Efficient Semi-Lagrangian Vlasov-Maxwell Simulations of High Order Harmonic Generation from Relativistic Laser-Plasma Interactions

Götz Lehmann

Institut für Theoretische Physik I, Heinrich-Heine Universität, 40225 Düsseldorf, Germany.

Received 4 December 2015; Accepted (in revised version) 23 February 2016

Abstract. We describe a relativistic semi-Lagrangian scheme for the numerical solution of the relativistic Vlasov-Maxwell system. The implementation strategy on a modern non-unified memory access (NUMA) architecture using the OpenMP framework is discussed. We demonstrated that close to perfect scaling can be obtained on modern many-core, multi-socket systems. Application of this code to the problem of relativistic generation of high-harmonic laser radiation is demonstrated. The results are compared to particle-in-cell (PIC) simulations, indicating in particular that for warm plasma the Vlasov simulation is superior. We discuss the impact of plasma temperature on the radiation spectrum and show that the efficiency of harmonic generation depends on the plasma temperature.

AMS subject classifications: 65Z05, 65M25, 70B05
PACS: 52.65.Ff, 52.27.Ny, 52.59.Ye
Key words: Relativistic Vlasov-Maxwell system, semi-Lagrange, multi-threading, relativistic laser-plasma interaction, high-harmonic radiation generation.

1 Introduction

The maximum achievable laser intensity has made significant progress in the last decade, allowing intensities on the order of $10^{19} - 10^{22} \text{W/cm}^2$. At the same time the produced pulses are very short, typically on the order of some 10 fs or even below. The interaction of such intense laser pulses with plasmas exhibits complex phenomena, many of which have their origin in the relativistic mass variation that charged particles experience in these fields [1].

One of the most widely used set of equations to model relativistic laser-plasma interaction are the Vlasov-Maxwell equations. This system describes the interaction between
a relativistically strong laser and a warm, collisionless plasma. In general collisions between particles tend to become negligible at high laser intensities, thus this set of equations describes a major part of laser-plasma interaction scenarios studied today.

Numerical solution of the Vlasov-Maxwell equations is mostly done via particle-in-cell (PIC) methods. These methods cluster a large number of real particles into artificial macro-particles and by doing so introduce a coarse-graininess of the plasma. The macro-particle trajectories follow the characteristics of the Vlasov equation, while the electromagnetic fields are calculated on a grid. By gathering the charge and current densities on the grid, the fields are updated. Typically the introduction of macro-particles introduces noise to the simulation results, with the noise level depending on the ratio of real to artificial particles. Since the works of Birdsall and Langdon [2], and Hockney and Eastwood [3] much effort has been dedicated to improving the noise level of these methods and there are now massively parallel implementations available that treat over $10^{10}$ macro-particles in full three-dimensional geometry (i.e. six-dimensional phase-space). Such codes are widely used today to study many phenomena in the field of laser-plasma interaction [1]. Nevertheless, full three-dimensional simulations of systems with large spatial extension still require considerable computational resources.

For applications where particles in the tail of the distribution function play a role, or where noise is an issue, direct solution of the Vlasov equation may give better results than PIC methods. These direct simulations of the advection of the particle distribution function in phase-space, without introducing an artificial graininess of the plasma, yields virtually noise-free results. Over the last ten years Vlasov solvers have been developed in the context of relativistic laser-plasma interaction [4–11].

The classical methods for the non-relativistic Vlasov equation, starting with [12], use a time splitting scheme to update the phase-space distribution on a Eulerian grid. Later works formulated finite volume schemes and combined them with flux limiting methods to guarantee the positivity of the distribution function, see e.g. [13]. For the relativistic case it was however shown that time splitting schemes introduce a numerical instability [6] and that semi-Lagrangian schemes are preferable. Semi-Lagrangian schemes have a long history in the treatment of advective problems, especially in climate and weather simulations [14]. These methods aim to combine an Eulerian and a Lagrangian fluid description approach. In the Eulerian picture, all quantities evolve over time on a fixed grid in space. This leads to limitations in the size of the time-step due to stability considerations. In the Lagrangian picture, the observer moves along with a certain fluid element, thus the quantity stays constant, but the position of the element changes. Usually, this allows for larger time-steps. However, an initially uniformly spaced grid usually evolves into a non-equidistant distribution of grid points.

Semi-Lagrangian schemes have been used previously in the context of Vlasov-Maxwell systems to study different aspects of laser-plasma interaction, such as for example wakefield acceleration [8], ion-acceleration [15], relativistic modulational instabilities [5] or self-induced-transparency [7, 16].
Relativistically strong laser fields can accelerate particles up to velocities many orders of magnitude larger than the thermal velocity. This is different from non-relativistic cases, where velocities might be on the same order of magnitude as the thermal velocity. Thus, for the interaction with very strong fields it is often necessary to simulate very large phase-space intervals with large extends along the momentum axis in order to capture the complete dynamics. At the same time a huge part of the phase-space might be unpopulated. In these cases a certain level of adaptivity would be beneficial (e.g. wavelets, see [17]), however their computational additional effort does not yet result in an overall reduction in runtime.

The aim of this work is to present an efficient OpenMP multi-threaded implementation of a semi-Lagrangian scheme. Modern distributed shared memory computers easily have tens of cores, specialized hardware up to hundreds. For data-parallel algorithms it is a challenge to efficiently utilize this great number of cores, given that the memory bandwidth usually does not scale proportionally to the number of cores present. We demonstrate that very good scaling performance is possible on a 32 core system. Even though the algorithm has many parts which are inherently suited for parallelization, one has to take into account details to obtain a good scaling behaviour. The OpenMP framework is used to introduce multi-threading in our application, both because of its ease of use and because of its portability.

Within this work, we will consider a two-dimensional phase-space, which already allows us to treat many interesting and relevant problems in relativistic laser-plasma interaction. Conclusions for the implementation of higher-dimensional simulation codes are possible. Usually higher-dimensional problems may require more CPU resources and memory than is present in one machine. Higher-dimensional codes will thus probably run in a hybrid MPI/OpenMP environment, where the OpenMP part might be similar to the implementation presented here.

To validate the code performance, we present results from simulations related to the generation of high-harmonic radiation from the relativistic interaction between a ultra-short laser pulse and overdense plasma. We compare the results obtained from our code to results from PIC simulations and show that in general we obtain good agreement between both, but the Vlasov simulation is superior in terms of noise. Furthermore, we demonstrate that the temperature of the overdense plasma will affect the efficiency by which higher harmonics of the incoming radiation are generated. This is especially important in the view that PIC results typically become more noisy for larger plasma temperatures and most simulations in this context so far assume initially cold plasma.

The manuscript is organized as follows. In Section 2 we formulate the relativistic Vlasov-Maxwell model to clarify notation. Section 3 gives a short review of the semi-Lagrangian numerical scheme we employ to solve the model equations. In Section 4 we describe our implementation strategy. The code performance is then summarized in Section 5. Section 6 discusses simulation results for the example of generation of harmonic radiation from overdense plasmas. The manuscript concludes with a short summary.
2 Model equations

The relativistic Maxwell-Vlasov system can be used to describe the interaction between a relativistically intense laser with a plasma. Vlasov’s equations for electrons and ions are coupled via the full set of Maxwell’s equations. The electric field is composed of the electrostatic contribution due to charge separation, and the electromagnetic part due to the laser fields.

The model will be limited to one-dimensional (1D) geometry. All quantities will spatially only depend on the coordinate \( x \) along the laser pulse propagation direction. This approach is justified as long as the typical scale-length for parallel variations is much smaller than the scale-length of transversal variations.

We begin with the relativistic formulation of Vlasov’s equation for the distribution functions \( F_{e,i} \) and \( F_{e,i} \), Maxwell’s equations for \( E \) and \( B \), and Poisson’s equation for the scalar electrostatic potential \( \phi \),

\[
\frac{\partial F_{e,i}}{\partial t} + \frac{p_{e,i}}{m_{e,i} \gamma_{e,i}} \cdot \frac{\partial F_{e,i}}{\partial \mathbf{r}} + q_{e,i} \left( E + \frac{1}{c} \frac{p_{e,i}}{m_{e,i} \gamma_{e,i}} \times \mathbf{B} \right) \cdot \frac{\partial F_{e,i}}{\partial p_{e,i}} = 0, \tag{2.1}
\]

\[
\frac{1}{c} \frac{\partial E}{\partial t} = \nabla \times \mathbf{B} - \frac{4\pi}{c} \mathbf{j}, \tag{2.2}
\]

\[
- \frac{1}{c} \frac{\partial \mathbf{B}}{\partial t} = \nabla \times \mathbf{E}, \tag{2.3}
\]

\[
\nabla^2 \phi = - \frac{4\pi}{c} \rho, \tag{2.4}
\]

where \( \rho = q_{i,n_i} + q_{e,n_e} \), \( \mathbf{j} = q_{i,n_i} \mathbf{v}_i + q_{e,n_e} \mathbf{v}_e \) and \( p_{e,i} = m_{e,i} \gamma_{e,i} \mathbf{v}_{e,i} \). The relativistic Lorentz factor is \( \gamma_{e,i} = \sqrt{1+p_{e,i}^2/(m_{e,i}c)^2} \). The subscripts \( e,i \) denote electrons or ions, respectively.

The electric and the magnetic fields can be expressed via the potentials

\[
E = -\nabla \phi - \frac{1}{c} \frac{\partial \mathbf{A}}{\partial t}, \tag{2.5}
\]

\[
B = \nabla \times \mathbf{A}. \tag{2.6}
\]

The 1D geometry allows us to make use of the conservation of transverse canonical momentum \( p_{\perp e,i} = p_{\perp e,i} \pm \left( q_{e,i}/c \right) A_\perp \) and choose \( p_{\perp e,i} = 0 \). Thus, \( p_{\perp e,i} = \mp \left( q_{e,i}/c \right) A_\perp \), where \( p_{\perp e,i} \) is the perpendicular momentum of a particle and \( A_\perp \) is the transverse vector potential of the electromagnetic wave. We suppose the wave propagates along the \( x \) direction and is constant in the perpendicular direction. The particle distribution function \( F_{e,i}(\mathbf{r}, p_{e,i}, t) \) is then of the form

\[
F_{e,i}(\mathbf{r}(t), p_{e,i}(t), t) = f_{e,i}[x(t), (p_{e,i}(t)), \delta(p_{\perp e,i}(t) \pm (q_{e,i}/c)A_\perp(t))]. \tag{2.7}
\]

In order to follow the dynamics of the fluid populating the phase-space, we have to solve for \( f_{e,i}(x, p_{e,i}, t) \). From here on we write \( p \equiv p_x \) in order to simplify the notation and intro-
duce dimensionless values by substituting

\[
\begin{align*}
\omega_{pe}t & \rightarrow t, \\
\frac{\omega}{\omega_{pe}} & \rightarrow \omega, \\
\frac{e}{\omega_{pe}m_\text{e}c}E & \rightarrow E, \\
\frac{\epsilon \phi}{m_\text{e}c^2} & \rightarrow \phi, \\
\frac{p_\text{e}}{m_\text{e}c} & \rightarrow p_\text{e}, \\
\frac{q_e}{e} & \rightarrow q_e, \\
\end{align*}
\]

where \( \omega_{pe} = \sqrt{4\pi n_0 e^2/m_\text{e}} \) is the plasma frequency of the plasma of density \( n_0 \). In these dimensionless units Vlasov’s equation becomes

\[
\frac{\partial f_{e,i}}{\partial t} + \epsilon_{e,i} \frac{p_{e,i}}{\gamma_{e,i}} \frac{\partial f_{e,i}}{\partial x} + \left( \mp E_x - \frac{\epsilon_{e,i}}{2\gamma_{e,i}} \frac{\partial |A_\parallel|^2}{\partial x} \right) \frac{\partial f_{e,i}}{\partial p_{e,i}} = 0, \tag{2.8}
\]

where \( \epsilon_{e,i} = m_\text{e}/m_{e,i} \). The Lorentz \( \gamma \)-factors can now be expressed as

\[
\gamma_{e,i} = \sqrt{1 + \epsilon_{e,i}^2 \left( p_{e,i}^2 + |A_\parallel|^2 \right)}. \tag{2.9}
\]

The species densities and currents are obtained via

\[
\begin{align*}
n_{e,i}(x) &= \epsilon_{e,i} \int f(x,p_{e,i}) \, dp_{e,i}, \tag{2.10} \\
j_{xe,i} &= \mp \epsilon_{e,i}^2 \int f(x,p_{e,i}) \frac{p_{e,i}}{\gamma_{e,i}} \, dp_{e,i}, \tag{2.11} \\
j_{\perp e,i} &= \mp \epsilon_{e,i}^2 A_\parallel \int f(x,p_{e,i}) \, dp_{e,i}. \tag{2.12}
\end{align*}
\]

The perpendicular electromagnetic fields enter the Vlasov equation (2.8) only in form of the transverse vector potential. In the Coulomb gauge it follows from \( \nabla \cdot A = 0 \) that the vector potential in 1D geometry is purely transverse, i.e. \( A_x = 0 \). The wave-equation for the transverse components of the vector potential is

\[
\frac{\partial^2 A_\parallel}{\partial t^2} - \frac{\partial^2 A_\parallel}{\partial x^2} = j_{\perp e} + j_{\perp i}. \tag{2.13}
\]
Eq. (2.13) (in combination with expressions (2.5) and (2.6)) is equivalent to the dimensionless form of Maxwell’s equations in 1D,
\[
\frac{\partial}{\partial t} \begin{pmatrix}
    E_y \\
    E_z
\end{pmatrix} = \begin{pmatrix}
    -\partial_x B_z - j_y \\
    -\partial_x B_y - j_z
\end{pmatrix},
\]
(2.14)
\[
\frac{\partial}{\partial t} \begin{pmatrix}
    B_y \\
    B_z
\end{pmatrix} = \begin{pmatrix}
    -\partial_x E_z \\
    -\partial_x B_y
\end{pmatrix}.
\]
(2.15)

The scalar potential \( \phi \) is determined by Poisson’s equation
\[
\frac{\partial^2 \phi}{\partial x^2} = n_e - n_i.
\]
(2.16)

As \( A_x = 0 \), the longitudinal electric field \( E_x \) depends only on the charge distribution \( \rho \), i.e.
\[
E_x = -\frac{\partial \phi}{\partial x},
\]
(2.17)

and for the transverse components we have
\[
\frac{\partial A_\perp}{\partial t} = -E_\perp.
\]
(2.18)

Finally, we remark that solving Ampère’s law
\[
\frac{\partial E_x}{\partial t} = -(j_{xi} + j_{xe})
\]
(2.19)
is equivalent to solving Poisson’s equation (2.16).

3 Numerical scheme

We make use of a semi-Lagrangian scheme to solve the one-dimensional Maxwell-Vlasov model, consisting of Eqs. (2.8)-(2.12), (2.14)-(2.15) and (2.17)-(2.19). Our numerical scheme is in most parts based on the work of Sonnendrücker [4] and Shoucri [18].

Let us start by formulating the semi-Lagrange concept for the solution of the Vlasov equation. The distribution function \( f \) is represented on a grid which is equidistantly spaced, covering a certain range in \( x \) and an interval of momenta \( p_x \). In order to determine the value of the distribution function \( f^{n+1}(x_j,p_k) \), \( x_j = j\Delta x + x_{\text{min}}, p_k = k\Delta p + p_{\text{min}} \) at time \( t = (n+1)\Delta t \), we follow the phase-space characteristic backwards in time to \( t = n\Delta t \).

Let the coordinates of the origin of this trajectory be \((\tilde{x}_j, \tilde{p}_k)\). After determination of this origin, we have to interpolate the distribution function \( f^n \) at the position \((\tilde{x}_j, \tilde{p}_k)\). The value obtained from this interpolation will then be assigned to \( f^{n+1}(x_j,p_k) \). For the interpolation we make use of 3rd order B-splines, which allow us to break the 2D interpolation problem down into 1D interpolations. The details of this will be discussed in detail later.

Let us note, that by using the scheme outlined above, we avoid problems that will arise in flux-splitting algorithms for relativistic Vlasov-Maxwell systems [6].
3.1 Characteristic curves

First, let us summarize how to determine the origin of the characteristic in the $(x, p_x)$ plane on which the distribution function stays constant. Eq. (2.8) is of the form

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial f}{\partial p} \frac{\partial p}{\partial t} = 0$$

and describes the evolution of the distribution function $f(r, t)$, where $r(t) = (x(t), p(t))$. This implies that along the characteristic curve

$$\frac{dr}{dt} = U(r, t), \quad (3.1)$$

where

$$U(r, t) = \begin{pmatrix} \epsilon_{ij} \frac{p_{ij}}{s_{ij}} & -E_x - 2s_{ij} \frac{\partial |A_\perp|^2}{\partial x} \end{pmatrix}, \quad (3.2)$$

the value of $f(r, t)$ stays constant.

For trajectories determined by Eq. (3.1) we note that

$$f(r(t + \Delta t), t + \Delta t) = f(r(t - \Delta t), t - \Delta t).$$

Writing $r(n\Delta t) = r^n$, by discretization of Eq. (3.1) we get, to second order accuracy,

$$\frac{r^{n+1} - r^n}{2\Delta t} = U^n(r^n). \quad (3.3)$$

In our simulation, we represent values of $f$ on a discretized phase-space $r^n_{jk} = (x^n_j, p^n_k)$, where $x^n_j = x(j\Delta x + x_{min}, n\Delta t)$, $p^n_k = p(k\Delta p + p_{min}, n\Delta t)$. Since the value of $f$ is constant along the characteristic curve, we determine the value of $f$ at a given position $r^n_{jk}$ by calculating the origin $r^{n-1} = (x_0, p_0)$ of the trajectory via Eq. (3.1). For this, we substitute on the right-hand side of Eq. (3.3)

$$r^n = \frac{r^{n+1} + r^{n-1}}{2}.$$

Introducing $dr = r^{n+1} - r^n$ and writing $r^{n-1} = r^{n+1} - 2dr$, Eq. (3.3) becomes

$$dr = \Delta t U^n(r^{n+1}_{ij} - dr). \quad (3.4)$$

This implicit equation for $dr$ can be solved by an iterative scheme, starting with $dr = 0$. Within these iterations we have to interpolate the electrostatic field $E_x$ and the transverse vector potential $A_\perp$. We use linear interpolations for this step, which was found to be sufficient in similar schemes [14, 19].
Once the origin of the trajectory is known, we calculate the value of \( f \) at the position \( r_{jk}^{n+1} \),
\[
f^{n+1}(r_{jk}^{n+1}) = f^{n-1}(r_{jk}^{n+1} - 2\mathbf{dr}).
\]
The value of \( f \) at \( r = r_{ij}^{n+1} - 2\mathbf{dr} \) has to be obtained by interpolation, since the origin of the trajectory does not usually coincide with a position on the grid.

### 3.2 Field integration

In order to update the electric fields, we determine the densities \( n_{e,j}^{n+1} \) and currents \( j_{i,j}^{n+1} \) via Eqs. (2.10)-(2.12). Rather than using Eq. (2.13) for the transverse field components, we use Eqs. (2.14)-(2.15) and (2.18). Eventhough this means we have to solve five 1D equations instead of two, we can now make use of a staggered Yee mesh for \( \mathbf{E}, \mathbf{B} \) which allows for simple treatment of emitting and absorbing boundary conditions. The field updates only consume a very tiny amount of the total computational time, so the additional overhead has little impact on the overall runtime.

Using staggered meshes in time and space for \( \mathbf{E}_\perp \) and \( \mathbf{B}_\perp \) results in the scheme

\[
\begin{align*}
\frac{E_{y,j}^{n+1/2} - E_{y,j}^{n-1/2}}{\Delta t} &= \frac{B_{z,j+1/2}^{n} - B_{z,j-1/2}^{n}}{\Delta x} - j_{y,j}^{n}, & (3.5) \\
\frac{E_{z,j}^{n+1/2} - E_{z,j}^{n-1/2}}{\Delta t} &= \frac{B_{y,j+1/2}^{n} - B_{y,j-1/2}^{n}}{\Delta x} - j_{z,j}^{n}, & (3.6) \\
\frac{B_{y,j}^{n+1/2} - B_{y,j}^{n-1/2}}{\Delta t} &= \frac{E_{x,j+1/2}^{n} - E_{x,j-1/2}^{n}}{\Delta x}, & (3.7) \\
\frac{B_{z,j}^{n+1/2} - B_{z,j}^{n-1/2}}{\Delta t} &= \frac{E_{y,j+1/2}^{n} - E_{y,j-1/2}^{n}}{\Delta x}. & (3.8)
\end{align*}
\]

To update the electrostatic electric field \( E_x \) we make use of Eq. (2.19),
\[
\frac{E_{x,j}^{n+1/2} - E_{x,j}^{n-1/2}}{\Delta t} = -j_{x,j}^{n}. & (3.9)
\]

In order to update the distribution functions we need the parallel electric field at time \( n\Delta t \), which we approximate as \( E_{x,j}^{n} = (E_{x,j}^{n+1/2} + E_{x,j}^{n-1/2})/2 \).

From Eq. (2.18) we obtain the vector potential at time \( (n+1)\Delta t \),
\[
\frac{A_{\perp,j}^{n+1} - A_{\perp,j}^{n}}{\Delta t} = -E_{\perp,j}^{n+1/2}. & (3.10)
\]

The time-step is \( \Delta t = \Delta x \) for all our simulations. This is the *magic* time-step for which the numerical field-integration becomes analytically exact (given that the sources are exact). There is no numerical dispersion due to the numerical integration scheme for the electromagnetic fields, and perfectly absorbing boundary conditions are simple to implement.
3.3 Interpolation

Let us now come back to the interpolation problem we need to solve in order to obtain the value of \( f_{n+1}(r^{n+1}) \). Following earlier works [4, 15, 20], we make use of 2D cubic B-spline interpolation. This has the advantage that the two-dimensional interpolating spline is obtained as a tensor product of one-dimensional splines of the form

\[
s(x, p) = \sum_{\mu=-3}^{N_x-2} \sum_{\nu=-3}^{N_p-2} \eta_{\mu\nu} B_{\mu}(x) B_{\nu}(p).
\]

Here, the functions

\[
B_{\nu}(x) = \frac{1}{6h^3} \begin{cases} 
(x-x_{\nu})^3, & x_{\nu} \leq x < x_{\nu+1}, \\
h^3 + 3(h^2(x-x_{\nu+1}) + h(x-x_{\nu+1})^2 - (x-x_{\nu+1})^3), & x_{\nu+1} \leq x < x_{\nu+2}, \\
h^3 + 3(h^2(x_{\nu+3}-x) + h(x_{\nu+3}-x)^2 - (x_{\nu+3}-x)^3), & x_{\nu+2} \leq x < x_{\nu+3}, \\
(x_{\nu+4}-x)^3, & x_{\nu+3} \leq x < x_{\nu+4}
\end{cases}
\]

are cubic polynomials which are only non-zero within four intervals \([B_{\nu}, B_{\nu+4}]\) of the mesh. Details about the cubic B-splines can be found in Ref. [21].

Here, we will only reproduce the steps necessary to obtain the coefficients \( \eta_{\mu\nu} \), because this is one of the steps we can speed up by parallelization. First, we solve the \( N_p \) independent 1D interpolation problems

\[
s(x_i, p_k) = \sum_{\mu=-3}^{N_x-2} \theta_{\mu k} B_{\mu}(x_i), \quad \text{for} \quad i = 1, \ldots, N_x,
\]

with \( k = 1, \ldots, N_p \) constant. From here we obtain \( N_x + 2 \) values \( \theta_{\mu k} \) for each \( k \).

Next, we interpolate in \( p \) direction, solving the \( N_x + 2 \) (i.e. \( \mu = -3, \ldots, N_x - 2 \)) sets of linear equations

\[
\theta_{\mu k} = \sum_{\nu=-3}^{N_p-2} \eta_{\mu\nu} B_{\nu}(p_k)
\]

of dimension \( N_p + 2 \). Once again these sets of linear equations are independent of each other.

After determination of the \( \eta_{\mu\nu} \), we can evaluate the spline \( s(x, p) \) following Eq. (3.11). We note that for each position only sixteen basis functions \( B_{\nu} \) carry non-zero coefficients.

4 Implementation

The code is written in C++ and multi-threaded using the OpenMP framework. OpenMP provides efficient data-parallel parallelization constructs which we utilize in many subroutines of our code.
Most of the computational load is located in the update of the distribution functions, following the scheme outlined above. First, we have to calculate for each grid point of the discretized phase-space the origin of the trajectory which will end at this point (i.e. a for-loop over all grid points). This loop can be perfectly parallelized, since finding the origin of the trajectory is an independent problem for each position on the grid. It is important to note that this calculation involves a fixed-point iteration to solve the nonlinear Eq. (3.1). Usually about three iterations are sufficient to converge, but the number of required iterations may change from point to point. Thus, the computational effort for each grid point is not constant. In order to maximize the efficient use of all processor cores, we do not assign the same number of grid points to each core. Instead, we make use of the dynamic scheduling provided by OpenMP for parallelizing for-loops. Initially we only distribute a fraction of the total problem (i.e. a certain interval of the for-loop) amongst the threads (in this case each thread starts with the same number of grid points). Subsequently, the OpenMP scheduler will then distribute further intervals of the loop to the threads which finished their recent task. The size of these further intervals can either be fixed or vary in size. By using a `omp parallel for schedule(guided)` clause, we let OpenMP decide the size of the intervals to be distributed amongst the threads. In this way we are able to benefit from many cores and achieve a speedup scaling well with the number of processor cores, even though the computational effort for each loop iteration varies.

Once the origins of the trajectories reaching the grid points at the time \((n+1)\Delta t\) are known, we have to interpolate the distribution function \(f^{n-1}\) at these positions. Thus, we have to calculate the spline coefficients \(\eta_{\mu\nu}\). As outlined above, this is a two stage process, requiring two times solving sets of independent systems of linear equations. In step one, we have to solve \(N_p\) systems of dimension \(N_x+2\). Each of these systems has the same coefficient matrix, only the right-hand sides differ. The coefficient matrix is tridiagonal with a \(2 \times 2\) block coupled to it [4]. Since all systems of the set deal with the same matrix, we initially precompute the LU decomposition of the coefficient matrix. In the second stage, we deal with \(N_x+2\) systems of \(N_p+2\) linear equations each. Again, all the systems share the same coefficient matrix of the same form as for the systems in stage one. We precalculate the LU decomposition of this matrix as well.

In both stages we can parallelize the work load efficiently, since each system of linear equations within the sets is independent of the others. The computational effort to solve each single system is constant, hence we evenly distribute the work load amongst the different threads.

The last step in the update of the distribution function is the evaluation of the B-spline (3.11). This is also an independent problem for each grid point and the computational effort is the same for each evaluation. Hence, we parallelize this in a static way, assigning each thread the same number of interpolations to be carried out.

Updates of the species densities (2.10) and the currents (2.11), (2.12) are calculated following the updates of the distribution functions. Consider our reduction strategy in the updates of these quantities using the density as an example. We store the distribution functions in row-major order, the coordinate \(x\) changes along the row of the matrix, while
\( p \) changes along the column. This layout has to be considered when devising an efficient way to carry out the reduction in \( p \) direction. It is our goal to calculate

\[
    n_j = \Delta p \sum_{k=1}^{N_p} f_{kj},
\]

for \( j = 1, \ldots, N_x \). We could divide the interval for \( j \) in equally large sub-intervals, for which then each thread calculates \( n_j \) within its assigned interval. This would allow us to share the array which stores all \( n_j \) amongst the threads, where each thread would update only the part of this array which it was assigned to. However, to calculate \( n_j \) each thread would have to traverse the matrix storing \( f \) in \( p \) direction, that is along the column, i.e. we would have non-contiguous memory access. This heavily affects the cache efficiency and substantially degrades parallel performance. The efficient use of CPU cache is of great importance since the total memory bandwidth of the system does not scale as the number of cores available in the system.

In order to have contiguous memory access, we share the work not by splitting the problem (4.1) in \( j \), but in \( k \) direction. Each thread \( m = 1, \ldots, N_{\text{threads}} \) now first calculates an intermediate density \( n_j^m \) based on an interval \( k = k_{\text{min}}^m, \ldots, k_{\text{max}}^m \). This requires each thread to have its own temporary array to store \( n_j^m \). Once all intermediate results have been calculated, the final sum \( n_j = \sum_{m=1}^{N_{\text{threads}}} n_j^m \) is generated. The array storing \( n_j \) is shared amongst the threads, thus we have to prevent race conditions when performing the final sum. To circumvent this, each thread adds its contribution to the total density within an \texttt{omp critical} section to the shared array \( n \). The \texttt{critical} statement ensures, that only one thread at a time is adding its value \( n_j^m \) to \( n_j \). The use of \texttt{critical} sections of course degrades the overall parallelism, since it forces the threads to act sequentially in this section. However, in the end this has very little impact on the overall runtime as the speedup due to the sequential memory access is larger by orders of magnitude.

To achieve good multi-threading efficiency beyond a single CPU, on a multi-processor machine, the memory access pattern of the code has to reflect the NUMA (non-unified memory access) architecture of current multi-socket machines. In a NUMA environment each CPU has local memory, which is then shared with the other CPUs via a common logical address space. Accessing data that is non-local to the CPU will be slower compared to local access. OpenMP operates on the logical address space and is unaware of the underlying physical memory layout. Let us consider again the reduction operation which calculates \( n_j \) from \( f_{kj} \). Each thread \( m \) needs to sum up the rows \( k_{\text{min}}^m \) to \( k_{\text{max}}^m \) of the matrix which holds \( f \). The memory in which these rows are stored should be local to the CPU on which the according thread is executed. To achieve this, we make use of the first-touch policy of the operating system. The mapping of logical to physical memory addresses is done at the first write-access to the address. Thus, when we initialize memory to hold i.e. a matrix, we at first write zeros into all entries using an OpenMP multi-threaded for-loop with exactly the same scheduling as we will use later to read data from this memory. In this way we make sure that all memory is mostly local to the CPUs which will execute
the threads. During the execution of the code we have to prevent thread-migration between different CPUs, which would spoil data locality. To this end we use the LIKWID toolkit [22].

5 Performance

A typical problem requires dimensions of about $N_x = 15k$ to $30k$ mesh points in parallel direction and between $N_p = 2k$ and $N_p = 20k$ mesh points for the momentum coordinate. Possible applications for the code are the simulation of laser pulse amplification via stimulated Raman and Brillouin backscattering [23–25] or the generation of high harmonic radiation (see Section 6).

We measured code performance on a quad-CPU system, consisting of four Intel Xeon E4820 CPUs operating at 2GHz. Each CPU has eight cores, which yields a total of 32 physical cores in the machine. In Fig. 1 we show results of a strong-scaling benchmark for simulations discussed in Section 6. The dimensions are $(N_x = 20k) \times (N_p = 5k)$. We measured the wall clock time for different numbers of threads. The speedup factor is defined as the ratio of time it takes to execute with $N$ threads compared to one thread.

We find almost perfect scaling behavior, with close to 95% percent efficiency when using all 32 cores (speedup of 30.3 when compared to one core). Note that this is the speedup taking into account the whole simulation, not only of the multi-threaded part. The electric and magnetic field updates for instance are sequential code, since they only require simple operations on arrays of dimension $N_x$ and overhead for threading outweighs the benefit. For larger problem dimensions we find equivalently good strong-scaling behavior.

![Figure 1](image_url)  

Figure 1: (color online) The relative speedup in runtime is plotted versus the number of threads used. Shown are results for a problem size of $(N_x = 20k) \times (N_p = 5k)$ resolution. The dashed black line indicates perfect scaling behavior.
Soft-scaling was also checked, i.e. increasing the problem size but increasing the number of threads at the same time. The total amount of work for each thread thus stays constant. Starting from the smallest problem with dimensions $4k \times 1k$ and one thread, going up to 32 threads and a problem with dimensions $16k \times 8k$ we observe only little deviation from perfect scaling (about 5% increase in runtime).

To measure the absolute performance, we simulated the problem of ion acceleration treated by Shourci et al. [15]. In their paper they reported a runtime of about four weeks for their specific problem related to ion acceleration. Their Vlasov code is very similar in terms of algorithms, but is a purely serial code. By running our code in sequential mode, we estimated a comparable runtime of about 40 days on our machine. Using 64 threads (i.e. enabling hyper-threading) on our machine we can reduce the total runtime down to 29 hours. This corresponds to a speedup of about a factor of 33, i.e. close to the actual number of physical cores present in the machine. This is massive reduction of runtime and opens up the possibility to use high-resolution Vlasov simulations on daily basis.

In the Section 6 the code will be used to study aspects of the interaction between warm plasma and relativistically intense laser radiation, however, it is not limited to this scenario. The correctness of the code has been verified through simulations of linear stages of instabilities such as Raman and Brillouin scattering. To verify nonlinear dynamics, the ion acceleration simulations from Refs. [15, 26] have been repeated, confirming the reported results, obtained with a very similar semi-Lagrangian Vlasov scheme. On the other hand the present code has been used in the context of plasma-based laser pulse amplification via Raman and Brillouin scattering, respectively [23, 24, 27]. Furthermore, the code has been tested against a PIC code in a study related to the motion of the electron plasma boundary in front of an overdense target which is subject to incident relativistic laser radiation [28].

6 Relativistic generation of higher harmonics laser radiation

One feature of the interaction between a relativistically intense laser pulse and a overdense plasma is the generation of higher harmonics of the incoming laser field. The ponderomotive force exerted on the plasma surface will enforce anharmonic oscillations of the electrons close to the surface. As a consequence the reflected radiation will be composed out of radiation with frequencies multiple of the incident frequency $\omega_0$. Gaseous targets also provide mechanisms for the generation of high order harmonics, but based on analytical models and numerical simulations, it is expected that the interaction with overdense plasma is a far more efficient process [29, 30]. A great number of experimental studies have been carried out over the last decade, using pico- to femto-second laser pulses (see Refs. [31, 32] and references therein). Provided the efficiency predicted by theory can be accomplished experimentally, high harmonic generation from overdense plasmas could compete in certain parameter regimes with other sources of ultrashort extreme-ultraviolet (XUV) radiation, such as free-electron lasers.
The complex electron dynamics at the surface of the overdense plasma has been subject to many detailed PIC simulations (see e.g. [33–35]), while the analytical understanding of the spectral features evolved from the relativistic oscillating mirror (ROM) model [33, 36, 37] to much more detailed descriptions of the electron-vacuum-interface dynamics [30, 38, 39].

Based on the ROM model, general features of the high-harmonic spectrum can be predicted. The most prominent feature is the scaling of the intensity $I_n$ of the $n$-th harmonic according to $I_n \sim n^{-8/3}$ up the frequency cut-off $\omega_{\text{max}} \sim \gamma^3$ [40], after which an exponential roll-off in the spectrum appears. The ROM model is based on several assumptions, including a step-like plasma profile and that the plasma can be treated as cold. In an experiment the plasma surface will not be step-like, but have a finite gradient due thermal expansion of the plasma [41] which can be triggered by a pre-pulse.

The presence of a plasma gradient can lead to a mixing of relativistic and non-relativistic sources of high harmonics. In a very steep gradient the anharmonic motion of the critical-density surface dominates and the ROM model is applicable. For gradients of length $L/\lambda > a_0/2\pi$, plasma waves can be excited resonantly at the critical density and driven nonlinearly. The harmonics $N\omega_{pe}$ of the plasma frequency will mix with the incoming radiation and generate harmonics at $N\omega_0$ [41]. Furthermore, the electrons accelerated by breaking of the nonlinear plasma wave will be accelerated into the target where they bunch and subsequently excite plasma waves in the overcritical region. This process is called coherent wake emission and can generate radiation at the local plasma frequency $\omega_{pe}$ up to the maximum density of the plasma.

To study the influence of a finite plasma temperature on the spectrum of reflected radiation, we carried out a sequence of Vlasov simulations. At the same time we ran equivalent simulations with the EPOCH PIC code [42] to benchmark the Vlasov code.

We simulate the generation of high-harmonics from a plasma with maximum density $50n_c$. At the target front side we have a linear density ramp of length 0.1$\lambda_0$. The central laser wavelength is $\lambda_0 = 800\text{nm}$. The pulse envelope is of the form $a_0 \exp(-x^2/\sigma^2)$ with $a_0 = 5$ (corresponding to an intensity of about $5.3 \times 10^{19} \text{W/cm}^2$). The FWHM pulse duration is 4 cycles, i.e. corresponds to approx. 10fs. The initial temperature is varied between 1 and 80 keV. In the propagation direction we use 20k grid points to cover 9.1$\lambda_0$ of total box length ($2\lambda_0$ vacuum, 0.1$\lambda$ linear ramp from 0 to 50$n_c$, $5\lambda_0$ plateau, $1\lambda_0$ linear ramp and $1\lambda_0$ vacuum). In $p_x$ direction we use 20k grid points, covering $p_{\text{min}} = -8m_e c$ to $p_{\text{max}} = 9m_e c$. Ions are assumed to be immobile.

The laser fields are emitted from the left boundary of the simulation box, where the reflected fields are also recorded. At time $t = 9T_0$, where $T_0$ is the laser period, the maximum of the pulse is at $x/\lambda_0 = 1$.

Fig. 2 shows spectra that have been obtained for three different plasma temperatures, $T_e = 1, 20$ and 40 keV. It becomes evident that the plasma temperature has strong influence on the intensity of the highest harmonics. With rising temperature, we observe a clear drop in the intensity. For the 10th harmonic the difference in intensity is a factor of 3 between the 1keV and the 40keV case. For high harmonic numbers this difference grows
to more than three orders of magnitude. This clearly demonstrates that a pre-heating of the target by pre-pulses ought to be avoided. A finite temperature $T_e$ will induce a thermal electron momentum $\sqrt{T_e/m_e c^2}$ and an electric field at the boundary proportional to $\sqrt{n_0 T_e}$. For a laser amplitude $a_0 = 5$, the laser-induced electron momentum $a_0/\sqrt{n_0}$ is for the temperatures considered here always larger than the thermal momentum. However, the laser electric field is not always larger than the field due to the finite electron temperature.

In order to benchmark the code, the same physical setup has been simulated with the EPOCH PIC code. In the PIC simulations the spatial result was equal to the Vlasov simulation, i.e. $\lambda_0/\Delta x \approx 2000$. To obtain results close to the Vlasov simulation, we used 2000 particles per cell. Fig. 3 shows the phase-space densities obtained from both simulations at $t = 10T_0$, i.e. when the laser maximum reaches the overdense plasma for the case of $T_e = 5$keV. The phase-space densities have been normalized to their respective maxima and are shown using a logarithmic color scale. While the qualitative agreement is good, one can also observe that very fine details of the distribution function are not well resolved in the PIC results.

To investigate the influence of very fine structures in phase-space on the generation of high-harmonics, we compare the spectra we obtain from the two codes for two temperatures. Fig. 4 shows spectra for the case of $T_e = 1$keV and 40keV. In both cases we note that the agreement between the codes is very good for the low order harmonics and degrades with increasing harmonic order. A general feature of the spectra obtained from the PIC simulation is that the highest harmonics are not well resolved, whereas the signal from the Vlasov simulation still shows regular peaks at the harmonic frequencies,
Figure 3: (color online) Normalized phase-space densities from Vlasov (left) and PIC (right) simulations at $t = 10\tau_0$. Shown is $\log_{10}(f_e/\max(f_e))$ for the case of an initial plasma temperature of $T_e = 5\text{keV}$.

Figure 4: (color online) Harmonic spectrum obtained from Vlasov (blue/darker line) and PIC (orange/lighter line) simulations for temperatures a) $T = 1\text{keV}$ and b) $T = 40\text{keV}$. 

see insets in Fig. 4. Towards the end of the spectrum, $k/k_0 > 150$ for 1keV and $k/k_0 > 80$ for 40keV, we observe a clear difference between Vlasov and PIC results. For both temperatures the spectral intensity of the Vlasov results plateaus at normalized intensities of about $10^{-16}$. The PIC curve, on the other hand, exhibits a noisier signal at intensities between $10^{-12} - 10^{-10}$, which is growing towards larger $k$ values. Note that the maximum $k/k_0$ value shown in the graph is not the maximum resolved by both simulations ($k_{\text{max}}/k_0 \approx 1100$). The noise level of the PIC simulation is influenced only marginally by the number of particles per cell. When reducing the number from 2000 to 100, the results vary only little in terms of noise.
7 Conclusion

Numerical solution of the relativistic Vlasov-Maxwell system is one of the most basic approaches currently feasible to simulate problems in relativistic laser-plasma interaction. We discussed a semi-Lagrangian approach to solve the relativistic Vlasov-Maxwell system in 1D geometry.

Direct simulation of the Vlasov equation is favourable compared to PIC simulations in terms of noise. However, the numerical effort is typically larger compared to the latter. Hence, only efficient parallelization strategies can lead to acceptable simulation efforts that allow the use of high-resolution Vlasov simulations on regular basis.

In our manuscript we focused on the implementation strategy on a modern many-socket, many-core architecture and demonstrated that very good parallelization efficiency can be obtained. Paying respect to the distributed shared-memory topology of current multi-socket systems, we were able to achieve substantial speedups, scaling close to perfectly with the number of cores present in the system.

High resolution Vlasov simulations in higher dimensions will inevitably have to run on multiple machines. The environment might then require a hybrid MPI/OpenMP approach. From our results it is possible to draw the conclusion that it is possible to use OpenMP efficiently in many basic building blocks of such a higher dimensional Vlasov code.

We applied our Vlasov code to problems related to the generation of harmonic radiation from overdense plasmas. When comparing with corresponding PIC simulations, we find good agreement, however with a substantially reduced noise level in the Vlasov results. The spectrum obtained from the Vlasov simulation is clearer and shows more details at high frequencies. Furthermore, we addressed the impact of plasma temperature on the spectrum of harmonics. The simulation results show a clear degradation of the conversion efficiency to higher-frequency radiation with rising plasma temperature. This observation is very important in the context of experiments, where a finite pulse-contrast ratio will always lead to pre-heating of the target or the heating will take place during irradiation with a long pulse. A more in-depth analysis of the mechanism is now due, in order to understand the detailed mechanism. This is however beyond the scope of the current manuscript and will be subject of a separate study.

Acknowledgments

This work was performed under the auspices of the DFG-supported project SFB TR-18. Computing time granted by the John von Neumann Institute for Computing (NIC) and provided on the supercomputers JUROPA and JURECA at Jülich Supercomputing Centre (JSC) is acknowledged. Further computational infrastructure was provided by the Centre for Information and Media Technology (ZIM) at the University of Düsseldorf. The EPOCH code used in this research was developed under UK Engineering and Physics
Sciences Research Council grants EP/G054950/1, EP/G056803/1, EP/G055165/1 and EP/M022463/1. Valuable discussions with the group of A. Pukhov are acknowledged.

References


