Embedded grid refinement for Semi-Lagrangian parallelized relativistic Vlasov-Maxwell solver

Maxence Antoine, A. Ghizzo, Erwan Deriaz, Daniele Del Sarto

Physics of Plasmas

RESEARCH ARTICLE | APRIL 10 2025

Embedded grid refinement for semi-Lagrangian parallelized relativistic Vlasov–Maxwell solver

M. Antoine ; A. Ghizzo ; E. Deriaz ; D. Del Sarto

Phys. Plasmas 32, 043905 (2025)
https://doi.org/10.1063/5.0252704

Embedded grid refinement for semi-Lagrangian parallelized relativistic Vlasov–Maxwell solver

Cite as: Phys. Plasmas **32**, 043905 (2025); doi: 10.1063/5.0252704
Submitted: 11 December 2024 · Accepted: 26 March 2025 ·
Published Online: 10 April 2025

M. Antoine,[1,a)] A. Ghizzo,[1] E. Deriaz,[2] and D. Del Sarto[1]

AFFILIATIONS

[1]Université de Lorraine, CNRS, IJL, F-54000 Nancy, France
[2]International Innovation Institute, Beihang University, Hangzhou 311115, Yuhang District, China

[a)]Author to whom correspondence should be addressed: antoine.maxence@hotmail.com

ABSTRACT

We present a new dynamic embedded grid refinement method performed only on the momentum coordinates, which we applied to a relativistic Vlasov–Maxwell model. This refinement strategy, which maintains a uniform space coordinate grid, differs from the one usually adopted in particle in cell (PIC) code. It involves embedded rectangles and may evolve over time. From an algorithmic point of view, it displays the further advantage of allowing one to preserve MPI/OpenMP parallelization strategies possibly implemented in equivalent Vlasov semi-Lagrangian solvers on an Eulerian grid, although it requires a change in the boundary conditions of the interpolation algorithm and in the computing of integrals. A comparison between two versions of the same code, with and without mesh refinement, is here shown for a transition between the oblique Weibel-type instability to the current filamentation instability for 2D2V and 2D3V geometries in both nonrelativistic and relativistic regimes. The mesh refinement method is perfectly adapted to follow the formation of thin filaments in the momentum space associated with the energy transfer in the Weibel-like instabilities and allows a remarkable computational gain.

I. INTRODUCTION

In collisionless electromagnetic plasma physics, to follow the evolution of the distribution function of the species, we have to solve the Vlasov equation coupled with Maxwell's equations. The resolution of this system requires the use of numerical methods where among the main ones we can cite here: particle in cell (PIC)[1,2] and semi-Lagrangian (SL).[3–5] The former one is more commonly used because it is often cheaper in terms of computational time, but due to the finite-size superparticles there is a numerical noise. In comparison, SL methods often require a large computational effort but have no statistical noise.[6] Therefore, reducing the computational cost of SL methods is interesting.

Concerning the PIC method, several codes based on an explicit approach in time have been developed, e.g., OSIRIS,[7] EPOCH,[8] SHARP,[9] SMILEI,[2] VPIC,[10] WARPX,[11] or PHARE.[12] The latter includes an adaptive mesh refinement (AMR) method. Severe limitations may arise on the applicability of explicit codes, where the time step and the grid spacing are determined by the most restrictive plasma conditions in the whole simulation domain. When large density gradients are involved, the local skin depth (d_e) and the inverse plasma frequency (ω_p^{-1}) could vary dramatically within the domain, and interesting physics may take place only in very localized regions where $d_e = c/\omega_p$ and ω_p^{-1} are very small. Therefore, only those regions would require finer grids and smaller time steps, but explicit methods have the drawback of imposing restrictive simulation parameters in the whole domain. For these reasons, extensive research has been devoted to the development of implicit PIC codes. These essentially involve an implicit discretization of Maxwell's equations, together with a particle pusher that may be nonlinearly coupled (resulting in the so-called full implicit PIC solver) or may not be coupled to the electromagnetic field solver (leading in the last case to the semi-implicit PIC codes like those of Refs. 13 and 14).

Another type of scheme is the finite volume method, where the discrete unknowns are now the averages of the distribution function on volumes paving the phase space. Taking the form of a flux balance, this method was first introduced to the Vlasov equation by Fijalkow.[15] An improved version is the positive and flux conservative (PFC) method[16] or more recently,[17,18] which is not only conservative but also preserves the positivity and the maximum of the distribution function. In addition to the high dimensionality (the Eulerian or SL descriptions being computationally much more demanding due to the six-dimensional (6D) phase space), a further challenge stems from the fact that

important plasma features can occur on small scales, in a way that they only start to appear at fine resolutions in the simulation. Due to the importance of the high dimensionality and small scales physics (as met for instance in Weibel type instabilities), it becomes obvious that simulations that are resolved finely enough may become too expensive in CPU time and/or memory even for today's HPC ressources. The semi-Lagrangian approach thus constitutes a third technique compared to the two standard methods mentioned earlier. It retains the advantages of both Eulerian and Lagrangian methods while avoiding some of their drawbacks. Unlike the Eulerian method, it does not require a Courant–Friedrichs–Lewy (CFL) condition on the time step, which necessitates a significant reduction of the time step in the Eulerian approach. Moreover, it is inherently parallelizable due to its Eulerian structure. Compared to particle-in-cell (PIC) codes, it is considered "noiseless" in the statistical sense, as it satisfies the condition of a graininess parameter g equal to zero.

Other types of semi-Lagrangian models have been developed, notably introducing a forward integration of characteristics,[19] as generally done in explicit PIC codes. The AMR technique presented here maintains a standard formulation and will only be used for the integration of the distribution function in the momentum space. It relies on a technique for tracing the feet of characteristics backward, a method that has proven effective. A refinement criterion for nested grids is introduced, particularly based on a condition on the L-infinity norm. This effectively amounts to using a forward characteristics technique to determine, in a first step, the regions of momentum space where grid refinement is necessary.

Coming back to the semi-Lagrangian (SL) approach, the original algorithm by Chen and Knorr[20] has been cast into the more general framework of SL methods by Sonnendrücker *et al.*[21] and Nakamura and Yabe.[22] Hamiltonian splitting methods have been used in Ref. 23. The semi-Lagrangian scheme uses two main steps.

First, the transport of the distribution f is made using the SL backward characteristics, and then the value of f at the origin of the characteristics is computed by reconstructing f from the mesh points using an interpolation technique.

The PIC and SL approaches are thus two complementary techniques, both of which solve the Vlasov equation under the assumption that the grain parameter $g = 1/(n_0\lambda_D^3)$, with n_0 as the density and λ_D as the Debye length, tends to zero and both describe a fundamental property of the Vlasov equation: the filamentation of f in momentum space. While PIC codes are little affected by the filamentation problem (but at the price of a considerable increase in plasma thermal noise due to the reintroduction of grain effects (with a factor $g_{PIC} \neq 0$) related to the finite size of the superparticles), SL or Eulerian codes generally require high sampling in the momentum space to follow the dynamics of the increasingly thin filaments that appear therein. The problem of filamentation is an intrinsic property of the Vlasov equation contained in the free streaming term.[24] The source of filamentation lies in the treatment of the free-streaming term $\frac{\partial f}{\partial t} + \frac{\mathbf{p}}{m\gamma} \cdot \nabla_x f = 0$, whose exact solution in the Fourier space reads as $f(\mathbf{k}, \mathbf{p}, t) = f(\mathbf{k}, \mathbf{p}, 0)\exp(i\frac{\mathbf{k}\cdot\mathbf{p}t}{m\gamma(\mathbf{p})})$. Instabilities can lead to the possibility of enhancing the filamentation process.[25] The ongoing (relativistic) filamentation leads to stronger and stronger momentum gradients, whose inaccurate treatment leads to oscillations and finally to numerical instabilities.

In PIC codes, the processing is finally achieved by re-introducing a discrete representation of the phase space, from the concept of superparticles, in contradiction in some way with the non-grained character of an ideal continuous collisionless solution. While the PIC method involves a smoothing of information (which is in principle numerically conserved during the dynamics), the SL scheme invariably implies a loss of information linked to the emergence of thin structures that may reach the size of the numerical grid. Free-streaming filamentation (typically of kinematic nature) is thus well resolved in PIC codes, and the Lagrangian scheme is very well adapted to study the mechanism of filamentation amplification driven by Weibel-type or current filamentation instabilities (provided that there are enough particles in the affected regions), but a price must be paid in terms of the corresponding amplified thermal noise. The numerical noise, however, limits the efficiency of PIC codes to model resonant wave-particle Landau-type interactions in the tail of the distribution function, where the small number of superparticles does not allow an accurate description of the particle trapping. Instead, Vlasov codes provide a powerful tool for low noise studies of collisionless plasmas with a fine resolution of the phase space including those regions where trapping occurs or where particles move at velocities close to the phase wave velocity. A comparison between the two standard approaches, namely, the Lagrangian particle-in-cell (PIC) method (with the forward characteristic determination, a pioneering technique used in Dawson 1983[1]) and a classical "semi-Lagrangian" method (which uses a backward tracing of the characteristics of the Vlasov equation, as previously applied in Cheng and Knorr 1976[20]) provides an estimation of the resources required to tackle a problem with a given "graininess parameter" (imposed by the specific problem). Of course, these estimates offer a fairly general idea considering the advancements of various techniques since the early days of plasma modeling, where the methods employed for both PIC codes and Vlasov simulations have significantly evolved (notably, semi-Lagrangian methods can now use forward characteristics, and implicit PIC codes can utilize backward characteristics). Nevertheless, this method provides a rough order of magnitude. Assuming an equivalent sampling of the spatial coordinate (i.e., the configuration space), the ratio of the numerical efforts of a Vlasov code to a PIC code can be estimated by $N_{Vlas}/N_{part} = g_{PIC}N_p^{d_p} \sim 1$,[26,27] where N_{Vlas} denotes the overall phase-space sampling used in the Vlasov solver, N_{part} denotes the number of superparticles by cell, and d_p denotes the number of momentum dimensions. Typically, the sampling in the one-dimensional space N_{p_x} is close to 100. Consequently, the effort is given by Table I, reproduced from Ref. 26.

In Table I, both Vlasov and PIC codes use the same sampling in the configuration space but not in the whole phase-space. In some physical situation, we can take $g_{PIC} \sim 10^{-2}$ to study basic phenomena. While the numerical efforts are equivalent on the diagonal, we see clearly that the Vlasov code is more expensive in 2D and 3D

TABLE I. PIC vs semi-Lagrangian Vlasov code: ratio N_{Vlas}/N_{part}.

N_{Vlas}/N_{part}	$d_{\mathbf{p}} = 1$	$d_{\mathbf{p}} = 2$	$d_{\mathbf{p}} = 3$
$g_{PIC} = 10^{-2}$	1	10^2	10^4
$g_{PIC} = 10^{-4}$	10^{-2}	1	10^2
$g_{PIC} = 10^{-6}$	10^{-4}	10^{-2}	1

momentum space. In the 3D case, the treatment of kinetic effects can be investigated by both methods with an equivalent computational cost, but only for a smaller value of g_{PIC}, i.e., $g_{PIC} \sim 10^{-6}$, which corresponds to a large number of macro-particles, seldom used in PIC codes. Thus, to address a physical problem that requires a relatively small graininess parameter on the order of 10^{-6}, the CPU times become very similar between PIC codes and Vlasov codes for accurately describing wave-particle interactions (with an equivalent level of description). It becomes evident that, in the case of semi-Lagrangian techniques, a significant limitation is imposed by the necessity to use high sampling, primarily in the momentum space in 3D dimensions. Therefore, an AMR-type technique, or nested grids, restricted solely to momentum space, allows for the alleviation of this constraint. Generally, AMR techniques are usually applied in the global phase space, meaning both in momentum space and in configuration space, which can lead to additional difficulties in handling Maxwell equations. An AMR approach, limited to the momentum space alone, helps to simplify the problem.

While in one dimension in the momentum space, the semi-Lagrangian code offers better capabilities in terms of description of wave-particle interactions and accuracy, the treatment of plasma dynamics for a higher dimensionality in the momentum space constitutes a technical obstacle and a real challenge in computational physics. Thus, PIC codes and Vlasov codes (semi-Lagrangian) are two complementary approaches that both solve the Vlasov equation, which corresponds to the case where the grain parameter g is zero (the Vlasov equation being a continuous system). Therefore, the use of finite-size superparticles reintroduces a non-zero grain parameter (g_{PIC}), while g is zero in the Eulerian approach. It is worth noting that in real situations, g remains finite but very small, on the order of 10^{-9} for astrophysical plasmas.

One step toward improving the efficiency of Vlasov codes is the implementation of adaptative grid methods which can enhance the resolution in the momentum space, while saving the computational CPU time. A method that may be compatible with MPI/OpenMP is the use of AMR. The aim is to locally refine the mesh of a numerical problem only where more accuracy is needed. If the region of the domain where the refinement is needed is small enough with respect to the whole domain, the computational effort to perform the AMR can be largely compensated by the gain associated with the smaller number of points needed to achieve an equivalent resolution in a uniform mesh. When the refinement of the mesh evolves over time, it is referred to as dynamically/automatically AMR. Some codes have already tried some AMR methods for Vlasov–Poisson,[28,29] and for non-relativistic and—more recently—also for relativistic Vlasov–Maxwell codes.[27,30–33] We can differentiate two types of AMR methods: tree methods[29] and patch methods.[34] The mesh refinement we use here relies on a patch method.

This work presents a dynamically adaptive mesh refinement method only in momentum. This method differs from traditional refinement methods applied to the coordinate space, like it is the case for most PIC codes[12,35,36] and Vlasov codes.[27–29,31] Using a local error estimate for measuring the rapid variation in the solution, we may identify regions of interest, i.e., the "grids" that should be refined. The refinement process can be repeated recursively to form a hierarchy of refinement levels, each composed of a rectangular patch/grid.

While standard AMR methods require at least two types of mesh hierarchies—one confined to the configuration space ($\boldsymbol{x}$) and another in the momentum space ($\boldsymbol{p}$)—the AMR technique employed here uses a single hierarchy limited to the treatment of the momentum space. This approach offers two major advantages: the first relates to the absence of nested grids in configuration space, thereby avoiding instabilities caused by boundaries where electromagnetic waves may reflect, leading to various numerical instabilities (see Refs. 35 and 36) The second, of a more fundamental nature, allows for the consideration of the filamentation process of the distribution function in velocity space (or momentum space), which inevitably leads to the emergence of increasingly thinner and thinner filaments. This can induce, through weak turbulence, a heating process of the distribution function as observed in Refs. 25 and 37.

We adapted this method to VLEM (VLasov ElectroMagnetic) solver, a fully parallelized relativistic Vlasov–Maxwell code.[4,5] The AMR upgrade of VLEM has been made for VLEM2D2V first and then for VLEM2D3V and is expected to be made for VLEM3D3V soon.

In the Sec. II, we will recall the architecture of VLEM with the time-splitting scheme and the field computations. In Sec. III, we will describe the way we implement the rectangular embedded grid mesh refinement method. To show an application of the AMR solver to a physics test case, we consider a case of nonlinear evolution of beam-plasma instabilities of Weibel/two-stream type.

Previous linear studies of the Weibel-type instabilities have identified different classes of electron beam-plasma instabilities: the current filamentation instability (CFI)[38] with a wave vector $\mathbf{k}$ perpendicular to the beam direction, the (electrostatic) two-stream instability (TSI) with $\mathbf{k}$ parallel to the beams, and finally the oblique instability (OI)[39,40] for an arbitrary direction of $\mathbf{k}$. The special case of two counter-propagating beams is relevant to astrophysical plasmas, or the gamma-ray burst production scenario,[41] and in interpenetrating plasma flows in Ref. 42. Such physical instabilities are linked to the emergence of a spatial filamentation of f that makes an impact on the kinematic filamentation property of the Vlasov equation. Thus, a turbulent cascade can take place, which produces thinner and thinner filaments first in the configuration space, and finally in the phase space. Such a process induces an enhancement of the filamentation of f (a kind of "augmented" filamentation[25]) that leads to strong modifications in the energy transfer and can lead to a stochastic plasma heating. The emergence of these nonlinear effects introduces asymmetries (element of irreversibility) between two fundamental aspects of the Vlasov equation: the filamentation of the distribution function related to the time reversibility of the Vlasov equation and the transfer of the magnetic energy into kinetic energy, a process already observed in magnetic reconnection.

The specific choice of an AMR technique only in momemtum space, allows us to highlight the multiscale character of the $\mathbf{p}$ space as well and the possibility to follow the phenomenon of filamentation of the distribution in space. To this purpose, two test cases for OI and CFI instabilities will be examined, to compare the accuracy and acceleration of the new versions of VLEM. The first case will be a low relativistic regime, which we will study with an AMR version of VLEM2D2V and VLEM2D3V and the second will be a relativistic case, which we will only study in the two-dimensional momentum space (2D2V).

II. VLEM CODE

A. Basic equations

VLEM is a semi-Lagrangian code that solves the evolution of the electron distribution function, where the ions are considered as a fixed background, with the relativistic Vlasov equation

$$\frac{\partial f}{\partial t}+\frac{\mathbf{p}}{m\gamma}\cdot\nabla_{\mathbf{x}}f+e\left(\mathbf{E}+\frac{\mathbf{p}\times\mathbf{B}}{m\gamma}\right)\cdot\nabla_{\mathbf{p}}f=0, \tag{1}$$

with γ the Lorentz factor that can be written as

$$\gamma=\sqrt{1+\frac{\mathbf{p}^2}{m^2c^2}}. \tag{2}$$

From Eq. (1), one also obtain the differential form of the conservative Vlasov equation

$$\frac{df}{dt}=\frac{\partial f}{\partial t}+\nabla_{\mathbf{x}}\cdot\left(\frac{\mathbf{p}}{m\gamma}f\right)+\nabla_{\mathbf{p}}\cdot\left[e\left(\mathbf{E}+\frac{\mathbf{p}\times\mathbf{B}}{m\gamma}\right)f\right]=0. \tag{3}$$

By introducing $\nabla=(\nabla_{\mathbf{x}},\nabla_{\mathbf{p}})$ the six-dimensional phase space vector

$$\begin{aligned}\mathbf{U}&=(\mathbf{U}_{\mathbf{x}},\mathbf{U}_{\mathbf{p}})=(\dot{\mathbf{x}},\dot{\mathbf{p}})\\&=\left(\mathbf{V}=\frac{\mathbf{p}}{m\gamma},\mathbf{F}=e\left(\mathbf{E}+\frac{\mathbf{p}\times\mathbf{B}}{m\gamma}\right)\right)\end{aligned} \tag{4}$$

the six-dimensional phase space flow vector, the Vlasov equation takes a conservative form

$$\frac{df}{dt}=\frac{\partial f}{\partial t}+\nabla\cdot(\mathbf{U}f)=0. \tag{5}$$

Both conservative form (3) and advective form (1) are strictly equivalent thanks to the Liouville's theorem.

In Ref. 4, the numerical resolution of VLEM is explained in detail, but we will give a short summary here. Different versions of the code exist but we will focus on the 2D2V and 2D3V [i.e., two dimensions for the space and three for the momentum (velocity)]. This is motivated by the fact that 3D codes require much more computational resources.

VLEM combines Eq. (1) with three Maxwell equations, Eqs. (6) and (7) to solve the evolution of the electromagnetic field, and Eq. (8) to add a correction to the electric field and ensure a better conservation of the energy

$$\frac{\partial B}{\partial t}=-\mathrm{rot}(\mathbf{E}), \tag{6}$$

$$\frac{\partial\mathbf{E}}{\partial t}=\mathrm{c}^2\mathrm{rot}(\mathbf{B})-\mathrm{c}^2\mu_0\mathbf{J}, \tag{7}$$

$$\mathrm{div}\mathbf{E}=\frac{\rho}{\varepsilon_0}. \tag{8}$$

The current can be computed with the integral

$$\mathbf{J}=e\int\frac{\mathbf{p}}{m\gamma}fd^3p, \tag{9}$$

together with

$$\rho=e\int fd^3p-en_0, \tag{10}$$

where n_0 is the fixed ion density, and with charge conservation condition

$$\frac{\partial\rho}{\partial t}+\mathrm{div}\mathbf{J}=0. \tag{11}$$

The various variables in the code are normalized: this normalization is determined by the choice of normalizing two variables, velocity (v) and time (t), which are normalized, respectively, to the light velocity in vacuum c, due to special relativity, and inversely to the plasma frequency defined by $\omega_p=\sqrt{ne^2/m\varepsilon_0}$. This choice leads to the normalization of the different quantities that appear in the code. Thus, the momentum (p) is normalized to mc, the electric E and magnetic B fields components are normalized to $m\omega_p c/e$ and $m\omega_p/e$ respectively, the mean density n is normalized to the ion density n_0 assumed constant, and the kinetic energy density is normalized to n_0mc^2.

It should also be noted that the normalization of the distribution function depends on the dimensionality of the momentum space. In the case of 3D, the distribution function is normalized to m^3c^3/n_0, which allows for the description of relativistic distributions of the Maxwell–Jüttner type in the general form (3D)

$$f(\mathbf{p})=\frac{n_0}{4\,\pi\,m^3\,c^3\,\theta\,K_2\left(\frac{1}{\theta}\right)}\exp\left(-\frac{\gamma(\mathbf{p})}{\theta}\right), \tag{12}$$

where $\gamma(\mathbf{p})$ is the Lorentz factor, and K_2 is the modified Bessel function of the second kind and $\theta=k_BT/(mc^2)$. Note that this standard normalization also leads to the same type of normalization for the parameter ε, defined subsequently based on the definition of the parameter d given in Eq. (30).

To solve the Vlasov equation, a time-splitting method is used. The idea is to separate Eq. (1) into many simpler equations that can be solved by doing advections sequentially in each direction. In relativity, due to the Lorentz factor γ, we cannot do this separation for the momentum components[43] so all the momentum advections are done together leading to what we will call the rotation.

B. The numerical scheme

Although the overall integration scheme of the Vlasov–Maxwell system has been presented in Ref. 4, it is useful to outline the main steps involved in solving the VLEM code. A time-splitting technique is introduced, allowing the Vlasov equation (1) to be decomposed into two primary advection processes

$$\frac{\partial f}{\partial t}+\frac{\mathbf{p}}{m\gamma}\cdot\nabla_{\mathbf{x}}f=0, \tag{13}$$

$$\frac{\partial f}{\partial t}+e\left(\mathbf{E}+\frac{\mathbf{p}\times\mathbf{B}}{m\gamma}\right)\cdot\nabla_{\mathbf{p}}f=0. \tag{14}$$

Since Eq. (13) can be easily expressed in a conservative form given by $\frac{\partial f}{\partial t}+\partial_x\left(\frac{p_x}{m\gamma}f\right)+\partial_y\left(\frac{p_y}{m\gamma}f\right)=0$, where the Lorentz factor γ is independent of the spatial variables, an additional splitting along the x and y directions can also be performed. This allows the establishment of one-dimensional spatial advections, a property that is utilized in the VLEM code to achieve local interpolations.

The overall temporal integration scheme is illustrated in Fig. 1, utilizing a time-splitting technique with a global sequence $\frac{\mathcal{Y}}{2}\frac{\mathcal{X}}{2}\mathcal{R}\frac{\mathcal{X}}{2}\frac{\mathcal{Y}}{2}$ to

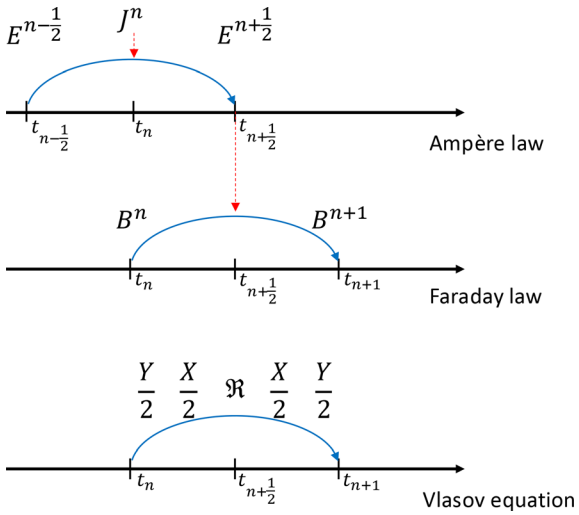

FIG. 1. Numerical scheme used in full kinetic and AMR VLEM codes: the top frame correponds to the computation of the electric field component $\mathbf{E}^{n+\frac{1}{2}}$ at time $t_{n+\frac{1}{2}} = \left(n+\frac{1}{2}\right)\Delta t$, while the middle frame is related to the computation of the magnetic field $\mathbf{B}^{n+1}$, at time $t_{n+1} = (n+1)\Delta t$, using a temporal leapfrog scheme. At the end of this sequence, the quantity $\mathbf{B}^{n+\frac{1}{2}}$ is computed using $\mathbf{B}^{n+\frac{1}{2}} = (\mathbf{B}^n + \mathbf{B}^{n+1})/2$, used in the rotation. The electric field is then corrected before each rotation using the Poisson equation to ensure charge conservation.

determine the distribution function $f^{n+1} = f(\mathbf{x}, \mathbf{p}, t = t_{n+1})$. This indicates a symmetrized sequence of five successive advections: one-dimensional local advections in the spatial directions y and x (represented by the operators $\frac{\mathscr{Y}}{2}$ and $\frac{\mathscr{X}}{2}$, where the factor of 1/2 denotes advections performed over a half time step), and a three-dimensional (or two-dimensional, depending on the dimensionality of the system) advection in momentum space denoted by $\mathscr{R}$ (for the global rotation). Furthermore, the symmetrization of the various advections enables the achievement of a global scheme that is second-order in time step (see Refs. 4 and 44 for more details). Without symetrization, the splitting time scheme would be of first order accuracy only. This leads to apply the successive sequences of shifts and rotation in the form $(\frac{\mathscr{Y}}{2}\frac{\mathscr{X}}{2}\mathscr{R}\frac{\mathscr{X}}{2}\frac{\mathscr{Y}}{2})^n$ which is equivalent to the following sequence:

$$\frac{\mathscr{Y}}{2}\frac{\mathscr{X}}{2}\left(\mathscr{R}\frac{\mathscr{X}}{2}\mathscr{Y}\frac{\mathscr{X}}{2}\right)^{n-1}\mathscr{R}\frac{\mathscr{X}}{2}\frac{\mathscr{Y}}{2}.$$

Except the edges of the sequence, the main steps for the resolution may be summarized as

$$\text{rotation } \mathscr{R}: \quad f^*(x,y,\mathbf{p}) = f^n(x,y,\mathbf{P}(x,y,\mathbf{p})), \tag{15}$$

$$\text{shift } \mathscr{X}/2: \quad f^{**}(x,y,\mathbf{p}) = f^*\left(x - \frac{p_x}{m\gamma}\frac{\Delta t}{2}, y, \mathbf{p}\right), \tag{16}$$

$$\text{shift } \mathscr{Y}: \quad f^{***}(x,y,\mathbf{p}) = f^{**}\left(x, y - \frac{p_y}{m\gamma}\Delta t, \mathbf{p}\right), \tag{17}$$

$$\text{shift } \mathscr{X}/2: \quad f^{n+1}(x,y,\mathbf{p}) = f^{***}\left(x - \frac{p_x}{m\gamma}\frac{\Delta t}{2}, y, \mathbf{p}\right), \tag{18}$$

with f^n the distribution function at the n^{th} iteration.

Numerically, to solve Eqs. (15)–(18), we will use a semi-Lagrangian method. The distribution function is defined on grid points using an Eulerian grid [i.e., $f(i_x, i_y, j_x, j_y)$ for two dimensions in momentum space, where we use i_x, i_y, j_x, and j_y as indices for x, y, p_x, and p_y respectively], with an interpolation method to compute its value after the advection. Spatial advections and those in momentum space are not performed in the same way, primarily due to the constraints related to parallelism, i.e., the domain decomposition that is carried out in the position space.

The standard semi-Lagrangian technique is carried out in two successive steps: "particles" are generally positioned at the grid points of phase space at the time corresponding to the end of the advection, so it is necessary to perform a backward characteristic search to determine the required shifts. In a second step, an interpolation is then needed to determine the values of the distribution function. This is a global method that typically requires the inversion of a tridiagonal matrix. It is used for the determination of the distribution function during rotation in momentum space, which corresponds Eq. (15). This method employs a 3D (or 2D) characteristic, and the interpolation is done using a tensor of three (or two) one-dimensional B-splines in momentum quantities.

The spatial advections corresponding to Eqs. (16)–(18) are performed using a local technique. Due to the domain decomposition introduced here, the inversion of the tridiagonal matrix is abandoned in favor of a local spline interpolation method, based on a modification of the boundary conditions.

This local approach (i.e., local spline interpolation[4,45]) is essential due to the parallelization of VLEM. Indeed, VLEM is decomposed into rectangular patches over the spatial domain. This means that each patch is computed using different MPI (message passing interface) processes. The local advections lead to reduce considerably the MPI communications only on the two neighboring processes for a given spatial direction x or y. Due to the relativistic nature of the various shifts in the position space (the components of the velocity vector $\mathbf{p}/m\gamma(\mathbf{p})$ in shifts $\mathscr{X}$ and $\mathscr{Y}$ being smaller than the speed of light) the advected quantities in shifts $\mathscr{X}$ and $\mathscr{Y}$ will respect the condition $|\frac{p_x}{m\gamma}|\Delta t \leq c\Delta t$. As the CFL condition [Eq. (27)] ensures that $c\Delta t \leq \Delta x$, the interpolation method for each MPI process can be restricted to the closest points of the first neighbor process in the positive and negative direction (i.e., $|\frac{p_x}{m\gamma}|\Delta t \leq \Delta x$, for example).

Those local advections are done using cubic spline interpolation with Hermite boundary conditions that only need the ten closest points of the value of the distribution function of the two neighboring cells,[45] i.e., ten points in the positive direction and ten points in the negative direction.

When the electric field is zero, the characteristics of Eq. (14) can be exactly determined since the particle rotates around a magnetic field line at constant velocity, at the cyclotron frequency $\omega_c = e|\mathbf{B}|/m\gamma$ (the Lorentz factor γ being conserved). This equation is solved very efficiently by decoupling the electric and magnetic fields in the standard way. This is done with the Boris method used in PIC model.

However, there is a difference between the Lagrangian and Eulerian approaches. In the Lagrangian approach, the Boris formulation is not Lorentz invariant and can lead to significant errors, necessitating a correction in the treatment of the leapfrog scheme used in PIC codes.[46] Since it is possible to decouple the electric and magnetic fields in Eq. (14), leading to two 3D advections of the form $\frac{\partial f}{\partial t} + e\mathbf{E}\cdot\mathbf{V}_{\mathbf{p}}f = 0$ and $\frac{\partial f}{\partial t} + e\frac{\mathbf{p}\times\mathbf{B}}{m\gamma}\cdot\mathbf{V}_{\mathbf{p}}f = 0$, this decoupling can be achieved, provided that a 3D advection is maintained when the Lorentz force is taken into

account. In particular, in relativity due to the Lorentz factor, it is not possible to further divide the 3D advection that includes the Lorentz force into three one-dimensional advections (see Ref. 43).

The rotation is made at Eq. (15) and can be developed as

$$P_x = -\frac{e\Delta t E_x}{2} + p_x^{\delta} R_{xx} + p_y^{\delta} R_{xy} + p_z^{\delta} R_{xz}, \tag{19}$$

$$P_y = -\frac{e\Delta t E_y}{2} + p_x^{\delta} R_{yx} + p_y^{\delta} R_{yy} + p_z^{\delta} R_{yz}, \tag{20}$$

$$P_z = -\frac{e\Delta t E_z}{2} + p_x^{\delta} R_{zx} + p_y^{\delta} R_{zy} + p_z^{\delta} R_{zz}, \tag{21}$$

where

$$\mathbf{p}^{\boldsymbol{\delta}} = \mathbf{p} - \frac{e\Delta t \mathbf{E}}{2}, \tag{22}$$

and with $\boldsymbol{R}$ the rotation matrix[4]

$$\boldsymbol{R} = \begin{pmatrix} b_x^2 + (1 - b_x^2)\cos\phi & b_x b_y (1 - \cos\phi) - b_z \sin\phi & b_x b_z (1 - \cos\phi) + b_y \sin\phi \\ b_x b_y (1 - \cos\phi) + b_z \sin\phi & b_y^2 + (1 - b_y^2)\cos\phi & b_y b_z (1 - \cos\phi) - b_x \sin\phi \\ b_x b_z (1 - \cos\phi) - b_y \sin\phi & b_y b_z (1 - \cos\phi) + b_x \sin\phi & b_z^2 + (1 - b_z^2)\cos\phi \end{pmatrix}, \tag{23}$$

with $b_\alpha = B_\alpha/|\mathbf{B}|$ and $\phi = \omega_c \Delta t$, with ω_c the cyclotron frequency. This matrix is only valid for VLEM2D3V. For VLEM2D2V we should take $b_x = b_y = 0$ and $b_z = 1$ because this version fixes $B_x = B_y = E_z = 0$ as they are related to J_z which needs p_z to be computed. Matrix (23) is used to accurately determine the characteristics during the rotation of the particle around the magnetic field.[47] $\boldsymbol{P}$ represents the solution obtained from the characteristics of the Vlasov equation in momentum space.

To solve Eq. (15), we need a three-dimensional (or two-dimensional) interpolation method to make the advection related to the rotation for VLEM2D3V (or VLEM2D2V) [Eqs. (19)–(23)]. The cubic B-spline method, in three (or two dimensions) is used. It was already described in Ref. 21 for a two dimensional case but the extension to three dimension is trivial. For each momentum dimension, it solves a system of linear equations to find the spline coefficients. This limits the dimension of the distribution function to a rectangle as the number of points for p_α should be constant and not depend on the other p_β coordinate. Moreover, the step Δp_α should be constant for a given direction.

Concerning the Maxwell equations, we here limit the presentation to a linearly polarized electromagnetic case with E_x, E_y, and B_z components. In this algorithm, the continuous derivatives in space and time are approximated by second-order accurate, two-point centered difference forms. A leapfrog integration in time is used to update the fields. The determination of the electric and magnetic components of the electromagnetic field ($\boldsymbol{E}$, $\boldsymbol{B}$) is carried out by solving Eqs. (7) and (6), respectively. A leapfrog scheme in time is used to update the fields, provided that initially the electric field satisfies Maxwell–Gauss's equation (8). The fact that the initial electric field corresponds to the electrostatic component is sufficient to achieve an accurate determination of the overall magnetic field.[48]

The E_x component is located at half x and integer y grid points, i.e., $\left(i_x + \frac{1}{2}, i_y\right)$ while the E_y component is located at $\left(i_x, i_y + \frac{1}{2}\right)$. The magnetic field component B_z is located at $\left(i_x + \frac{1}{2}, i_y + \frac{1}{2}\right)$. In general, in 3D, the electric field components are at the center of the edge of a cell, while the magnetic field components are at the center of a face. These grid points are chosen to accommodate the interleaved leapfrog algorithm that makes up the Yee method. The discretized and unnormalized versions are

$$E^{n+\frac{1}{2}}_{x\, i_x+\frac{1}{2}, i_y} = E^{n-\frac{1}{2}}_{x\, i_x+\frac{1}{2}, i_y} + \frac{c^2 \Delta t}{\Delta y}\left(B^n_{z\, i_x+\frac{1}{2}, i_y+\frac{1}{2}} - B^n_{z\, i_x+\frac{1}{2}, i_y-\frac{1}{2}}\right) - \frac{\Delta t}{\varepsilon_0} J^n_{x\, i_x+\frac{1}{2}, i_y}, \tag{24}$$

$$E^{n+\frac{1}{2}}_{y\, i_x, i_y+\frac{1}{2}} = E^{n-\frac{1}{2}}_{y\, i_x, i_y+\frac{1}{2}} - \frac{c^2 \Delta t}{\Delta x}\left(B^n_{z\, i_x+\frac{1}{2}, i_y+\frac{1}{2}} - B^n_{z\, i_x-\frac{1}{2}, i_y+\frac{1}{2}}\right) - \frac{\Delta t}{\varepsilon_0} J^n_{y\, i_x, i_y+\frac{1}{2}}, \tag{25}$$

$$B^{n+1}_{z\, i_x+\frac{1}{2}, i_y+\frac{1}{2}} = B^n_{z\, i_x+\frac{1}{2}, i_y+\frac{1}{2}} + \frac{\Delta t}{\Delta y}\left(E^{n+\frac{1}{2}}_{x\, i_x+\frac{1}{2}, i_y+1} - E^{n+\frac{1}{2}}_{x\, i_x+\frac{1}{2}, i_y}\right) - \frac{\Delta t}{\Delta x}\left(E^{n+\frac{1}{2}}_{y\, i_x+1, i_y+\frac{1}{2}} - E^{n+\frac{1}{2}}_{y\, i_x, i_y+\frac{1}{2}}\right). \tag{26}$$

The numerical scheme uses a CFL condition,[4] related for solving the Maxwell equations, such that

$$c^2 \Delta t^2 (\Delta x^{-2} + \Delta y^{-2}) \leq 1. \tag{27}$$

From a numerical point of view, the Gauss equation [Eq. (8)] is rarely verified in its discrete form, which can lead to poor conservation of the global charge. To overcome this problem, the Poisson equation is solved, at each time iteration, at the end of the sequence of solving Maxwell's equations (see Refs. 4 and 48–51) So, in addition to the Yee algorithm, a correction is made to the electric field using Eq. (8), by calculating the error such that

$$\varepsilon_E = \mathrm{div}\boldsymbol{E} - \frac{\rho}{\varepsilon_0}. \tag{28}$$

We compute the electric potential associated with this error with $\nabla^2 \varphi = \varepsilon_E$ using Fast Fourier Transform and Inverse Fast Fourier Transform, we can then correct the electric field with

$$\boldsymbol{E}_{corrige} = \boldsymbol{E} - (-\boldsymbol{\nabla}\boldsymbol{\varphi}). \tag{29}$$

This correction of the electric field using the Poisson equation helps to improve the charge conservation of the code.

The temporal evolution of the Yee scheme and the sequence of advections are schematically shown in Fig. 1. We can also summarize

in a simple way the global algorithm for the compact sequence $((\mathscr{R})(\mathscr{X}/2)(\mathscr{Y})(\mathscr{X}/2))$ by removing the part in ∗∗ in Algorithm 1.

III. EMBEDDED GRID

A. Refinement strategy

As we have seen in the introduction (Sec. I), the AMR techniques prove to be particularly useful in the momentum space, especially with respect to the computational gain that they can bring to a standard semi-lagrangian Vlasov solver with respect to a PIC solver with equivalent space resolution and fixed number of macro-particles. Moreover, it is in the **p**-space that the phenomenon of kinematic filamentation of f occurs and this constitutes the strongest constraint, since it requires the use of a high sampling in the **p**-space to describe the filamentation process. The filamentation of the distribution function is not a numerical artifact and can be forced or even amplified by certain physical instabilities or during reconnection processes. These lead to the necessity of adding mesh refinement methods to VLEM but the complexity of the latter leads to the choice of a specific type of mesh refinement: the embedded grid.

The technique subdivides cells that not only satisfy the required refinement criterion (here associated with the filamentation description) but also the cells that complete the rectangle form of the grid, and enhances the local resolution in the momentum space. If a certain cell needs to be refined, 2^{d_p} child cells with half the size of the parent cell are generated, where d_p is the dimension of the momentum space.

First, VLEM is a fully parallelized code that uses a domain decomposition in space allowing to perform the different advections in a parallel way using MPI processes together with open-MP parallelization. Therefore, the choice of a mesh refinement in the momentum space maintains this parallelization without changing the architecture of VLEM. Indeed, a mesh in space would require more MPI processes and communications betweeen neighboring processes and optimizing such a code is really challenging. Moreover, if you only want to increase the accuracy in space, i.e., the number of points, you can simply increase the number of processors you need. The computational time will remain the same thanks to the good weak scalling in VLEM,[4] but the computational cost will increase.

While the configuration space uses a domain decomposition, allowing MPI-type parallelization in the spatial directions (corresponding to the shifts in x and y), the OpenMP parallelization is used at local level for the AMR technique. This allows simple use of the method since no AMR technique is used for solving Maxwell's equations, thus avoiding any numerical instability of Yee's method. Indeed, it is well known that the propagation of electromagnetic waves at the connection zones between the different refinement grids can produce numerical instabilities.[35,36]

Another criterion that we have to take into account when implementing the mesh refinement is the rotation ($\mathscr{R}$). Indeed, Eq. (15) is solved by using cubic spline method, which requires to have the information for the distribution function at a given **x** in a third (or second) order tensor that can be compared to a kind of rectangular cuboid (or a rectangle). That is why we chose the term "rectangle" embedded grid. The term has been retained for the 2D3V version but we are talking about a rectangular parallelepiped. This means that the distribution function can be seen as a set of embedded rectangles, one per level of refinement, with the number of points per momentum volume increasing as you move to more internal rectangles.

The AMR technique benefits from a reduction of the total number of cells and computer resources associated with the introduction of finer cells hierarchically placed only in regions, of the momentum space, where particle trapping, filamentation or wave-particle interaction take place, and where a higher resolution is required.

Finally, the spatial advections Eqs. (16)–(18) also introduce a constraint on the refinement. Indeed, these advections are made using local spline interpolations,[4] which means that we need to have the information of the distribution function at a given p_i for all the x_i points of the patch and the closest ten points of the two neighboring patches. This means that the "rectangle" of a given point in (i_x, i_y) should be the same for all the values of (i_x, i_y), i.e., for the whole spatial grid.

In this way, we end up with the idea of a "rectangle" embedded grid in the momentum space, with the same grid over the whole space.

B. Method

In this section, for the sake of simplicity but with no loss of generality, all the explanations and figures will refer to an implementation on VLEM2D2V.

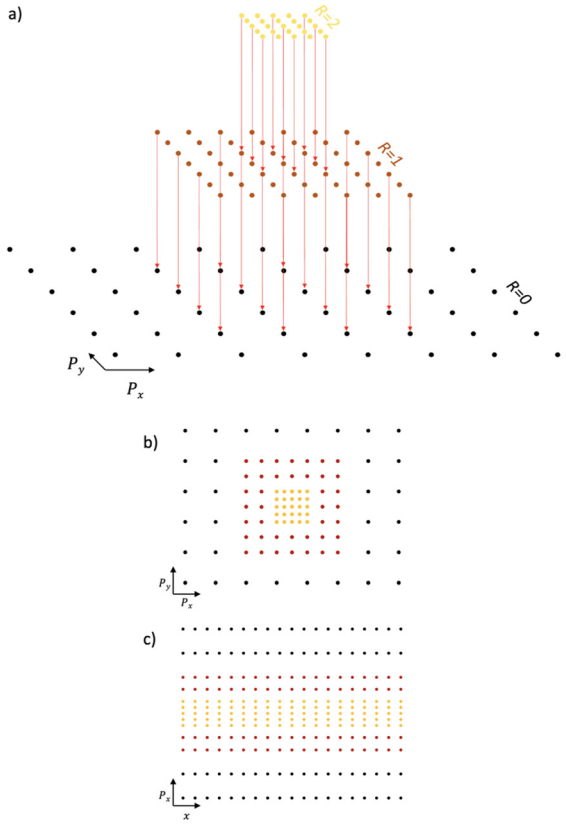

FIG. 2. Representation of the embedded grid mesh refinement. (a) View of the communication between the different grids at a given point in (i_x, i_y). The red arrows show the downhill of information between points located at the same place in phase space. (b) View of the combined distribution function in p_x/p_y for a given point in (i_x, i_y). (c) View of the combined distribution function in x/p_x for a given point in $(i_y, j_y = N_{py}/2)$.

As we speak about embedded grid, which means that the distribution function will be described on different levels defined by their refinement level R. The basic level is $R = 0$ and it will be fixed in dimension for the boundary from $p_{x_{min}}(0)/p_{y_{min}}(0)$ to $p_{x_{max}}(0)/p_{y_{max}}(0)$ which will be the dimension of our system. Then, other levels will have their own $p_{x_{min}}(R)$, $p_{y_{min}}(R)$, $p_{x_{max}}(R)$, $p_{y_{max}}(R)$ which will change during the iterative process and which can go up to the limit of $R = 0$. We will call R^{max} the maximum level of refinement allowed, which can be adapted depending on the simulation, and R_n^{max} the level of refinement at the time $t = n\Delta t$. So, we will have $R_n^{max}+1$ rectangular grids discretising the distribution function. Each of them is independent and will only communicate at some specific points (Fig. 2).

The refinement ratio between two grids is two. So, for VLEM2D2V, a grid of level $R = m+1$ will have a density of points in momentum space four times greater than the grid $R = m$ and for VLEM2D3V it will be eight times greater. Figure 2 shows how each grid is located with respect to each other. The arrows show the points that are defined in both grids, located on the same phase space and that will communicate after each advection with a downhill of information (see Sec. III E). We can also comment the fact that the border of a grid of level $R+1$ should end on point described by the level R. This will be justified in Subsec. III E.

Recall that we use i_x, i_y, j_x, and j_y as indices, respectively, for x, y, p_x, and p_y. The point located at (i_x, i_y, j_x, j_y) of the grid R describes the same position as $(i_x, i_y, 2j_x, 2j_y)$ of the grid $R+1$. The points that are specific of $R+1$ and undefined on R will be the points that contain one or more odd values for $j_\alpha(R+1)$.

The modified version of the code VLEM containing the embedded grid mesh refinement will be called VLEM_EG.

C. Initialization of the grid

Before starting the time evolution of the problem, we need to make the right initialization. The main steps to refine the initial mesh are given as follows.

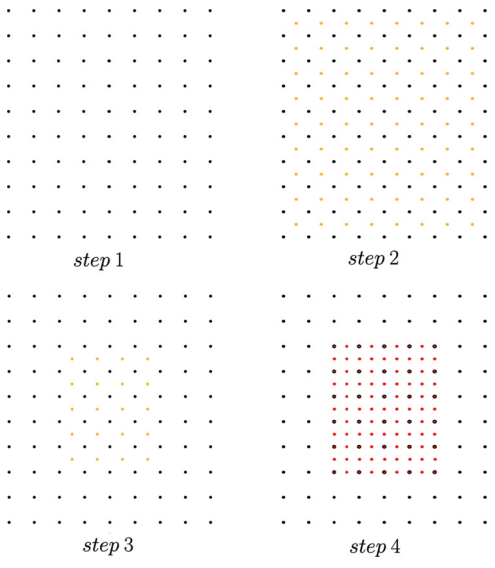

FIG. 3. Representation of the process of initializing the grid in p_x/p_y for all the points in space. The steps are indicated and they are explained in Subsec. III C.

- *Step* 1: At first the base grid ($R = 0$) is defined in the classical way with the distribution function adapted to the problem (Fig. 3).
- *Step* 2: Then, to create and define the new grids, we use the cubic spline interpolation to obtain the values of the distribution function at $(j_x+\frac{1}{2}, j_y+\frac{1}{2})$. We choose those points for the interpolation because they should provide the worst interpolation results compared to any other location.
- *Step* 3: The values of this interpolation will be compared with the real values of the distribution function known at those points

$$d = \left| f_{\text{interpolated}}\left(i_x, i_y, j_x+\frac{1}{2}, j_y+\frac{1}{2}\right) - f_{\text{real}}\left(i_x, i_y, j_x+\frac{1}{2}, j_y+\frac{1}{2}\right)\right|. \tag{30}$$

If we have

$$d > \varepsilon 2^{-(R+1)}, \tag{31}$$

then the grid needs to be refined on that point, otherwise it is not necessary.The parameter d introduced above allows for a measurement of the level of variation of the distribution function. Thus, the parameter d, along with the coefficient ε used to estimate the level of refinement, are two quantities with similar normalizations to that of f, that is, with a normalization of $(mc)^{d_p}/n_0$ where d_p is the dimension of momentum space. In the case of 3D, this normalization is m^3c^3/n_0. This refinement criterion is inspired from Ref. 28 and is chosen to reduce the L^∞ error on the first momentum derivative. In this way the parameter ε, which is normalized, is the maximum of this error. In the case of only two refinement level (i.e., $R^{max} = 1$), it is also directly the maximal L^∞ error wanted on the distribution function.
- *Step* 4: This difference [Eq. (30)] will be computed for each point (i_x, i_y) and we will keep the minimum and maximum value of each j_α. If there is no point where the creation condition is reached no grid will be created because the interpolation is already good enough to describe the problem. If it is not the case, we create the first refined grid $R = 1$ and we complete the value of its distribution function with the initial value of the distribution. We also fixed the condition that the edge of a grid should end on an even value of j_α (Fig. 3).

We then restart this process with the interpolation of the grid $R = m$ to know if we create the grid $R = m + 1$ until we reach either $m + 1 = R^{max}$ (the maximum level of refinement we wanted) or if a grid is not created. The last grid created will fix the initial value of R_n^{max}. Note that R^{max} will not change over time as it is an input parameter fixing the maximum level of refinement where R_n^{max} can change from 0 to R^{max} depending on the current refinement.

When all the grids are created the code can normalize the mean density n to n_0 and start the time iterative process.

D. Grid evolution in time

The time evolution process is a sequence $((\mathscr{R})(\mathscr{X}/2)(\mathscr{Y})(\mathscr{X}/2))^{n_{max}}$ and every κ iterations of this sequence, we will check if the mesh is still adapted to our problem. We choose $\kappa = 5$ but this one can be adapted to the simulation. This value is a good compromise between a good grid evolution and low computational impact.

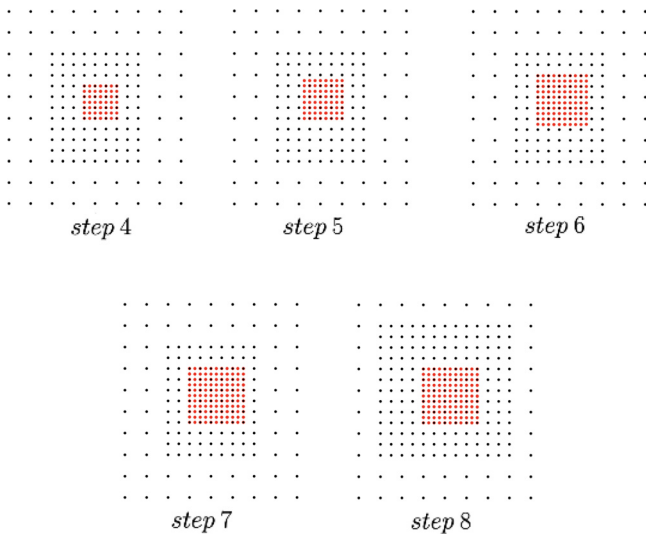

FIG. 4. Representation of the iterative refinement of the grid in p_x/p_y for all the points in space. The steps are indicated and explained in Subsec. III D. Step 4 is the same as in Fig. 3 and Subsec. III C.

In the future, we can suppose a value that will change depending on the current evolution of the process, mostly when there will be a strong evolution of the field or a huge acceleration of particles.

This process will work in a close way as the initialization one (Fig. 3). That means that we will use the grid R to interpolate the value at the point $(j_x + \frac{1}{2}, j_y + \frac{1}{2})$ and we will compare it with the value of the point located at $(2j_x + 1, 2j_y + 1)$ for the grid R+1. This last point will be f_{real} in Eq. (30) (Fig. 4, step 4). We recall all the steps and we start with $R = 0$.

- *Step* 1: We got our distribution function defined on the level R.
- *Step* 2: We use the cubic spline interpolation to obtain the values of this distribution function at $(j_x + \frac{1}{2}, j_y + \frac{1}{2})$.
- *Step* 3: We compute d [Eq. (30)] with the values of this interpolation ($f_{interpolated}$) and with the already existing values of the distribution function on the level R+1 (f_{real}). Again we have the condition (31) to refine the grid on that point. We add another condition for d to make the possibility to create a new level of refinement if $R_n^{max} \neq R^{max}$. So, if $d > \varepsilon 2^{-R}$, that means that this point also needs to be refined on the level R+2. The value of ε is here the same as previously.
- *Step* 4: We will keep the minimum and maximum value of each j_α. If there is no point where the creation condition is reached, no grid will be created and if the grid R+1 already existed, it is destroyed.
- *Step* 5: If at least one point needs to be refined we will anticipate where it will be located in κ time steps. This means that we will do a forward advection. As we have the same grid on each point (i_x, i_y), it will be useless to do the spatial advections. So, we will only look at the forward rotation such as

$$P_\alpha = \frac{e\Delta T E_\alpha}{2} + p_x^{\delta'} R_{\alpha x}^{-1} + p_y^{\delta'} R_{\alpha y}^{-1} + p_z^{\delta'} R_{\alpha z}^{-1}, \tag{32}$$

with $\Delta T = \kappa * \Delta t$ and where $\boldsymbol{R^{-1}}$, the inverse of $\boldsymbol{R}$, is

$$\boldsymbol{R^{-1}} = \begin{pmatrix} b_x^2 + (1 - b_x^2)\cos\phi & b_x b_y(1 - \cos\phi) + b_z \sin\phi & b_x b_z(1 - \cos\phi) - b_y \sin\phi \\ b_x b_y(1 - \cos\phi) - b_z \sin\phi & b_y^2 + (1 - b_y^2)\cos\phi & b_y b_z(1 - \cos\phi) + b_x \sin\phi \\ b_x b_z(1 - \cos\phi) + b_y \sin\phi & b_y b_z(1 - \cos\phi) - b_x \sin\phi & b_z^2 + (1 - b_z^2)\cos\phi \end{pmatrix}, \tag{33}$$

where $p_\alpha^{\delta'} = p_\alpha + \frac{e\Delta T E_\alpha}{2}$ and $\phi = \omega_c \Delta T$. With Eq. (32) we can now take the minimum or maximum value between p_α and P_α to have the biggest rectangle. In that way, we are now sure that the refined grid will be well defined for all κ iterations, before the next recalculation of each grid.

- *Step* 6: We will increase the size of the rectangle from one point in each direction to create "ghost cells." Their utility will be explained in Subsec. III E.
- *Step* 7: Again, we force the boundaries of the rectangle to be on even values of j_α.

We repeat steps 1–7 until either we reach $R = R^{max}$ or until no more grids need to be created and we proceed to step 8:

- *Step* 8: For this mesh refinement method, we need a "graded grid". This means that between two meshes with two levels of difference (R and R+2) there should be a certain number of points of the R+1 level mesh. The number of points is not only chosen to be three to reduce the numerical propagation of the error in the distribution function but also to be consistent with the integration method explained in Subsec. III F. So, we check whether this condition is already respected by the previous steps, either we will increase the size of the grid of R+1 level to match this condition. This condition is necessarily met when the grid is at the edges of the momentum box, since the value of the distribution function outside the box is fixed at 0. Note that at the end of this process, the "ghost cells" of each grid will be located at an even value of (j_x, j_y) and will still be at the edge of the grid.

All the steps are summarized in Fig. 4.

At the end of all those steps, there are now three scenarios for a point of a grid of level R+1:

- The point does not exist anymore, in that case its value is destroyed.
- The point is still there, here the value will stay the same.
- The point needs to be created. In this case we will do a four-points Lagrange interpolation with R to determine the new value of the point. This is one of the reasons why we need a graded grid. The Lagrange interpolation can also be replaced by the cubic

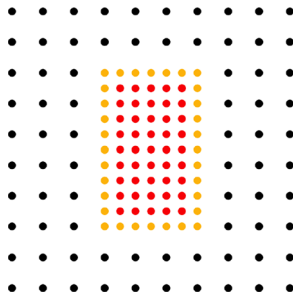

FIG. 5. Representation of ghost cells between two grids in the plan (p_x, p_y). In orange are represented the ghost cells that are always located on even points, in red the other points of level $R+1$ and in black the points of level R.

spline interpolation depending on the precision or speed we want for the algorithm.

E. Modification of the global numerical scheme

The time splitting scheme remains the same $((\mathscr{R})(\mathscr{X}/2)(\mathscr{Y})(\mathscr{X}/2))$ sequence as in VLEM.

First the rotation will happen and will be done in a separate way on each grid. To work with the cubic spline method and the boundary conditions. We will use the "ghost cells" only for the interpolation, but their values will be determined at the level below. Here, is the plan for the new rotation $\mathscr{R}$.

- Grid $R{=}R_n^{max}$ computes the rotation and replaces its distribution function except for "ghost cells".
- Grid $R{=}R_n^{max} - N > 0$ computes the rotation and replaces its distribution function except for "ghost cells" and for the points that have already been computed by $R_n^{max} - N{+}1$. It also uses the cubic spline interpolation to determine the values of the ghost cells on the level $R_n^{max} - N{+}1$.
- Grid $R = 0$ computes the rotation and replaces its distribution function by the interpolated values except for the edge that are fixed to 0 and for the points that have already been computed by $R = 1$. We also determine the values of $R = 1$ ghost cells.

Note that the reason we put the "ghost cells" on even value of j_α is to make less errors on the computation of their values.

After that, there is a downhill information sharing from the points that are at the same places as the ones from the grid bellow. This sharing happens after each advection. It is represented by the red arrow in Fig. 2(a). This closes the calculation for the rotation.

Then, we do the first $(\mathscr{X}/2)$ advection. In VLEM the local spline is computed for each p_x but we will do the advection for each grid only at the points where the grid level is the most refined. This means that the level R will compute this advection only on the black points in Fig. 5 and not on the inner red and orange points which are better defined by grid $R{+}1$. The downhill of information, which is carried out after each advection, replaces the values that are not computed by the level R.

The same process is done for $(\mathscr{Y})$ and the other $(\mathscr{X}/2)$. We summarized in a simplified way the new version of the iterative algorithm in Algorithm 1.

ALGORITHM 1. Simplified algorithm of VLEM_EG, applied to the 2D2V case, where $**$ represents the terms that are specific to VLEM_EG and do not exist in VLEM. f_R is the distribution function defined on the level R and n_{max} is the maximum number of iterations.

$*$Initial Refinement with Steps 1–4$*$
While $n \leq n_{max}$ **do**
  Solving Eq. (6) $(B_z(t_{n-1/2}) \rightarrow B_z(t_{n+1/2}))$
  Computing $E_x^*(t_n)$, $E_y^*(t_n)$, $B_z^*(t_n)$
  If $mod(n, 5) = 1$ **then**
    $*$Iterative Refinement with Steps 1–8$*$
  end
  Rotation $\mathscr{R}$
  $*$Downhill $(f_R(x, y, j_x, j_y) = f_{R+1}(x, y, 2j_x, 2j_y))*$
  Local advection $\mathscr{X}/2$
  $*$Downhill $(f_R(x, y, j_x, j_y) = f_{R+1}(x, y, 2j_x, 2j_y))*$
Local advection $\mathscr{Y}$
  $*$Downhill $(f_R(x, y, j_x, j_y) = f_{R+1}(x, y, 2j_x, 2j_y))*$
  Local advection $\mathscr{X}/2$
  $*$Downhill $(f_R(x, y, j_x, j_y) = f_{R+1}(x, y, 2j_x, 2j_y))*$
  Solving Poisson's equation (correction of E_x and E_y)
  Solving Eq. (7) $\begin{pmatrix} E_x(t_n) \rightarrow E_x(t_{n+1}) \\ E_y(t_n) \rightarrow E_y(t_{n+1}) \end{pmatrix}$
  $n = n + 1$
end

F. Electromagnetic field

The electric and the magnetic fields are not directly impacted by the mesh refinement as they are defined only in the space variables. Their time evolutions are computed using Eqs. (6) and (7) and the standard Yee method.[4]

Equation (6) shows that the time evolution of the magnetic field will only depend on the electric field but the evolution of the electric field will depend on the magnetic field and the current density [Eq. (7)]. The current density will be computed with the normalized version of Eq. (9). In VLEM, the integral is just a summation that can be written for the specific 2D2V case

$$J_\alpha^{VLEM}(x, y) = \int \frac{p_\alpha}{\gamma(\mathbf{p})} f(x, y, \mathbf{p}) \mathbf{dp} = \sum_{j_x=j_{x_{min}}}^{j_{x_{max}}} \sum_{j_y=j_{y_{min}}}^{j_{y_{max}}} \frac{p_\alpha}{\gamma(j_x, j_y)} f(x, y, j_x, j_y) \Delta p, \quad (34)$$

with $\Delta p = \Delta p_x \Delta p_y$. We can generalize this to all the integrals $I(x, y)$ in the momentum space by writting

TABLE II. Value of $16 \times C(j_x, j_y)$ at a border. The points that are not represented and are not affected by other grid interfaces have a value of $C(j_x, j_y) = 1$.

n − 2	n − 1	n	n + 1	n + 2	
14	16	32		16.5	2M
16	16	16			2M + 1

TABLE III. Value of $256 \times C(j_x, j_y)$ at a corner. The points that are not represented and are not affected by other grid interfaces have a value of $C(j_x, j_y) = 1$.

n − 2	n − 1	n	n + 1	n + 2	
264.25		262		255.75	m + 2
					m + 1
488	256	704		262	m
256	256	256			m − 1
191	256	488		264.25	m − 2

$$I(x,y) = \sum_{j_x=j_{x_{min}}}^{j_{x_{max}}} \sum_{j_y=j_{y_{min}}}^{j_{y_{max}}} F(x,y,j_x,j_y)\Delta p, \tag{35}$$

with F as the function we want to integrate.

In VLEM_EG this summation cannot be done because we do not have a regular step between each point. We assume a homogeneous grid at the level R_n^{max} [i.e., with $\Delta p(R_n^{max})$] defined everywhere on the momentum box. Since it is a homogeneous grid, we can consider the integral as a summation of all terms, as in VLEM [Eq. (35)]. We redistribute the values of the missing points, those that are not defined on the grid R_n^{max}, by using a 4-point interpolation method that will modify the weight of the points that are defined on the level $R_n^{max} - 1$, such that

$$\begin{aligned} F_{R+1}(2j_x{+}1, 2j_y) = \frac{1}{16}\big[&-F_R(j_x - 1, j_y) + 9F_R(j_x, j_y) \\ &+ 9F_R(j_x{+}1, j_y) - F_R(j_x{+}2, j_y)\big], \end{aligned} \tag{36}$$

$$\begin{aligned} F_{R+1}(2j_x{+}1, 2j_y{+}1) = \frac{1}{16}\big[&-F_{R+1}(2j_x{+}1, 2j_y - 2) \\ &+ 9F_{R+1}(2j_x{+}1, j_y) + 9F_{R+1}(2j_x{+}1, 2j_y{+}2) \\ &- F_{R+1}(2j_x{+}1, 2j_y{+}4)\big], \end{aligned} \tag{37}$$

where $F_R(j_x, j_y) = F_R(i_x, i_y, j_x, j_y)$ is the function we want to integrate, which depends on the distribution function defined on the level R. We recall that the point located at (x, y, j_x, j_y) of the grid R describes the same position as $(x, y, 2j_x, 2j_y)$ of the grid $R{+}1$. So, only the point that does not exist on the grid $R{+}1$ and that contains one or more odd values of j_α will be redistributed along the points of the grid R.

This process is then repeated (if possible) for the missing points of level $R_n^{max} - 1$ to redistribute their values on level $R_n^{max} - 2$. This stops when the level $R = 0$ is reached, which is defined on the whole momentum box.

This will give more weight to the point where the grid is less refined, that is why we will reintroduce $\Delta p(R)$. Also, the points located close to the edges of a refined grid will have more weight than those inside of the grid. Therefore, we introduce another factor C in Eq. (38) that will differentiate the weights between the points of the same region. We can now rewrite the general integral in VLEM_EG such that

$$\begin{aligned} I(x,y) = &\sum_{R_n=0}^{R_n^{max}-1} \left[\sum_{j_x,j_y \in S_{R_n} \setminus S_{R_{n+1}}} F_{R_n}(j_x,j_y) C(j_x,j_y) \Delta p(R) \right] \\ &+ \sum_{j_x,j_y \in S_{R_n^{max}}} F_{R_n^{max}}(j_x,j_y) C(j_x,j_y) \Delta p(R_n^{max}). \end{aligned} \tag{38}$$

If we applied that to the current, we have

$$\begin{aligned} J_\alpha^{VLEM\_EG}(x,y) &= \int \frac{p_\alpha}{\gamma(\mathbf{p})} f(x,y,\mathbf{p}) \mathbf{dp} \\ &= \sum_{R_n=0}^{R_n^{max}-1} \left[\sum_{j_x,j_y \in S_{R_n} \setminus S_{R_{n+1}}} \frac{p_\alpha(j_\alpha)}{\gamma(j_x,j_y)} f_{R_n}(j_x,j_y) C(j_x,j_y) \Delta p(R) \right] \\ &\quad + \sum_{j_x,j_y \in S_{R_n^{max}}} \frac{p_\alpha(j_\alpha)}{\gamma(j_x,j_y)} f_{R_n^{max}}(j_x,j_y) C(j_x,j_y) \Delta p(R_n^{max}). \end{aligned} \tag{39}$$

Equations (38) and (39) are valid only if $R_n^{max} \geq 1$ either there is no mesh refinement and the expression of Eqs. (35) and (34) are used. The integrated domain $S_{R_n^{max}}$ corresponds to the grid of maximum level of refinement, for example, the yellow dot in Fig. 2. While the integrated domain $S_{R_n} \setminus S_{R_{n+1}}$ can be compared to the brown dot for $R = 1$. As we only have rectangular grids, the interfaces between two grids are simple, for example, in 2 V we will have the border and the corners of the rectangle. So, we can write the value of the coefficient C in tables depending on the kind of interface we have, for two dimensions in momentum we reference the value of C in Tables II and III and for the three dimensional cases we reference the values in the Appendix A (Tables X–XII). It is important to take into account the points that are not represented in the limits of the tables, but which play an important role, to recover the values on the tables.

Table II shows the coefficient of C for a 2 V code in the case of a side between two grids R and $R - 1$. The parameters n and M will help writing those coefficients in a general way for the four different sides. We have $n = j_{\alpha_{max}}(R)$ and $n + 1 = j_{\alpha_{max}}(R) + 1$ where α will represent the dimension orthogonal to the side. We also have $2M$ that represents the even values and $2M + 1$ the odd values of $j_\beta(R)$, with β the other direction of the 2 V system. This is valid as long as $j_{\beta_{min}}(R) + 2 < 2M < j_{\beta_{max}}(R) - 2$. The blank spaces in Tables II and III represent the points in $R - 1$ that do not exist and so do the values of C.

Table III shows the values of C where the intersection between the grids R and $R - 1$ is a corner located at (n,m). The values of n are the same as for Table II and those of m are $m = j_{\beta_{max}}(R)$ and $m + 1 = j_{\beta_{max}}(R) + 1$. As this table is symmetrical, we can choose α and β to represent either x and y or y and x.

For both Tables II and III, the points that should exist and that are not represented got a value of $C = 1$.

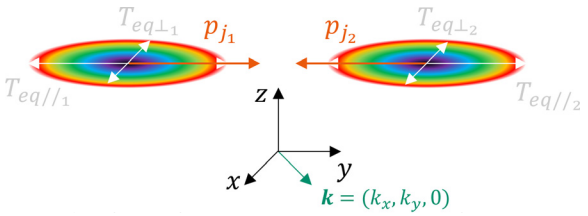

FIG. 6. Sketch of the initial configuration with two counter-propagating beams along the y direction. The linear perturbation has a wave vector k with components in the x, y plane. For Secs. IV A and IV B, we have $T_{eq\perp_1} = T_{eq//_1} = T_{eq\perp_2} = T_{eq//_2} = 6keV/k_B$. For Sec. IV C, we have $T_{eq\perp_1} = T_{eq//_1} = 2MeV/k_B$ and $T_{eq\perp_2} = T_{eq//_2} = 5keV/k_B$.

All integrals over the momentum components done with VLEM_EG use this method. It also extends to all Casimir invariants (i.e., also the moments of the distribution function) and the kinetic energy.

Note that the choice of three points for the graded grid is compatible with this method as the modification affects the three closest points to the edge of the grid.

IV. RESULTS

In this section, we test the computational acceleration and accuracy of the results for VLEM2D2V and VLEM2D3V with and without embedded grids. The amplification of magnetic fields can be driven also by any momentum anisotropy. The first process is exemplified by the Weibel instability (WI),[52–55] when the perpendicular temperature $T_\perp$ is larger than the parallel temperature $T_\parallel$, where the symbols $\perp$ and $\parallel$ refer to the direction with respect to the wave vector.

As mentioned in the introduction, previous linear studies of the Weibel-type instabilities have identified different classes of electron beam-plasma instabilities: the current filamentation instability (CFI),[38,56] driven by a momentum anisotropy (with the wavevector $\mathbf{k}$, being perpendicular to the beams), the electrostatic two-stream instability, with $\mathbf{k}$ aligned to the beams, and finally the oblique instability (OI),[39,40] for an arbitrary direction of $\mathbf{k}$.

Current filamentation and oblique instabilities can lead to the emergence of a direct cascade to large wave numbers in the presence of turbulent processes. Therefore, these instabilities provide an ideal framework for studying these processes and the effectiveness of AMR techniques in describing them.

Our code is first checked against the problem of two counter-streaming electrons beams (Fig. 6) for CFI and OI [taking into account the possibility of an electrostatic two-stream instability (TSI)] that takes place in the refined regions in the momentum space. We will look at the transition between OI and CFI first for a low relativistic case and then for an antisymmetrical relativistic case.

A. Study of the instability transition from the OI to the CFI with VLEM2D2V

1. Performance

We study a problem with two counterpropagating electron beams in the y direction as represented in Fig. 6. We can write the distribution function as $f(x, y, \mathbf{p}, t = 0) = \sum_{j=0,1} n_j F_{0j}(\mathbf{p})$ where

$$F_{0j} = \frac{1}{2\pi m^2 c^2 \beta_{th_j}^2} \exp\left(-\frac{p_x^2 + (p_y - p_j)^2}{2m^2 c^2 \beta_{th_j}^2}\right), \tag{40}$$

with $\beta_{th_j} = \sqrt{k_B T_{eq_j}/mc^2}$ as the normalized thermal velocity and p_j as the normalized mean momentum of the beam. This kind of initial distribution function is well adapted to VLEM_EG because there is no initial spatial fluctuation.

Depending on the direction of the wave vector $\boldsymbol{k}$ in the plane, the nature of the instability is modified leading to two stream instability ($\boldsymbol{k} = (0, k_y)$), the current filamentation instability ($\boldsymbol{k} = (k_x, 0)$) or more generally to the Weible-type oblique ($\boldsymbol{k} = (k_x, k_y)$) instability. In Ref. 57, the authors solve the dispersion relation and find two different branches for the oblique instability, the "high" and the "low" frequency branches. We will focus here on the "low" frequency branch already studied in case B of Ref. 5. This branch is propagative and for $k_B T_{eq_1} = k_B T_{eq_2} = 6keV$, $p_0 = -p_1 = 0.9mc$ and $n_1 = n_2 = \frac{n_0}{2}$ can be triggered by exciting the mode $k_x d_e \approx 3.07$ and $k_y d_e \approx 5$ ($d_e = c\omega_p^{-1}$). Fixing the normalized dimension of the box to $L_x = 4.088\, d_e = \frac{2\pi}{\Delta k_x}$ and $L_y = 2.513\, d_e = \frac{2\pi}{\Delta k_y}$, this corresponds to the Fourier mode $(n_x, n_y) = (2, 2)$. We will put an initial perturbation for this mode on the magnetic field in the form

$$\delta B_z = \delta B_0 \sin(n_x \Delta k_x x + n_y \Delta k_y y). \tag{41}$$

We take $e\delta B_0/(m\omega_p) = 0.005$ as initial normalized perturbation, $\Delta t\, \omega_p = 0.004$ as a time step with 60 000 iterations, the space grid $N_x N_y = 128 \times 128$ distributed over 64 MPI-processes (512 cores) including eight OMP threads per patch.

As we perform the same numerical experiment as in case B of Ref. 5, we will be able to compare with their results and we will go further in time. This type of simulations will lead to a transition from oblique to current filamentation instabilities. This will produce magnetic structures where the distribution function will take more space with respect to the momentum variables. This is one of the reasons why a mesh refinement method is needed.

We will look at four different cases with VLEM_EG over two values of ε ($\varepsilon = 10^{-5}$ and $\varepsilon = 10^{-3}$) [Eq. (31)].

- Case 1, with normalized momentum boundaries between −3 and 3 and with 127 points uniformely located in each direction.

TABLE V. Different cases studied with VLEM2D2V.

VLEM	Case A	Case B	Case C
Boundary ($-p_{x/y}/p_{x/y}$) (in mc)	−3/3	−3/3	−6/6
$\Delta p_{x/y}$ (in mc)	4.76×10^{-2}	2.38×10^{-2}	4.76×10^{-2}

TABLE IV. Different cases studied with VLEM2D2V_EG. Boundary for $R = 0$ means $(p_{(x/y)_{min}}(0)/p_{(x/y)_{max}}(0))$.

VLEM_EG	Case 1	Case 2	Case 3	Case 4
R^{max}	0	1	2	2
Boundary for $R = 0$ (in mc)	−3/3	−3/3	−3/3	−6/6
$\Delta p_{x/y}(R^{max})$ (in mc)	4.76×10^{-2}	4.76×10^{-2}	2.38×10^{-2}	4.76×10^{-2}

TABLE VI. Computational time information for different cases explicit in Table IV for VLEM_EG with $\varepsilon = 10^{-5}$.

	Case 1	Case 2	Case 3	Case 4
Computational time (in hours)	3.97	3.32	9.4	5.23
Theoretical computational time (in hours)	3.77	2.92	8.17	4.83
Mean number of momentum points	16 129	12 500	34 008	19 920
Mean computational time per point per iteration (in 10^{-7} s)	4.62	4.98	5.18	4.92

TABLE VII. Computational time information for different cases explicit in Table IV for VLEM_EG with $\varepsilon = 10^{-3}$.

	Case 1	Case 2	Case 3	Case 4
Computational time (in hours)	3.97	2.88	6.68	3.63
Theoretical computational time (in hours)	3.77	2.40	6.17	3.38
Mean number of momentum points	16 129	10 282	25 668	13 954
Mean computational time per point per iteration (in 10^{-7} s)	4.62	5.25	4.88	4.88

TABLE VIII. Computational time information for different cases explicit in Table V for VLEM.

	Case A	Case B	Case C
Computational time (in hours)	3.77	15.38	15.51
Number of momentum points	16 129	64 009	64 009
Mean computational time per point per iteration (in 10^{-7} s)	4.38	4.50	4.54

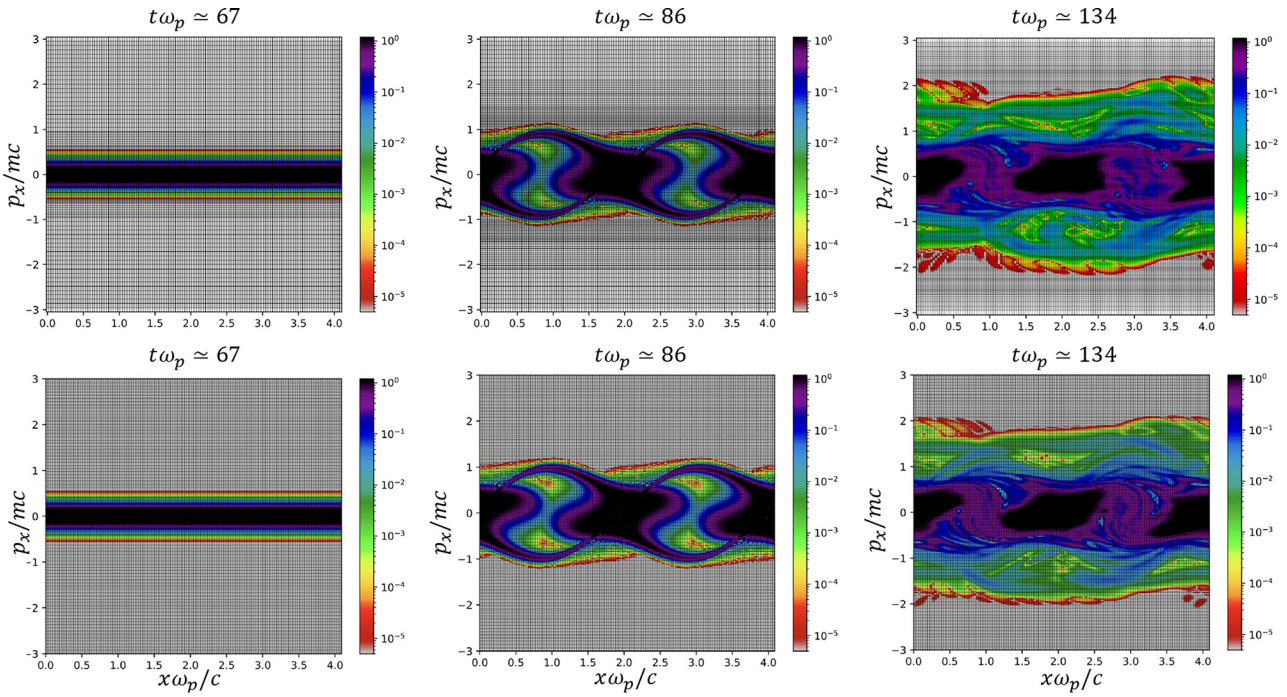

FIG. 7. Comparative representations of the distribution function in $x - p_x$ phase space, integrated over p_y, for case 3 with $\varepsilon = 10^{-5}$ (top) and case B (bottom) at $t\omega_p \simeq 67$, 86, and 134 for a point located at ($i_y = N_y/2$). The grids are represented and for case 3 which is made with VLEM_EG, it follows the structures well.

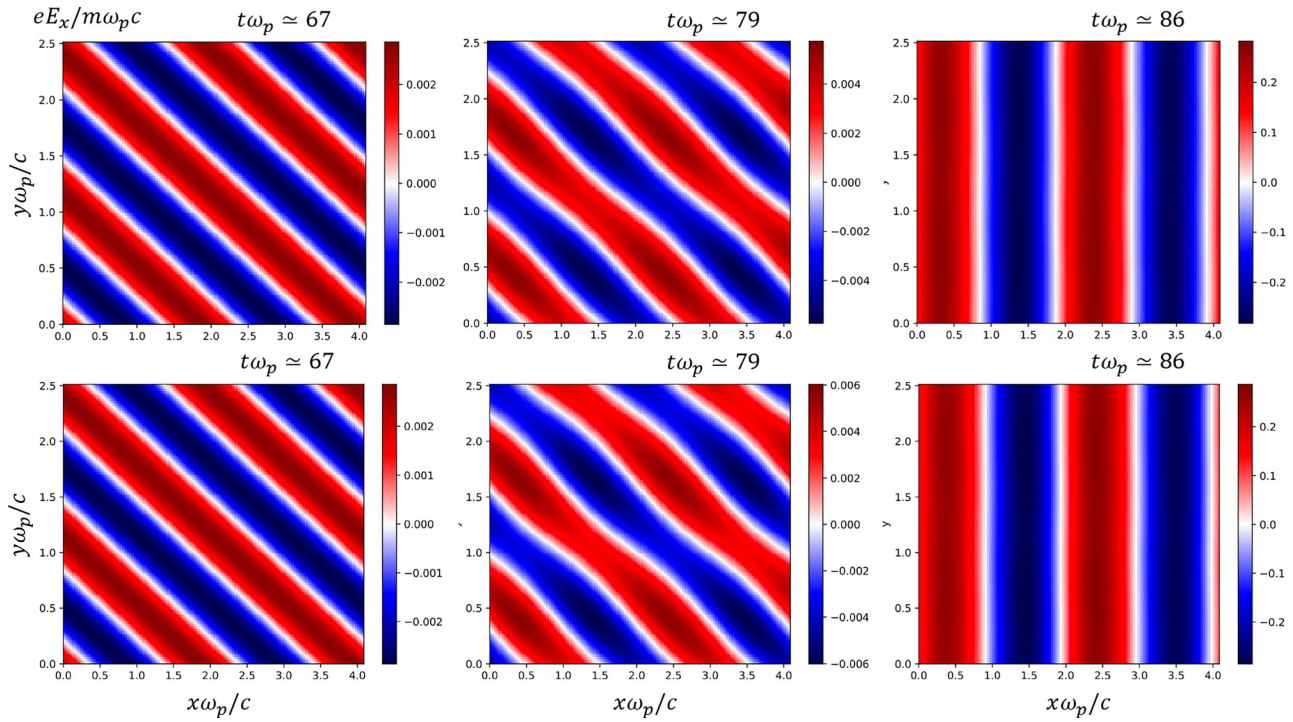

FIG. 8. Comparative plots of the normalized amplitude of E_x for the simulations case 3 (top) for $\varepsilon = 10^{-5}$ and case B (bottom) at $t\omega_p \simeq 67$, 79, and 86. The transition between OI and CFI happens in both codes in the same way.

- Case 2, with one level of refinement over the same boundary with the grid $R = 0$ with 64 points in each direction, leading to a maximum of 127 points in each directions for $R = 1$.
- Case 3, we add another level of refinement to case 2.
- Case 4, we keep three levels of refinement but we increase the size of the box up to −6/6 for the normalized momentum boundary. This will give the same $\Delta p_{x/y}$ as case 2. This should reduce the loss of density as the distribution function will be far from the edge of the box.

Those different parameters are recalled in Table IV.

We will compare these simulations with VLEM in three cases developed in Table V. The first case (case A) with normalized momentum boundaries between −3 and 3 for p_x and p_y and with 127 points in each direction. The second one (case B) with the same boundary and 253 points in each momentum direction. The last one (case C) with 253 points in each direction with normalized boundaries between −6 and 6.

Cases 1 and 2 from VLEM_EG will be compared with case A of VLEM, case 3 with case B and case 4 with case C, this is to have the same $\Delta p_{x/y}$ and the same box size.

The results of the computational time for each simulation are shown in Table VI for $\varepsilon = 10^{-5}$ and in Table VII for $\varepsilon = 10^{-3}$ in Eq. (31). We also add the theoretical time (τ_t) which corresponds to the time the code should take if the number of points is linearly proportional to the computational time of the reference code (τ_{VLEM}). We can determine it by using the mean number of momentum points of VLEM_EG ($N_{p_{EG}}$) and the number of momentum points of VLEM ($N_{p_{VLEM}}$) in

$$\tau_t = \frac{N_{p_{EG}}}{N_{p_{VLEM}}} \tau_{VLEM}. \tag{42}$$

The results for the comparative case with VLEM are shown in Table VIII.

For both values of ε, we can see that case 1 gives a slightly greater computational time of about 5% in comparison with case A. This shows that the changes induced by the AMR method, which are not related to the creation of the grid, are well implemented in VLEM. Although the increase in computational time may be due to a difference in the processors selected for the simulation, we recognize an increase in the computational effort due to the implementation of the AMR scheme.

For the next three cases we can see that both value of ε will give an acceleration of the computational time. Case 2 shows a small acceleration of 12% for $\varepsilon = 10^{-5}$ and 24% for $\varepsilon = 10^{-3}$ in comparison with case A. This is due to an already low precision in the initial parameters for the grid $R = 0$ so an important need of grid precision.

For case 3 and even more for case 4, the computational gains become much more important. It is respectively 39% and 66% for $\varepsilon = 10^{-5}$ and 57% and 76% for $\varepsilon = 10^{-3}$. Results for case 3 show that a high precision is needed but mostly at the center of the momentum grid (Fig. 7). This gives a good reduction of the computational time in comparison with VLEM that needs to take the same precision even where it is not necessary. For case 4, the computational time of VLEM_EG is greater than case 2 (about 50% more), even if there is the same momentum precision, mainly due to the limit of the grid of the momentum box.

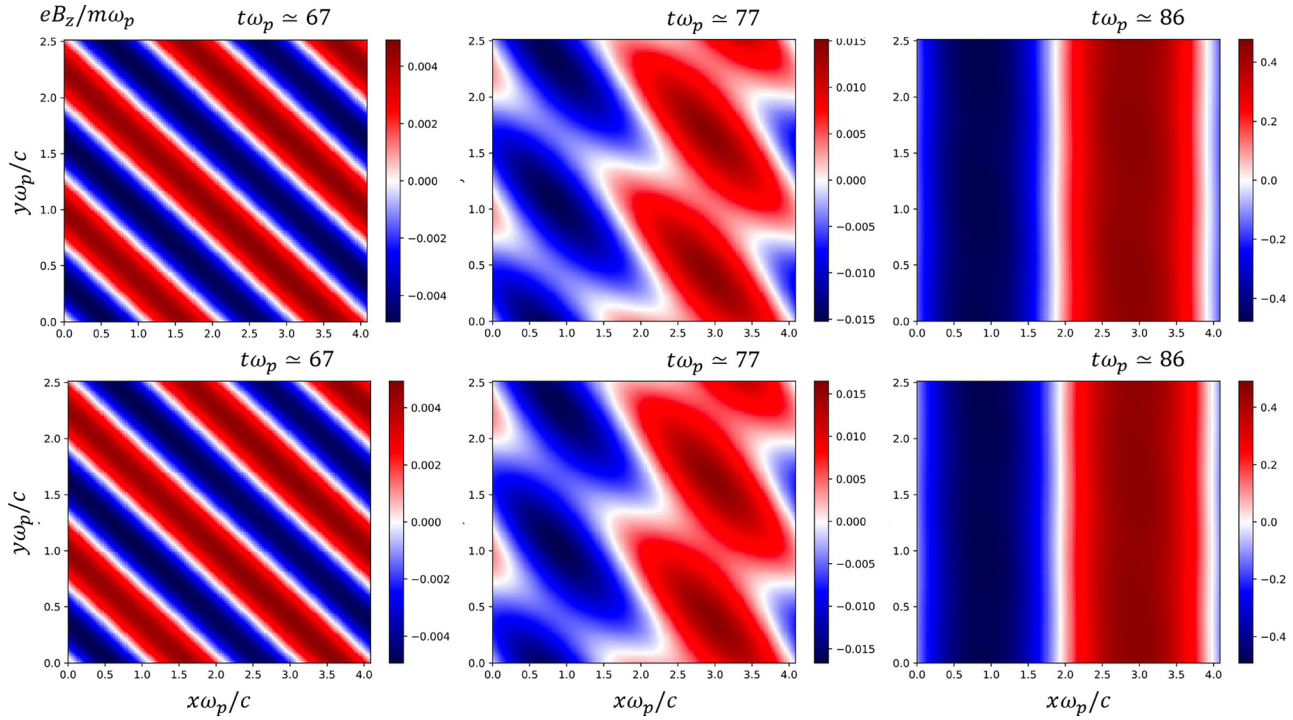

FIG. 9. Comparative plots of the normalized amplitude of B_z for the simulations case 3 (top) for $\varepsilon = 10^{-5}$ and case B (bottom) at $t\omega_p \simeq 67$, 77, and 86. The transition between OI and CFI happens in both codes in the same way.

Tables VI–VIII also show the mean computational time per point per iteration. This value counts a point located at (x, y, p_x, p_y) only once, even if it is defined on several grids, and takes into account the parallelization MPI and OpenMP to be more comparable between the simulations. The closer the results for VLEM_EG are to the results of VLEM, the better the implementation has been done. Looking at cases 2–4 we found a mean value of this parameter of $5.01 \times 10^{-7}s$, while the mean value of this parameter for VLEM is $4.52 \times 10^{-7}s$. This difference of about 11% is small and shows that the modification of the algorithm due to the mesh refinement method (Algorithm 1) is well implemented and not expensive in terms of computational time.

To verify the results of VLEM_EG we compare the value of the distribution function in (x, p_x) (Fig. 7) and also the spatial evolution of E_x (Fig. 8) and B_z (Fig. 9) between VLEM and VLEM_EG for $\varepsilon = 10^{-5}$. We choose to show the comparison between cases 3 and B because those are the cases with the greatest precision. It is easily verified that both simulations provide the same results for the transient time (left), the linear phase (middle), and the saturation (right) for Figs. 7–9. We also recovered in Fig. 7 the electrostatic and magnetic trapping structure that were observed in Ref. 5 but as we have been further in time we can oberve a kind of final stationary state. It is reached at time $t\omega_p \simeq 122$ and continues until the end of the simulation. The electrons will be trapped in the region of low magnetic field energy (i.e., when $B_z \approx 0$). This is maintained by the self-generated electric field, which oscillates twice as much in space as the magnetic field (Figs. 8 and 9). This evolution is typical of CFI and has already been discussed for a non-relativistic case in Ref. 58.

For $\varepsilon = 10^{-3}$, we get the same results for Figs. 8 and 9 but there is the appearance of an error of the order of 10^{-5} which appears in $f(x, p_x)$ near the edge which is not present here for $\varepsilon = 10^{-5}$ in Fig. 7. This one seems to be negligible for the timescales we are considering but could probably play a role on longer timescales. Even though the figure is not shown for this case, we will see later in Fig. 11 the error due to $\varepsilon = 10^{-3}$.

2. Influence of the filamentation

One of the most difficult points in developping the AMR code is the selection of the refinement criterion for cell splitting. The criteria have to be carefully determined so that key features involved in physical processes of interest are properly described. For instance in AMR techniques, used in PIC codes for study for instance of reconnection processes, the PIC method[35,36] requires that the cell size must be as small as the local Debye length and this spatial scale must be taken into account in the refinement criteria. In the oblique instabilities (OI), many unstable modes, e.g., driven by the high frequency branch of OI, are expected to grow leading to a direct-like cascade to smaller scales. In Refs.,[25,37] it has been shown that such a process can lead to the reversion of the energy transfer associated with this direct-like turbulent cascade. The current filamentation instability tend to re-inforce the production of thinner and thinner filaments in the phase space, coupling with the (kinematic) filamentation of the distribution function, a process referred to as the "reinforced" filamentation. This mechanism can induce deep modifications in the energy transfer and lead to a strong (stochastic) heating.

Here, the refinement criteria used in Eq. (31), allow such a physical process to be taken into account, particularly if the size of the elementary cell in momentum space is small enough to follow the process of augmented filamentation. This is particularly the case when the value of ε is chosen to $\varepsilon = 10^{-5}$. Furthermore, the AMR technique, restricted to the momentum space, is perfectly adapted to follow the formation of thin filaments and the method subdivides all cells that satisfy the refinement criterion so that it enables us to achieve an efficient refinement to recover a complicate re-inforced filamentation that takes place in the momentum space.

Note that this physical process is already at work in Fig. 10, which shows the distribution function in momentum space at three different times with the representation of the grid. The initial conditions chosen correspond to the excitation of a mode localized on the low-frequency branch, thus minimizing this type of process, which nevertheless seems to persist.

Indeed if we look at Fig. 10 that shows the evolution of the grid for case 4 for $\varepsilon = 10^{-5}$, we see the "heating" of the distribution function which will take more place in momentum space. At the bottom image, for $t\omega_p \simeq 134.4$ the internal grid is larger (around $p_{x_{min/max}}(2) = -4/4\,mc$) than the maximum grid for case 2 ($p_{x_{min/max}} = -3/3\,mc$). This shows that the size of the momentum box for case 2 needs to be increased, to follow some elements of the distribution function that may be lost due to the boundary of the momentum grid. This is the kind of scenario where this mesh refinement method is better suited. Note that even if some information is lost due to the fixed size of the momentum boundary in cases 1–3, the simulations still give accurate results as this has only a small impact.

Figure 10 also shows that AMR are well suited for the transient stage, here located at $t\omega_p \leq 75$. Indeed, during that phase the electric and magnetic fields are slowly increasing leading to low perturbation on the distribution function. This results in a lower number of points required for the refined grid, which in turn results in lower computational costs.

Figure 11 shows the same results as Fig. 10 but for $\varepsilon = 10^{-3}$. We can see that the number of points in the momentum space is always lower than for $\varepsilon = 10^{-5}$ which justifies the shorter duration of the simulation. However, this low number of points, together with the close proximity of the grid to the distribution function, leads to the appearance of an error of up to 5×10^{-5} in many cells. For this reason we will choose $\varepsilon = 10^{-5}$ in the next simulation.

B. Study of the instability transition from the OI to the CFI with VLEM2D3V

We will now look at the same instability but with three dimensions for the momentum. This makes possible the current in the z direction (J_z) and so the component of the related electric and magnetic fields, E_z, B_x, and B_y. Magnetic reconnection in the x-y plane is thus possible now. We now compare simulations performed with a smaller phase-space resolution than before, to reduce the global computational cost. So, we choose $\Delta t\omega_p =$

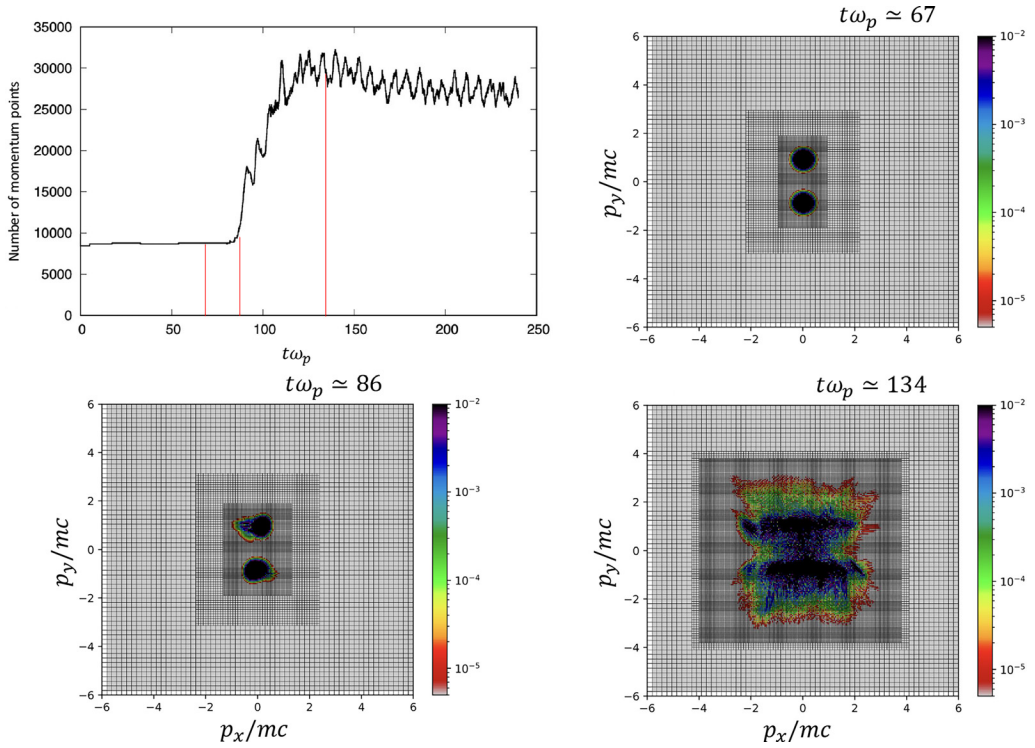

FIG. 10. (Top-left) Time evolution of the number of momentum points for case 4 with $\varepsilon = 10^{-5}$. The vertical red lines represent the different times of the three other figures. Representation of the distribution function in $p_x - p_y$ phase space for case 4 with $\varepsilon = 10^{-5}$ at $t\omega_p \simeq 67$ (top-right), 86 (bottom-left) and 134 (bottom-right) for a point located at ($i_x = N_x/2, i_y = N_y/2$). The embedded grids are represented and follow accurately the evolution of the distribution function.

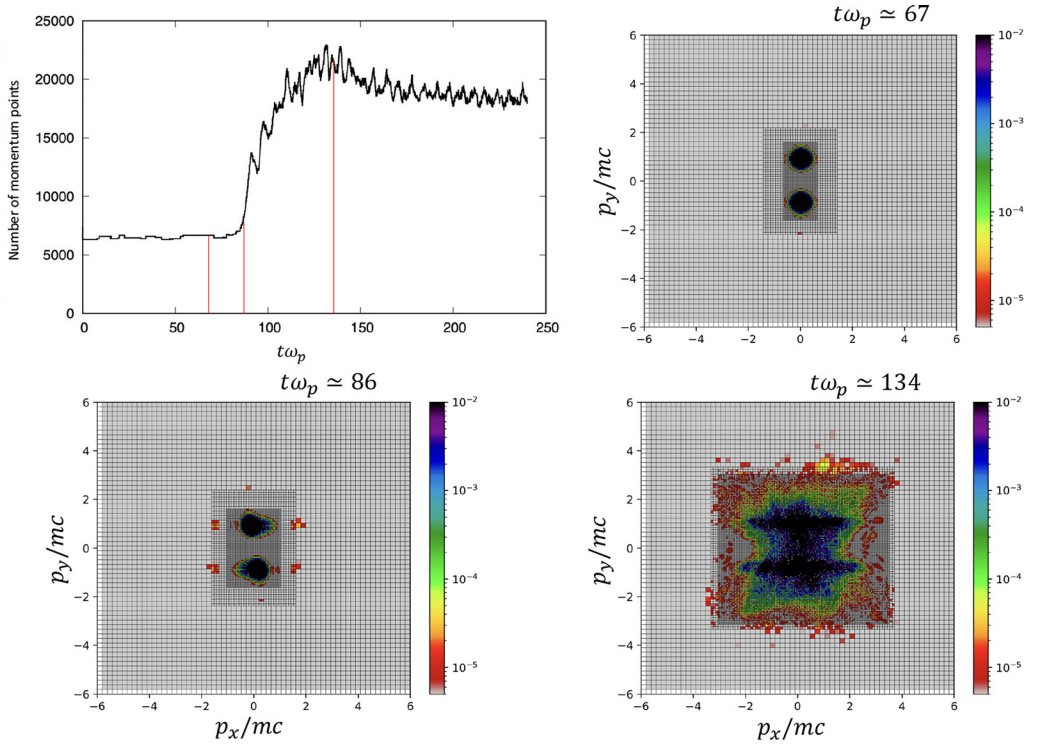

FIG. 11. (Top-left) Time evolution of the number of momentum points for case 4 with $\varepsilon = 10^{-3}$. The vertical red lines represent the different times of the three other figures. Representation of the distribution function in $p_x - p_y$ phase space for case 4 with $\varepsilon = 10^{-3}$ at $t\omega_p \simeq 67$ (top-right), 86 (bottom-left) and 134 (bottom-right) for a point located at $(i_x = N_x/2, i_y = N_y/2)$. The embedded grids are represented and showing the appearance of edge errors due to the large value of ε.

0.008 with 24 000 time steps over eight time intervals each after 3000 time steps, $N_x N_y = 64 \times 64$ over 128 processors with eight OMP threads per patch. All the other parameters describing the perturbation and the initial distribution function are the same as in Subsec. IV A.

For VLEM_EG we choose one level of refinement with boundaries for each normalized momentum coordinate between -3 and 3, a level $R = 1$ up to 79^3 points and $\varepsilon = 10^{-5}$. This will be compared to VLEM with 79^3 points for the momentum dimension. Both simulations have $\Delta p_x = \Delta p_y = \Delta p_z = 0.0769mc$.

TABLE IX. Computational time (in hours) for VLEM2D3V between the different time step intervals. The theoretical time is computed using Eq. (42), where the mean number of momentum point can be deduced from Fig. 13.

Time intervals (in $t\omega_p$)	0–24	24–48	48–72	72–96
VLEM	7.33	7.33	7.33	7.33
VLEM_EG	3.03	3.07	3.05	3.77
Theoretical	2.88	2.88	2.88	3.48
Time intervals (in $t\omega_p$)	96–120	120–144	144–168	168–192
VLEM	7.33	7.33	7.33	7.33
VLEM_EG	3.03	3.07	3.05	3.77
Theoretical	2.88	2.88	2.88	3.48

Looking at the computational time needed for each time interval (Table IX), we can again compare VLEM, VLEM_EG and the theoretical time. It is not surprising to find out that the computational time for VLEM is quite the same for each simulation due to the fixed number of points. For VLEM_EG, we found out an average reduction of the simulation time of 54%. This result however is strongly influenced by the number of points at each time of the simulation (Fig. 13). Indeed, before $80t\omega_p$, the number of points in momentum is fixed at 193 605. If we compare this to the evolution of the distribution function (p_x, p_y) in the 2D2V case (Fig. 10) or in (p_y, p_z) (Fig. 12), we are still in the case where there is no visible evolution in the shape of the distribution function, i.e., we are still in the transient stage. This means that the number of cells for $R = 1$ is the lowest, and so the computational cost is the smallest one (Fig. 13 and Table IX). Then, there is an increase in the number of points up to 303 575 with a time oscillation behavior. This corresponds, in the first part, to the linear stage and so to the expansion of the distribution function in the momentum space (Figs. 10 and 12). In the second part we are at the beginning of the saturation regime before the final stationary state, explaining the oscillations. For those two parts, the distribution function is defined on a larger set of cells which changes over time. This leads to an increase in the computational time with a maximum reached between $t\omega_p \simeq 96$ and $t\omega_p \simeq 120$ (Fig. 13). Then, the oscillations decrease and stop at 284 409 points for the last time interval leading to a slight decrease in the computational time. This final state corresponds to an equilibrium where the instability is reached and the electrostatic trapping keeps a regular motion in time (as in Fig. 7).

It is also interesting to focus on the reason for this low computational cost. Indeed, the number of points in each direction is lower than the one for case 2 in the previous subsection. That means that the grid $R = 1$ will fill the entire domain (p_x, p_y) at least at the saturation, but the computational gain in time is made on the p_z direction where there is almost no fluctuation or expansion of the distribution function (Fig. 12). The grid in p_z is around $p_{z_{max/min}}(R = 1) = -1.8/1.8\, mc$ leading to approximately 60% of theoretical acceleration. Looking at Fig. 12 it can be noted that there are slight differences between the two numerical results at $t\omega_p = 134.4$, due to the near edge of the box which has a small effect on the distribution function. However we will see from the energy comparison (Fig. 13) that this difference has little effect on the code.

If we also compute the mean computational time per point per iteration, we found out $5.57 \times 10^{-7} s$ for VLEM against $6.03 \times 10^{-7} s$ for VLEM_EG. This shows that VLEM2D3V is slightly less efficient than VLEM2D2V but also that the AMR technique is better implemented on this version.

The low expansion of the distribution function leads to low current in the z direction and so E_z, B_x, and B_y will have a low contribution to the problem. This explains the similar results between the two- and three- dimensional cases in the momentum space.

We will now compare the densities and the different energies obtained by VLEM and by VLEM_EG (Fig. 13). Due to the similarity between the simulations, the evolution of the energies is very close to the two-dimensional case in momentum. We calculate the kinetic, the magnetic and the electric energies respectively as $E_k = \frac{1}{L_x L_y} \int d^2x \int d^3p\, mc^2(\gamma - 1) f$, $E_m = \frac{\varepsilon_0}{2L_x L_y} \int d^2x c^2 B^2$ and $E_e = \frac{\varepsilon_0}{2L_x L_y} \int d^2x E^2$. We see that there is quasi no difference for the evolution of each energy between each code. This is also the case for all the 2D2V cases. The transfer between kinetic and magnetic energies is well computed and is a characteristic of the current filamentation instability. We also note that the total energy ($E_{tot} = E_k + E_m + E_e$) and the mean density $\left(n = \frac{1}{L_x L_y} \int d^3p \int d^2x f\right)$ are well conserved (around 1% of variation) and are quasi-similar in both codes. We remind that the integral for the computation of the density, as it is the case for all the integration involving the momentum space, is done for VLEM_EG with the method introduced in F while for VLEM the standard summation is done.

These cases validate the good implementation and the gain of computational time of the AMR method in VLEM2D2V and VLEM2D3V.

C. Study of the relativistic oblique instability with VLEM2D2V

In this part we will look at more relativistic and asymmetrical interaction between the electron beams. To better represent the distribution we will now describe each beam with a drifting Maxwell–Jüttner distribution function. The representation of Fig. 6 is still valid but the temperature will be the same in the x and y directions due to the Maxwell–Jüttner distribution. Its three-dimensional expression is

$$F_{0j} = \frac{1}{4\pi m^3 c^3 \theta_j K_2\left(\frac{1}{\theta_j}\right)} \exp\left(-\frac{(\gamma(\mathbf{p}) - \beta_j p_y)}{\theta_j}\right), \tag{43}$$

where β_j is the normalized drift velocity in the y direction, with $\gamma(\mathbf{p})$ the relativistic factor, $1/\theta_j = mc^2/(k_B T_j)$ is the normalized inverse temperature and K_2 is the modified Bessel function of second order. For the problem we are going to address, we can specialize Eq. (43), which is valid in a general 3D3V problem, to a general 2D2V problem to get a good momentum precision. Then, the distribution function can be rewritten in a similar way with $F_{0j} = A \exp(-\theta_j^{-1}(\gamma(\mathbf{p}) - \beta_j p_y))$, with A a normalization factor.

We will perturb the system to excite the oblique instability but this time with asymmetric temperature with two beams, a cold one and a hot one. For this case we take $L_x = 15.707\, d_e$ and $L_y = 20.943\, d_e$. This normalized spatial box size which corresponds to $\Delta k_x\, d_e = 0.40$, $\Delta k_y\, d_e = 0.30$. For the cold beam we take a temperature $k_B T_1 = 5keV$ and for the relativistic one $k_B T_2 = 2MeV$. We keep $n_1 = n_2 = n_0/2$ but we change $\beta_1 = -\beta_2 = -0.745$. These parameters correspond to Fig. 3 of Ref. 59 and so we can refer to that to choose the maximum growth rate. The initial perturbation is again applied to the z magnetic component in the same way as before [Eq. (41)] with $(n_x, n_y) = (1, 1)$ corresponding to the most unstable mode. For the other normalized parameters we take: $e\delta B_0/(m\omega_p) = 0.005$, $\Delta t\omega_p = 0.01$ with 30 000 time step, $N_x N_y = 256^2$ and 2048 processors. Each process simulates one patch on eight threads through an Open-MP parallelization.

This type of simulation has already been done in Ref. 60 where the authors take the same parameters except for the temperature. In first place they take $k_B T_1 = 10keV$ and $k_B T_2 = 2MeV$ and then $k_B T_1 = 20keV$ and $k_B T_2 = 1.5MeV$, these choices where made for computational reasons. Indeed as the hot beam occupies a large interval in momentum space, the size of the box needs to be large but as the cold beam is very narrow in comparison, the precision needs to be important to describe it well. Therefore, a warmer cold beam was chosen to reduce the need for a high sampling in momentum. It is for this type of cases that the AMR method is well suited as it increases the accuracy required for the cold beam but only in the momentum space where it is needed.

We make the VLEM2D2V simulation with a box $p_{x_{min}}/p_{x_{max}} = -30mc/30mc$ and $p_{y_{min}}/p_{y_{max}} = -20mc/80mc$ and with $N_{p_x} = 255$, $N_{p_y} = 511$. For the comparison we take VLEM2D2V_EG with one level of refinement where the maximum number of points corresponds to that of VLEM and with the same box size. We choose to decrease the parameter ε to 10^{-7} to better follow the micro-oscillations of the distribution function (i.e., as we have $R^{max} = 1$, the value of ε is related to the L^∞ error of distribution function).

Again, the results are the same, both for the evolution of the distribution function and for the evolution of the field. The reference code takes 19 h and 37 min to compute compared to 8 h and 56 min for the AMR method, hence about 55% of gain in computational time.

We can see from Fig. 14 that during the first steps, at the beginning of the transient time, the refined grid is well focused on the cold beam, resulting in a low number of points and thus a lower computational cost. Then, when the instability starts in the region of $p_x < 0$ the grid follows well the filamentation and acceleration of particles. The fact that at the end of the simulation the AMR grid still represents less than 60% of the total grid explains the good acceleration. This is due to the smoothness of the distribution function for $p_x > 0$.

The mean computational time per point per iteration is 5.64×10^{-7} for VLEM against 5.36×10^{-7} for VLEM_EG. This

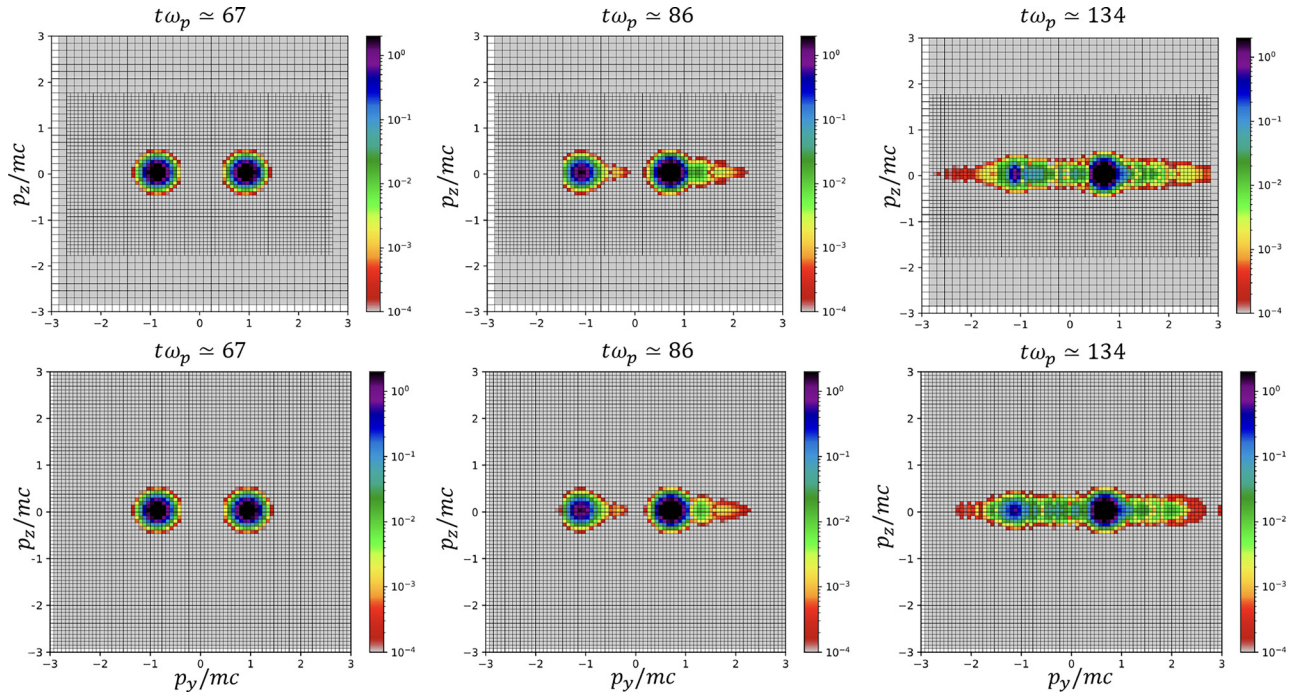

FIG. 12. Comparative representations of the distribution function in $p_y - p_z$ phase space for VLEM_EG2D3V (top) with $\varepsilon = 10^{-5}$ and VLEM2D3V (bottom) at $t\omega_p \simeq 67$, 86, and 134 for a point located at $(i_x = N_x/2, i_y = N_y/2)$. The embedded grids of VLEM_EG are represented explaining the computational gain time due to the nonexpansion in p_z.

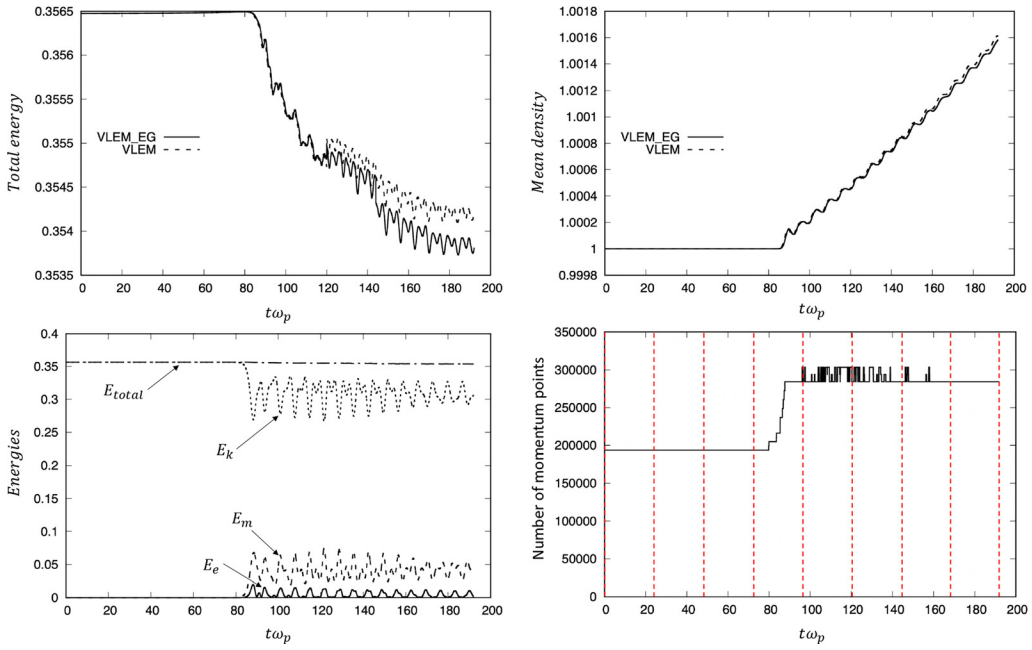

FIG. 13. Comparison of the evolution of the total energy (top-left) and the mean density (top-right) between VLEM2D3V and VLEM_EG2D3V. (Bottom-left) Evolution of the kinetic (dotted line), magnetic (dashed line), electric (line), and total energy for VLEM_EG2D3V. (Bottom-right) Time evolution of the number of velocity points of the finer grid $R = R_n^{max}$ for VLEM_EG2D3V. The red dashed lines represent the different time intervals used in Table IX.

result may seem unlogical as VLEM_EG has a more complex algorithm than VLEM and so should have a longer mean computational time per point per iteration, but it is possible that the smaller number of points per momentum grid could have led to easier memory access and thus a faster simulation per point.

At the end of this simulation, another transition from the OI to CFI is triggered. Further investigations on those relativistic oblique modes with VLEM_EG will be discussed later in another article.

V. CONCLUSION

The AMR technique is one of the most promising methods for studying physical phenomena which are highly localized and involve a numerous physical scales which are coupled with each others. The standard AMR technique has been first introduced in PIC codes to achieve large-scale kinetic simulations. The AMR technique, when used in the configuration space, subdivides the computational cells locally in space and dynamically in time. The technique leads to main problems. First, it is very difficult to reduce the numerical noise associated with the electromagnetic fields when implementing the AMR technique in the staggering spatial grid.[36,61] The electromagnetic field information must be communicated properly among the hierarchical levels, so that electromagnetic waves must propagate smoothly across the boundary refinement regions without causing a numerical instability. Another challenge in the AMR usually implemented in PIC solvers is to keep the charge load balance among the processors by controling the number of superparticules in refined cells.[36] Because of the Eulerian feature of the distribution function, the semi-Lagrangian scheme simplifies the programming structure because one does not have to cope with the allocation of the "particles" between processors since Euler elements stay in place.

Not only does the use of AMR techniques, restricted to the momentum space alone, remove one of the technical hurdles associated with sampling the momentum space, i.e., reducing the N_{vlas}/N_{pic} numerical cost of a Vlasov code compared with a PIC code with fixed g_{PIC} and with equal space coordinate sampling of the Vlasov code, but it also offers the possibility of adapting the refinement criterion to the filamentation process of the distribution function in the momentum space. This enables one to accurately follow multiscale process in **p**.

We have implemented a dynamically adaptive mesh refinement method in the relativistic solver VLEM, preserving the patch decomposition and the global architecture of the code. To do this, we have chosen to make embedded rectangular grids only in the momentum space. The main modifications were the boundary condition of the cubic spline interpolation method and the way of computing integrals. An important part is also the downhill of information (from finer to coarser grids) that should be done after each advection to keep the

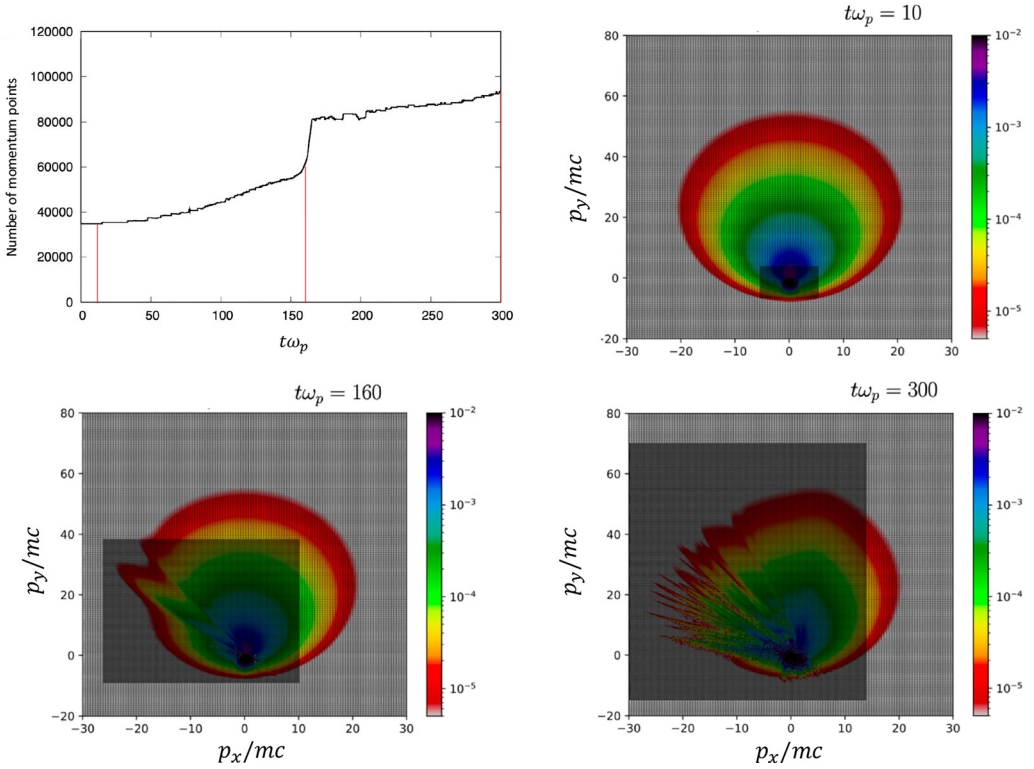

FIG. 14. (Top-left) Time evolution of the number of momentum points for VLEM_EG2D2V with $\varepsilon = 10^{-7}$. The vertical red lines represent the different times of the three other figures. Representation of the distribution function in $p_x - p_y$ phase space for VLEM_EG2D2V with $\varepsilon = 10^{-7}$ at $t\omega_p \simeq 10$ (top-right), 160 (bottom-left), and 300 (bottom-right) for a point located at ($i_x = 218 \times N_x/255, i_y = N_y/2$). The embedded grids of VLEM_EG are represented showing an initial good description of the cold beam then following the fluctuations of the distribution function.

independence between the different levels of refinement of the distribution function.

All together we test the code on an OI to a CFI case in both 2D2V and 2D3V configurations and in both nonrelativistic and relativistic cases. VLEM and VLEM_EG give the same results for the evolution of the distribution function and the fields. We reach up to 76% of acceleration for a case with a small value of ε and where the distribution function was initially really localized in momentum space before the "heat" of the distribution function caused it to spread out. The parameter ε, which is used for the refinement criterion [Eq. (31)], is one of the most important parameters to reduce the computational time of such a method but it must be chosen to adapt to the desired precision.

Beside of speeding up the computation, this type of mesh refinement methods is well suited to follow the acceleration of the distribution function. Indeed, a uniform grid need to find the balance between the size of the box, the accuracy and the computational time to follow acceleration process, where with the AMR we could take a wide box one and expect the mesh to be placed only where it is needed. This should reduce the losses at the edge while maintaining a reasonable computational time.

This AMR method is also well suited to describe relativistic processes characterized by an important scale mixing. This is the case for oblique and current filamentation instability with different beam temperatures, which is quite common in nature.

Future improvements that can be added to the method include spatial variation for the rectangular grids with redistribution of the MPI patch. Several rectangles could also be used to describe the same level to better follow the anisotropic process.

Although this article describes the method applied to VLEM2D2V and VLEM2D3V, the extension to VLEM3D3V is trivial and is already under development. It should facilitate its use by reducing its computational cost.

ACKNOWLEDGMENTS

The authors are indebted to the IDRIS (Institut du développement et des ressources en informatique scientifique) computational center, Orsay, France, for computer time allocation on their computers. This work was granted across to the HPC resources (Grant Nos. A110507290 and A0150507290) made by GENCI (Grand Équipement National de Calcul Intensif). This work has been partially carried out within the framework of the Eurofusion consortium (fundings obtained through FR-FCM AAP 2024 "Energy conversion processes mediated by the current and vorticity dynamics in low collision plasmas, in particular, are gratefully acknowledged). The views and opinions expressed here in do not necessarily reflect those of the European Commission.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

M. Antoine: Conceptualization (lead); Investigation (equal); Methodology (equal); Resources (lead); Software (lead); Validation (equal); Visualization (lead); Writing – original draft (lead); Writing – review & editing (equal). **A. Ghizzo:** Funding acquisition (equal); Investigation (equal); Resources (supporting); Software (supporting); Validation (equal); Visualization (supporting); Writing – review & editing (equal). **E. Deriaz:** Conceptualization (supporting); Investigation (equal); Methodology (equal); Validation (equal); Writing – review & editing (equal). **D. Del Sarto:** Funding acquisition (equal); Investigation (equal); Validation (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request. Different codes may be available.

APPENDIX A: VALUE OF C FOR THE 3V CASE

Here, we present the values of the C coefficient (see Sec. III F), used in the integrals, for the AMR method with three dimensions for the momentum space.

TABLE X. Value of $16 \times C(j_x, j_y, j_z)$ on a face. The points that are not represented and are not affected by other grid interfaces have a value of $C(j_x, j_y, j_z) = 1$.

n − 2	n − 1	n	n + 1	n + 2		
12	16	48		16.5	2M	2L
16	16	16			2M + 1	
16	16	16			2M	2L + 1
16	16	16			2M + 1	

TABLE XI. Value of $256 \times C(j_x, j_y, j_z)$ on an edge. The points that are not represented and are not affected by other grid interfaces have a value of $C(j_x, j_y, j_z) = 1$.

n − 2	n − 1	n	n + 1	n + 2		
264.25		262		255.75	m + 2	
					m + 1	
720	256	1152		262	m	2L
256	256	256			m − 1	
126	256	720		264.25	m − 2	
					m + 2	
					m + 1	
256	256	256			m	2L + 1
256	256	256			m − 1	
256	256	256			m − 2	

TABLE XII. Value of $4096 \times C(j_x, j_y, j_z)$ on a vertices. The points that are not represented and are not affected by other grid interfaces have a value of $C(j_x, j_y, j_z) = 1$.

n − 2	n − 1	n	n + 1	n + 2		
4091.875		4093		4096.125	m + 2	
					m + 1	
4888		4672		4093	m	l + 2
					m − 1	
5185		4888		4091.875	m − 2	
					m + 2	
					m + 1	
					m	l + 1
					m − 1	
					m − 2	
4195		4168		4093	m + 2	
					m + 1	
17 856	4096	23 040		4168	m	l
4096	4096	4096			m − 1	
10 728	4096	17 856		4195	m − 2	
					m + 2	
					m + 1	
4096	4096	4096			m	l − 1
4096	4096	4096			m − 1	
4096	4096	4096			m − 2	
4232.125		4195		4091.875	m + 2	
					m + 1	
10 728	4096	17 856		4195	m	l − 2
4096	4096	4096			m − 1	
−927	4096	10 728		4232.125	m − 2	

REFERENCES

[1]J. M. Dawson, Rev. Mod. Phys. **55**, 403 (1983).
[2]J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, J. Dargent, C. Riconda, and M. Grech, Comput. Phys. Commun. **222**, 351 (2018).
[3]O. Coulaud, E. Sonnendrucker, E. Dillon, P. Bertrand, and A. Ghizzo, J. Plasma Phys. **3**, 435 (1999).
[4]M. Sarrat, A. Ghizzo, D. Del Sarto, and L. Serrat, Eur. Phys. J. D **71**, 271 (2017).
[5]A. Ghizzo and D. D. Sarto, Phys. Plasmas **27**, 072104 (2020).
[6]G. Lehmann, Commun. Comput. Phys. **20**, 583 (2016).
[7]R. A. Fonseca, L. O. Silva, F. S. Tsung, V. K. Decyk, W. Lu, C. Ren, W. B. Mori, S. Deng, S. Lee, T. Katsouleas, and J. C. Adam, in *Proceedings of the Computational Science ICCS*, Lecture notes in computer science (2002), Vol. 2331, p. 342.
[8]T. D. Arber, K. Bennett, C. S. Brady, A. Lawrence-Douglas, M. G. Ramsay, N. J. Sircombe, P. Gillies, R. G. Evans, H. Schmitz, A. R. Bell, and C. P. Ridgers, Plasma Phys. Controlled Fusion **57**, 113001 (2015).
[9]M. Shalaby, A. E. Broderick, P. Chang, C. Pfrommer, A. Lamberts, and E. Puchwein, ApJ. **841**, 52 (2017).
[10]R. Bird, N. Tan, S. V. Luedtke, S. L. Harrell, M. Taufer, and B. Albright, IEEE Trans. Parallel Distrib. Syst. **33**, 952 (2022).
[11]L. Fedeli, A. Huebl, F. Boillod-Cerneux, T. Clark, K. Gott, C. Hillairet, S. Jaure, A. Leblanc, R. Lehe, A. C. Myers, C. Piechurski, M. Sato, N. Zaïm, W. Zhang, J. L. Vay, and H. Vincenti, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2022).
[12]N. Aunai, R. Smets, A. Ciardi, P. Deegan, A. Jeandet, T. Payet, N. Guyot, and L. Darrieumerlou, Comput. Phys. Commun. **295**, 108966 (2024).
[13]G. Lapenta, J. Comput. Phys. **334**, 349 (2017).
[14]F. Bacchini, Astrophys. J. **268**, 60 (2023).
[15]E. Fijalkow, J. Comput. Phys. **116**, 319 (1999).
[16]F. Filbet, E. Sonnendrucker, and P. Bertrand, J. Comput. Phys. **172**, 166 (2001).
[17]H. Liu, X. Cai, Y. Cao, and G. Lapenta, J. Comput. Phys. **492**, 224 (2023).
[18]N. Schild, M. Raath, S. Eibl, K. Hallatschek, and K. Kormann, Comput. Phys. Commun. **295**, 108973 (2024).
[19]N. Crouseilles, T. Respaud, and E. Sonnendrucker, Comput. Phys. Commun. **180**, 1730 (2009).
[20]C. Z. Cheng and G. Knorr, J. Comput. Phys. **22**, 330 (1976).
[21]E. Sonnendrucker, J. Roche, P. Bertrand, and A. Ghizzo, J. Comput. Phys. **149**, 201 (1999).
[22]T. Yakamura and T. Yabe, Comput. Phys. Commun. **120**, 122 (1999).
[23]N. Crouseilles, L. Einkemmer, and E. Faou, J. Comput. Phys. **283**, 224 (2015).
[24]P. Bertrand, D. Del Sarto, and A. Ghizzo, *The Vlasov Equation I: History and General Properties* (ISTE Ltd, London, 2019), chap. 5.4.
[25]A. Ghizzo, D. Del Sarto, and H. Betar, Phys. Rev. Lett. **131**, 035101 (2023).
[26]F. Huot, A. Ghizzo, P. Bertrand, E. Sonnendrucker, and O. Coulaud, IEEE Trans. Plasma Sci. **28**(4), 1170 (2000).
[27]N. Besse, G. Latu, A. Ghizzo, E. Sonnendrucker, and P. Bertrand, J. Comput. Phys. **227**, 7889 (2008).
[28]E. Deriaz, S. Peirani, and M. Mod. Simu, SIAM Comput. J. **16**, 583 (2018).
[29]E. Madaule, "Schemas Numériques Adaptatifs Pour Les Equations de Vlasov-Poisson," Ph.D. thesis, Université de Lorraine, 2016. https://theses.hal.science/tel-01446399v1.
[30]U. Ganse, T. Koskela, M. Battarbee, Y. Pfau-Kempf, K. Papadakis, M. Alho, M. Bussov, G. Cozzani, M. Dubart, H. George, E. Gordeev, M. Grandin, K. Horaites, J. Suni, V. Tarvus, F. Tesema Kebede, L. Turc, H. Zhou, and M. Palmroth, Phys. Plasmas **30**, 042902 (2023).
[31]L. Kotipalo, M. Battarbee1, Y. Pfau-Kempf, and M. Palmroth, Geosci. Model Dev. **17**, 6401 (2024).
[32]B. Svedung Wettervik, E. Siminos, and T. Fülöp, Eur. Phys. J. D **71**, 157 (2017).
[33]T. Pollinger, J. Rentrop, D. Pflüger, and K. Kormann, J. Comput. Phys. **491**, 112338 (2023).
[34]J. A. F. Hittinger and J. W. Banks, J. Comput. Phys. **241**, 118 (2013).
[35]K. Fujimoto and S. Machida, J. Comput. Phys. **214**, 550 (2006).
[36]K. Fujimoto, J. Comput. Phys. **230**, 8508 (2011).
[37]A. Ghizzo, D. Del Sarto, and H. Betar, Phys. Plasmas **31**, 072109 (2024).
[38]B. D. Fried, Phys. Fluids **2**, 337 (1959).
[39]A. Bret, M. E. Dieckmann, and C. Deutsch, Phys. Plasmas **13**, 082109 (2006).
[40]L. Gremillet, D. Benisti, E. Lefebvre, and A. Bret, Phys. Plasmas **14**, 040704 (2007).
[41]M. V. Medvedev and A. Loeb, Astrophys. J. **526**, 697 (1999).
[42]G. F. Swadling, C. Bruulsema, F. Fiuza, D. P. Higginson, C. M. Huntington, H. S. Park, B. B. Pollock, W. Rozmus, H. G. Rinderknecht, J. Katz, A. Birkel, and J. S. Ross, Phys. Rev. Lett. **124**, 215001 (2020).
[43]F. Huot, A. Ghizzo, P. Bertrand, E. Sonnendrucker, and O. Coulaud, J. Comput. Phys. **185**, 512 (2003).
[44]M. Antoine, A. Ghizzo, D. Del Sarto, and E. Deriaz, J. Plasma Phys. (submitted) (2024).
[45]N. Crouseilles, G. Latu, and E. Sonnendrucker, Int. J. Appl. Math. Comput. Sci. **17**, 335 (2007).
[46]J. Boris, "Relativistic plasma simulation-optimization of a hybrid code," in *Proceeding of Fourth Conference on Numerical Simulations of Plasmas* (Defense Technical Information Center, 1970).
[47]H. Cheng and K. C. Gupta, J. Appl. Mech. **56**, 139 (1989).
[48]C. Geronimi, F. Bouchut, M. R. Feix, H. Ghalila, M. Valentini, and J. M. Buzzi, Eur. J. Phys. **18**, 102–107 (1997).
[49]J. Villasenor and O. Buneman, Comput. Phys. Commun. **69**, 306–316 (1992).
[50]A. B. Langdon, Comput. Phys. Commun. **70**, 447–450 (1992).
[51]T. Zh. Esirkepov, Comput. Phys. Commun. **135**, 144–153 (2001).
[52]E. S. Weibel, Phys. Rev. Lett. **2**, 83 (1959).

[53]R. A. Fonseca, L. O. Silva, J. W. Tonge, W. B. Mori, and J. M. Dawson, Phys. Plasmas **10**, 1979 (2003).
[54]G. Shvets, O. Polomarov, V. Khudik, C. Siemon, and I. Kaganovich, Phys. Plasmas **16**, 056303 (2009).
[55]V. Y. Martyanov, V. V. Kocharovsky, and V. V. Kocharovsky, Astron. Lett. **36**, 396 (2010).
[56]G. Raj, O. Kononenko, M. F. Gilljohann, A. Doche, X. Davoine, C. Caizergues, Y. Y. Chang, J. P. Couperus Cabadag, A. Debus *et al.*, Am. Phys. Soc. **2**, 023123 (2020).
[57]A. Ghizzo, D. Del Sarto, and M. Sarrat, Phys. Plasmas **27**, 072103 (2020).
[58]D. Del Sarto, A. Ghizzo, and M. Sarrat, Phys. Plasmas **31**, 072113 (2024).
[59]A. Bret, L. Gremillet, and D. Bénisti, Phys. Rev. E **81**, 036402 (2010).
[60]A. Ghizzo and D. D. Sarto, Plas. Phys. Controlled Fus. **63**, 055007 (2021).
[61]K. Papadakis, Y. Pfau-Kempf, U. Ganse, M. Battarbee, M. Alho, M. Grandin, M. Dubart, L. Turc, H. Zhou, K. Horaites, I. Zaitsev, G. Cozzani, M. Bussov, E. Gordeev, F. Tesema, H. George, J. Suni, V. Tarvus, and M. Palmroth, Geosci. Model Dev. **15**, 7903 (2022).