without being too costly to compute. In SPS, we follow [20] and construct the preconditioner by accounting for the convolutions in Eq. (11) via linearization relative to some reference solution

$$
P_L^{-1} = \left[ I - \frac{\Delta t}{2} L_1 - \frac{\Delta t}{2} N(\cdot,F^{ref}) - \frac{\Delta t}{2} N(C^{ref},\cdot) \right]^{-1},
$$

where $I$ is the identity matrix. In principle, the preconditioner can be applied at every time step, where the reference solution corresponds to the solution at the previous time step. In practice, we typically compute it only once at the beginning of the simulation (i.e. the reference solution is given by the initial condition of the system) and apply it at all time steps. While this can lead to a deterioration of the performance of the preconditioner as the simulation progresses, in the example discussed below this strategy was quite efficient.
3. Implementation of the SPS code

The SPS code has been implemented in Fortran to solve problems in a two-dimensional spatial geometry with three velocity components. Fortran was chosen for its native support for complex number operations that are widely used in the code. The SPS code consists of approximately 4,000 lines of code and is based on the PETSc [22] and FFTW [27] libraries, and on OpenMP.

We use the PETSc library to solve the discretized governing equations with the JFNK method. PETSc is an open-source library for the numerical solution of partial differential equations. It is written in C and has an interface to Fortran. Even though PETSc has been primarily designed to support programming on distributed systems, current PETSc versions use OpenMP and pthreads for matrix and vector operations.

The FFTW library [27] is used to perform the convolutions in Eq. (6) by means of two Fast Fourier Transforms. The convolution is carried out by first Fourier-transforming the input vectors from spectral space to real space, multiplying them in real space and transforming their result back to the Fourier space. The complexity of the FFT-based convolution is \( O(M \log M) \), where \( M \) is the input vector size, as opposed to the \( O(M^2) \) complexity of the direct calculation.

The OpenMP API is used to parallelize the SPS code on shared-memory systems. A profiling of the SPS performance revealed that the most compute-intensive part of the code is the evaluation of the residual function during the JFNK solver iterations. For this reason, we mainly use OpenMP to parallelize loops in this part of the calculation.

4. Verification of SPS: Orszag-Tang Vortex Test

The SPS code has been benchmarked on several problems, including Landau damping, two-stream instabilities, beam-plasma instabilities and ion acoustic waves. Here we briefly discuss a complex test case: simulations of the two-dimensional Orszag-Tang vortex. The Orszag-Tang vortex is an initial configuration often used in studies of 2D plasma turbulence (e.g. [28, 29]). Evolution of the system leads to the formation of current sheets, which are narrow regions of intense current density corresponding to large gradients of the magnetic field. If the system size is large enough, the system eventually transitions to turbulence. We focus on collisionless magnetized plasmas with moderate ratios of thermal to magnetic pressures \( \beta_e \sim 1 \), where the characteristic thickness of the current sheets is expected to be of the order of \( d_s \sim \rho_e \). Here \( \beta_e = 8\pi n_e T_e / B_0^2 \), \( \rho_e = v_{te} / \omega_{pe} \), \( v_{te} = \sqrt{T_e / m_e} \), \( n_e \approx n_i \) are the electron and ion densities, and \( T_e \) is the effective temperature of species \( s \).

We consider a domain of size \( L_x \times L_y \), with the initial distribution function given by

\[
N_s(x, v, t = 0) = \frac{1}{(2\pi)^{3/2} v_{ts}^3} \exp \left( -\frac{|v - U_s(x)|^2}{2 v_{ts}^2} \right)
\]

and the initial magnetic field by \( B = e_z + \delta B(x) \), where \( U_s = U_0 [\sin(k_i y) e_x + \sin(k_x x) e_y] \) and \( \delta B = \frac{\delta B}{B_0} [\sin(k_y y) e_x + \sin(2k_x x) e_y] \), with \( e_\beta \) the unit vector along \( \beta \). Here \( U_0 = V_A \delta B / B_0 \), \( V_A = B_0 / \sqrt{4\pi n_0 m_i} \), \( k_x = 2\pi / L_x \), \( k_y = 2\pi / L_y \). To satisfy Ampere’s law, the initial configuration includes the electron flow \( U_{e,x} = -\frac{\delta B}{B_0} \omega_{pe} [2k_x \cos(2k_x x) + k_y \cos(k_y y)] \). The ion flow is \( U_{i,x} = 0 \). The following physical parameters were used: domain size \( L_x \times L_y = 10d_i \), \( m_i / m_e = 25 \), \( T_e = T_i \), \( \omega_{pe} / \omega_{ci} = 2 \), \( \beta_e = \beta_i = 0.25 \), \( \delta B / B_0 = 0.2 \).

For the SPS simulations, the configuration (15) is expanded in Hermite-Fourier basis functions numerically. We choose \( \alpha^s_\beta = \sqrt{2} v_{ts} \) and \( u^s_\beta = 0 \). Our representation with \( u^s_\beta const \) cannot capture Eq. (15) exactly with only one polynomial, therefore runs conducted with different values of \( N = N_n = N_m = N_p \) correspond to slightly different approximations of the initial condition. Other parameters used for the simulations are: \( N_x = N_y = 128 \), \( \omega_{pe} \Delta t = 1 \) and \( \nu = 1 \), while \( N \) will be varied parametrically. The results of the SPS simulations are compared against a reference solution obtained using a fully kinetic particle-in-cell (PIC) simulation of the same
The PIC simulation was conducted using the explicit relativistic simulation code VPIC [30]. The VPIC simulation with the parameters specified above had 880 × 880 cells with 4000 particles per cell per species at $t = 0$. The PIC time step was $\omega_{pe} \Delta t \approx 0.04$.

Figure 1 compares the current density $j_z$ normalized to a characteristic value $en_0 v_t$ at $\omega_{pe} t = L_x/V_A$ in two SPS simulations obtained with $N = 4$ and $N = 6$, against the reference PIC simulation. At this time the system has not transitioned to fully developed turbulence yet. Despite a relatively small number of Hermite polynomials, the test cases successfully reproduce both the large-scale structure observed in the PIC simulation and the small, intense current sheets. Some of the secondary features are arguably reproduced better in the case with larger number of polynomials in the basis. The multi-scale nature of fluctuations excited in the system is apparent from an estimate of the omnidirectional power spectrum of magnetic fluctuations $S_B$ presented in Fig. 2. The spectrum also clearly demonstrates potential advantages offered by noise-free spectral methods for turbulence simulations. While the spectrum at very high $k$ is eventually dominated by discrete particle noise even in this highly resolved PIC simulation, SPS can provide clean estimates of power in large-$k$ modes. It is interesting to note that both PIC and SPS simulations suggest that power spectra extend to relatively small scales $kd_e > 1$, with a possible transition to another power law at $k\lambda_D \sim 1$. A detailed investigation of energy dissipation and fluctuation spectra in these simulations will be presented elsewhere.

Figure 1. Normalized current density $\hat{j}_z = j_z/(en_0 v_t)$ in the Orszag-Tang simulations at $\omega_{pe} t = L_x/V_A$. Left: Reference PIC simulation; Middle: SPS simulation with $N_x = N_y = 128$ and $N=4$; Right: SPS simulation with $N_x = N_y = 128$ and $N=6$.

The global energy balance in the Orszag-Tang vortex simulations is illustrated in Fig. 3. SPS reproduces well the changes in the different energy channels measured in the reference PIC simulation. We note that in contrast to fluid and MHD turbulence, dissipation of fluctuating energy in weakly collisional plasmas is due to collective interactions of particles with the electromagnetic field, even though the ultimate dissipation in the sense of entropy generation is provided by collisions. Understanding of the turbulent energy dissipation is thus quite challenging and has been a focus of intense research effort. The methodology presented here may make a significant contribution to the field by enabling accurate simulations of kinetic phenomena in turbulence, free of inherent limitations of other currently available methods.

5. Performance Results

In this section we discuss the performance of SPS. The simulations were performed on a node of the Wolf supercomputer at the Los Alamos National Laboratory. Each Wolf node has two 8-core Intel Xeon E5-2670 processors with 2.6 GHz CPU and 64 GB of memory.
Figure 2. Omnidirectional power spectrum of magnetic fluctuations in the Orszag-Tang vortex simulations at the same time as in Fig. 1. The black dotted line shows the spectrum of thermal noise in the PIC simulation. The spectrum is normalized according to $\int B^2 dxdy = \int S_B(k) dk$.

Figure 3. Changes in the kinetic energy of electrons (e), ions (i), and of the magnetic field energy (B) in the Orszag-Tang vortex simulations. Here $\delta W(t) = W(t) - W(t = 0)$, $W_{\text{kin},s} = \int (m_s v_s^2/2) f_s d^3 v d^2 x$, $W_B = \int (B^2/2) d^2 x$, and $\omega_{pe} t = L_x/V_A$. The energies are normalized to the energy of the initial perturbation $W_0 = \int \left[ (m_i U_i^2/2 + m_e U_e^2/2 + \delta B^2/2) \right] d^2 x$. SPS simulation had $N_x = N_y = 128$ and $N = 6$.

The SPS code has been compiled with the Intel Fortran 15.0.5 compiler and linked to PETSc version 3.6.2 and to FFTW version 3.3.4. The relative and absolute tolerances of JFNK are set to $10^{-6}$. PETSc Incomplete LU (ILU) factorization is used to invert matrix $P_L$ for preconditioning. For preconditioned and unpreconditioned simulations, we set the PETSc options `-snes_mf_operator` and `-snes_mf`, respectively.

5.1. Performance of the preconditioner
Table 1 shows the running time and the average number of nonlinear and linear iterations per time step for the Orszag-Tang problem. The simulation parameters are described in Section 4, except that $N_x = N_y = 32$ and the simulation time corresponds to 100 time steps. Let us focus on the unpreconditioned case. For $N = 4$ the average number of linear iterations per time step is $\sim 14$ and the running time is 297 s. As we double $N$, the problem’s size increases by a factor 7.7 but the running time is $\sim 14$ times longer since the average number of linear iterations doubles.
Going from $N = 8$ to $N = 12$ increases the problem’s size by a factor 3.4 while the running time increases by a factor 5.3, again because of the higher number of linear iterations (now $\sim 52$) required for convergence. Let us now look at the preconditioned case. For $N = 4$, the average number of linear iterations is 5, about a third of those for the unpreconditioned case, translating into a 40% reduction of the running time. Up to $N = 16$ the number of linear iterations remains fairly constant and, consequently, the running time shows a fairly good linear scaling with the number of unknowns of the system. For $N = 12$ there is a factor of 4 speed-up relative to the unpreconditioned case and the gain will increase for larger problems. The preconditioner loses performance for $N = 20$, where the average number of linear iterations rises to 8.5 with a 20% increase in the running time relative to a linear scaling. This could arise from the problem not being exactly the same as $N$ changes, due to the approximation of the initial condition and the nature of Eq. (10). Finally, the number of nonlinear iterations is insensitive to preconditioning.

Table 1. Running time and average number of nonlinear and linear iterations per time step for the Orszag-Tang problem with $N_x = N_y = 32$ and 16 threads.

| N  | Unknowns | Time, s | Nonlin./ Lin. iter. | Time, s | Nonlin./ Lin. iter. |
|----|----------|--------|---------------------|--------|---------------------|
| 4  | 145,926  | 178    | 2/5.1              | 297    | 2/14.4             |
| 8  | 1,121,670| 1,475  | 2/5.4              | 4,138  | 2/30.8             |
| 12 | 3,770,118| 5,165  | 2/5.6              | 22,120 | 2/51.6             |
| 16 | 8,927,622| 13,130 | 2/5.9              |        |                    |
| 20 | 17,430,534| 31,200 | 2/8.5              |        |                    |

5.2. Parallel performance on shared-memory systems

A strong scaling test, where we kept the problem size fixed and increased the number of threads, has been carried out for the Orszag-Tang problem. The parameters are the same as those used for the preconditioning studies in the previous subsection and we focus on the case with $N = 8$. Table 2 shows the parallel speed-up varying the number of threads in use: the strong scaling tests show a factor of 10 speed-up with 16 threads. In addition, Amdahl’s law [31]:

$$\text{Speed-up} = \frac{1}{S + \frac{1 - S}{p}}$$

is used to evaluate the performance of SPS. In Eq. (16), $S$ is the fraction of time in the code that is not parallelized and $p$ is the number of threads. For the Orszag-Tang test, $S = 0.033$ was calculated by running the code with 1 thread and using Intel Vtune 15.3.0 for profiling. Table 2 shows that there is good agreement between the theoretical and measured speed-up.

6. Conclusions

We have presented the design and implementation of the SPS code, which solves the Vlasov-Maxwell equations for a plasma in multiple dimensions. The code is based on a Hermite-Fourier spectral decomposition of the particle distribution function. The resulting set of non-linear ODEs is discretized in time with a second-order accurate Crank-Nicolson scheme and is solved numerically with a preconditioned JFNK solver.

SPS is written in Fortran and uses the PETSc library for solving the nonlinear equations and the FFTW library to handle the convolution operations efficiently. The code is parallelized for
shared-memory machines using OpenMP. A comparison against a fully kinetic PIC simulation for a Orszag-Tang vortex problem shows that SPS can capture reasonably well the large-scale and small-scale current structure of the system, despite using a relative low number of Hermite modes. It also highlights the strength of the method and its ability to capture the high-k tail of the fluctuation spectrum which is normally obscured by noise in PIC simulations. Performance tests have also been presented on the same problem, showing a significant reduction of the number of linear iterations and of the running time obtained by preconditioning, and a factor of 10 speed-up for a strong scaling test with 16 threads.

Acknowledgments
This work was funded by the Laboratory Directed Research and Development (LDRD) program, under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy by Los Alamos National Laboratory, operated by Los Alamos National Security LLC under contract DE-AC52-06NA25396. This research used resources provided by the Los Alamos National Laboratory Institutional Computing Program. VR acknowledges support from NASA grant NNX15AR16G. PIC simulations utilized resources provided by the NASA High-End Computing Program through the NASA Advanced Supercomputing Division at Ames Research Center.

References
[1] Goldston R J and Rutherford P H 1995 Introduction to Plasma Physics Plasma Physics Series (Institute of Physics Publications)
[2] Fitzpatrick R 2014 Plasma Physics (CRC Press)
[3] Kolobov V and Arslanbekov R 2012 Journal of Computational Physics 231 839 – 869
[4] Markidis S, Henri P, Lapenta J, Rönnmark K, Hamrin M, Meliani Z and Laure E 2014 Journal of Computational Physics 271 415–429 frontiers in Computational Physics Modeling the Earth System
[5] Daldorff L K S, Tóth G, Gombosi T I, Lapenta G, Amaya J, Markidis S and Brackbill J U 2014 Journal of Computational Physics 268 236–254
[6] Birdsall C K and Langdon A B 2004 Plasma Physics Via Computer Simulation (Taylor & Francis)
[7] Hockney R and Eastwood J 1988 Computer Simulation Using Particles (Taylor & Francis)
[8] Cheng C Z and Knorr G 1976 Journal of Computational Physics 22 330–351
[9] Filbet F, Sonnendrucker E and Bertrand P 2001 Journal of Computational Physics 172 166–187
[10] Vay J L, Colella P, Kwan J W, McCrorquodale P, Serafini D B, Friedman A, Grote D P, Westenskow G, Adam J C, Hron A and Haber I 2004 Physics of Plasmas 11
[11] Fujimoto K and Machida S 2006 Journal of Computational Physics 214 550 – 566
[12] Arslanbekov R R, Kolobov V I and Frolova A A 2013 Phys. Rev. E 88(6) 063301
[13] Armstrong T P, Harding R C, Knorr G and Montgomery D 1970 Methods Comput. Phys. 9 29
[14] Camporeale E, Delzanno G L, Lapenta G and Daughton W 2006 Physics of Plasmas 13 092110
[15] Vencels J, Delzanno G L, Johnson A, Peng I B, Laure E and Markidis S 2015 Procedia Computer Science 51 1148–1157
[16] Manzini G, Delzanno G L, Vencels J and Markidis S 2015 J. Comput. Phys. (Submitted)
[17] Parker J T and Dellar P J 2015 Journal of Plasma Physics 81(02)
[18] Loureiro N F, Schekochihin A A and Zooco A 2013 Physical Review Letters 111(2) 025002

Table 2. Strong scaling test: parallel speed-up varying the number of threads for the Orszag-Tang problem with \( N_x = N_y = 32, \ N = 8 \) and with preconditioning.

| Threads | Speed-up | Amdahl’s law |
|---------|----------|--------------|
| 1       | 1        | 1            |
| 2       | 1.9      | 1.9          |
| 4       | 3.6      | 3.6          |
| 8       | 6.5      | 6.5          |
| 16      | 10.5     | 10.7         |
[19] Camporeale E, Delzanno G, Bergen B and Moulton J 2016 *Computer Physics Communications* **198** 47–58
[20] Delzanno G L 2015 *Journal of Computational Physics* **301** 338–356
[21] Loureiro N F, Dorland W, Fazendeiro L, Kanekar A, Mallet A, Vilelas M S and Zocco A 2015 *arXiv:1505.02649*
[22] Balay S, Abhyankar S, Adams M F, Brown J, Brune P, Buschelman K, Dalcin L, Eijkhout V, Gropp W D, Kaushik D, Knepley M G, McInnes L C, Rupp K, Smith B F, Zampini S and Zhang H 2015 PETSc Web page URL http://www.mcs.anl.gov/petsc
[23] Holloway J P 1996 *Transport Theory and Statistical Physics* **25** 1–32
[24] Klimas A J 1987 *Journal of Computational Physics* **202** 222–226
[25] Joyce G, Knorr G and Meier H K 1971 *Journal of Computational Physics* **8** 53 – 63
[26] Knoll D A and Keyes D E 2004 *Journal of Computational Physics* **193** 357–397
[27] Frigo M and Johnson S G 2005 *Proceedings of the IEEE* **93** 216–231
[28] Orszag S A and Tang C M 1979 *Journal of Fluid Mechanics* **90** 129–143
[29] Biskamp D and Welter H 1989 *Physics of Fluids B* **1** 1964–1979
[30] Bowers K J, Albright B J, Yin L, Bergen B and Kwan T J T 2008 *Physics of Plasmas* **15** 055703
[31] Amdahl G M 1967 Validity of the single processor approach to achieving large scale computing capabilities

*Proceedings of the April 18-20, 1967, spring joint computer conference (ACM)* pp 483–485