Computer Physics Communications 222 (2018) 351–373

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

SMILEI: A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation☆

J. Derouillat [a], A. Beck [b], F. Pérez [c], T. Vinci [c], M. Chiaramello [d], A. Grassi [d,e,f], M. Flé [g], G. Bouchard [h], I. Plotnikov [i], N. Aunai [j], J. Dargent [i,j], C. Riconda [d], M. Grech [c,*]

[a] Maison de la Simulation, CEA, CNRS, Université Paris-Sud, UVSQ, Université Paris-Saclay, F-91191 Gif-sur-Yvette, France
[b] Laboratoire Leprince-Ringuet, École Polytechnique, CNRS-IN2P3, F-91128 Palaiseau, France
[c] Laboratoire d'Utilisation des Lasers Intenses, CNRS, École Polytechnique, CEA, Université Paris-Saclay, UPMC Université Paris 06: Sorbonne Universités, F-91128 Palaiseau Cedex, France
[d] Laboratoire d'Utilisation des Lasers Intenses, UPMC Université Paris 06: Sorbonne Universités, CNRS, Ecole Polytechnique, CEA, Université Paris-Saclay, F-75252 Paris Cedex 05, France
[e] Dipartimento di Fisica Enrico Fermi, Università di Pisa, Largo Bruno Pontecorvo 3, I-56127 Pisa, Italy
[f] Istituto Nazionale di Ottica, Consiglio Nazionale delle Ricerche (CNR/INO), u.o.s. Adriano Gozzini, I-56127 Pisa, Italy
[g] Institut du Développement des Ressources en Informatique Scientifique, CNRS, F-91403 Orsay, France
[h] Lasers, Interactions and Dynamics Laboratory, CEA, CNRS, Université Paris-Saclay, DSM/IRAMIS, CEN Saclay, F-91191 Gif sur Yvette, France
[i] Institut de Recherche en Astrophysique et Planétologie, Université de Toulouse, UPS-OMP, F-31400 Toulouse, France
[j] Laboratoire de Physique des Plasmas, Ecole Polytechnique, CNRS, UPMC, Université Paris-Sud, F-91128 Palaiseau, France

ARTICLE INFO

Article history:
Received 16 February 2017
Received in revised form 30 August 2017
Accepted 26 September 2017
Available online 10 October 2017

Keywords:
Plasma kinetic simulation
Particle-In-Cell (PIC)
High-performance computing
Laser–plasma interaction
Astrophysical plasmas

ABSTRACT

SMILEI is a collaborative, open-source, object-oriented (C++) particle-in-cell code. To benefit from the latest advances in high-performance computing (HPC), SMILEI is co-developed by both physicists and HPC experts. The code's structures, capabilities, parallelization strategy and performances are discussed. Additional modules (e.g. to treat ionization or collisions), benchmarks and physics highlights are also presented. Multi-purpose and evolutive, SMILEI is applied today to a wide range of physics studies, from relativistic laser–plasma interaction to astrophysical plasmas.

Program summary

Program title: SMILEI (version 3.2)
Program Files doi: http://dx.doi.org/10.17632/gsn4x6mbrg.1
Licensing provisions: This version of the code is distributed under the GNU General Public License v3
Programming language: C++11, Python 2.7
Nature of the problem: The kinetic simulation of plasmas is at the center of various physics studies, from laser–plasma interaction to astrophysics. To address today's challenges, a versatile simulation tool requires high-performance computing on massively parallel super-computers.
Solution method: The Vlasov–Maxwell system describing the self-consistent evolution of a collisionless plasma is solved using the Particle-In-Cell (PIC) method. Additional physics modules allow to account for additional effects such as collisions and/or ionization. A hybrid MPI-OpenMP strategy, based on a patch-based super-decomposition, allows for efficient cache-use, dynamic load balancing and high-performance on massively parallel super-computers.
Additional comments: Repository https://github.com/SmileiPIC/Smilei
References: http://www.maisondelasimulation.fr/smilei

1. Introduction

The Particle-In-Cell (PIC) approach was initially developed for fluid dynamics studies [1]. Having various advantages (conceptual simplicity, efficient implementation on massively parallel computers, etc.), it has become a central simulation tool for a wide range of physics studies, from semiconductors to cosmology or accelerator physics, and in particular to plasma physics. Today, the

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: mickael.grech@polytechnique.edu (M. Grech).

https://doi.org/10.1016/j.cpc.2017.09.024
0010-4655/

kinetic simulation of plasmas in various environments, from the laboratory to astrophysics, strongly relies on PIC codes [2].

In this paper, we present the new, open-source PIC code Smilei. It has been developed in a collaborative framework including physicists and high-performance computing (HPC) experts to best benefit from the new HPC architectures.

Smilei's development was initially motivated by recent advances in ultra-high intensity (UHI) laser technology, and new projects aiming at building multi-petawatt laser facilities. UHI laser–plasma interaction has indeed been successfully applied to probing matter under extreme conditions of temperature and pressure, opening the way to various promising applications such as charged-particle (electron and ion) acceleration [3–8], ultra-bright light sources of unprecedented short duration [9], or abundant electron–positron pair production [10,11]. This wide range of applications, as well as the associated deeper understanding of fundamental processes, lead to the creation of the *Centre Interdisciplinaire de la Lumière EXtrême* (CILEX)[1] [12]. This academic center will host, in the forthcoming years, the laser Apollon that will deliver ultra-short (15 fs), ultra-intense (beyond 10^{22} W/cm^2) laser pulses, corresponding to a record peak power of 10 PW. This path toward the study of light–matter interaction at extreme intensities represents a significant experimental and technological undertaking. New numerical tools have to be deployed as laser–plasma interaction, at intensities beyond 10^{22} W/cm^2, is not only relativistic but also highly nonlinear and of quantum nature [13].

Furthermore, a paradigm shift has occurred in HPC. The number of cores available on massively parallel supercomputers has skyrocketed. Recent and future supercomputer architectures are also hybrid systems relying on both *distributed* and *shared memory*. This introduces challenges in workload management, resource scheduling and data structure at the algorithmic and software development levels. These tendencies are progressing quickly, and software development lags behind. Today, most of the codes used by the plasma community face difficulties when confronted with these new challenges. As a result, running today's software technology on the most recent and forthcoming machines is intractable as well as a tremendous waste of resources and electric power. These limitations are inherent to the internal structure of the codes, and can be overcome only through a strong collaboration between physicists and HPC specialists, to build, from the ground up, a new HPC-relevant simulation tool.

In this context, in early 2013, a consortium of laboratories of the *Plateau de Saclay* decided to join their efforts in developing the new PIC code Smilei (for *Simulating Matter Irradiated by Light at Extreme Intensities*). Intended as a multi-purpose and collaborative PIC code, Smilei addresses a wide range of physics problems, from laser–plasma interaction to astrophysics.

This paper presents an overview of the code's principles, structure, performance and capabilities, as well as benchmarks and examples. Section 2 reviews the general PIC approach for simulating collisionless plasmas (the governing equations, and the associated numerical methods), and specifies the algorithms used in Smilei. The C++ object-oriented programming and polymorphism, highlighted in Section 3, illustrate the multi-purpose, multi-physics and multi-geometry aspects of the code and its modularity and maintainability. We outline Smilei's components, their interactions and the I/O management strategy. Section 4 then presents the innovative parallelization strategy devised for Smilei. In particular, the hybrid MPI-OpenMP (for synchronization in between distributed and shared memory processes) and dynamic load balancing designs are built around "patches", which extend the notion of domain decomposition and improve data locality for faster memory access and efficient cache use. The code performance on massively-parallel super-computers is then discussed. The following Section 5 describes additional modules (binary collisions, ionization, etc.), and Section 6 explains the input interface and the output diagnostics. Section 7 features applications to different physical scenarii, the first two related to UHI laser–plasma interaction and the other two to astrophysics. Finally, Section 8 concludes on Smilei capabilities and perspectives.

[1] http://goo.gl/kzJCjY.

2. The Particle-In-Cell (PIC) method for collisionless plasmas

2.1. The Vlasov–Maxwell model

The kinetic description of a collisionless plasma[2] relies on the Vlasov–Maxwell system of equations. In this description, the different species of particles constituting the plasma are described by their respective distribution functions $f_s(t, \mathbf{x}, \mathbf{p})$, where s denotes a given species consisting of particles with charge q_s and mass m_s, and $\mathbf{x}$ and $\mathbf{p}$ denote the position and momentum of a phase-space element. The distribution f_s satisfies Vlasov's equation:

$$\left(\partial_t + \frac{\mathbf{p}}{m_s\gamma}\cdot\nabla + \mathbf{F}_L\cdot\nabla_{\mathbf{p}}\right) f_s = 0, \tag{1}$$

where $\gamma = \sqrt{1+\mathbf{p}^2/(m_s c)^2}$ is the (relativistic) Lorentz factor, c is the speed of light in vacuum, and

$$\mathbf{F}_L = q_s\left(\mathbf{E} + \mathbf{v}\times\mathbf{B}\right) \tag{2}$$

is the Lorentz force acting on a particle with velocity $\mathbf{v} = \mathbf{p}/(m_s\gamma)$.

This force follows from the existence, in the plasma, of collective electric [$\mathbf{E}(t, \mathbf{x})$] and magnetic [$\mathbf{B}(t, \mathbf{x})$] fields satisfying Maxwell's equations[3]:

$$\nabla\cdot\mathbf{B} = 0, \tag{3a}$$

$$\nabla\cdot\mathbf{E} = \rho/\epsilon_0, \tag{3b}$$

$$\nabla\times\mathbf{B} = \mu_0\mathbf{J} + \mu_0\epsilon_0\,\partial_t\mathbf{E}, \tag{3c}$$

$$\nabla\times\mathbf{E} = -\partial_t\mathbf{B}, \tag{3d}$$

where ϵ_0 and μ_0 are the vacuum permittivity and permeability, respectively.

The Vlasov–Maxwell system of Eqs. (1)–(3) describes the self-consistent dynamics of the plasma which constituents are subject to the Lorentz force, and in turn modify the collective electric and magnetic fields through their charge and current densities:

$$\rho(t, \mathbf{x}) = \sum_s q_s \int d^3p\, f_s(t, \mathbf{x}, \mathbf{p}), \tag{4a}$$

$$\mathbf{J}(t, \mathbf{x}) = \sum_s q_s \int d^3p\, \mathbf{v} f_s(t, \mathbf{x}, \mathbf{p}). \tag{4b}$$

2.2. Reference units

Smilei is a fully-relativistic electromagnetic PIC code. As such, it is convenient to normalize all velocities in the code to c. Furthermore, charges and masses are normalized to e and m_e, respectively, with $-e$ the electron charge and m_e its mass. Momenta and energies (and by extension temperatures) are then expressed in units of $m_e c$ and $m_e c^2$, respectively.

[2] The PIC method can be applied to (fully or partially ionized) plasmas as well as beams of charged particles. For the sake of simplicity however, we will refer to all these states as plasmas.

[3] It is important to stress that the electromagnetic fields considered here are macroscopic (mean) fields, and not microscopic fields. Therefore, the PIC simulation does not, in its standard form, accounts for particle collisions. Collisions are however introduced in an *ad hoc* module presented in Section 5.4.

Table 1
List of the most common normalizations used in Smilei. The value of ω_r is not defined *a priori*, but can be set *a posteriori* as a scaling factor. For simulations requiring the use of ionization and/or collision modules (see Section 5), ω_r needs to be defined, in SI units, by the user.

Units of velocity	c
Units of charge	e
Units of mass	m_e
Units of momentum	$m_e\,c$
Units of energy, temperature	$m_e c^2$
Units of time	ω_r^{-1}
Units of length	c/ω_r
Units of number density	$n_r = \epsilon_0\, m_e\, \omega_r^2/e^2$
Units of current density	$e\,c\,n_r$
Units of pressure	$m_e\,c^2\,n_r$
Units of electric field	$m_e\,c\,\omega_r/e$
Units of magnetic field	$m_e\,\omega_r/e$
Units of Poynting flux	$m_e\,c^3\,n_r/2$

The normalization for time and space is not decided *a priori*. Instead, all the simulation results may be scaled by an arbitrary factor. Denoting the (*a priori* unknown) time units by ω_r^{-1}, distances are normalized to c/ω_r. Electric and magnetic fields are expressed in units of $m_e c\,\omega_r/e$ and $m_e\omega_r/e$, respectively. We define the units for number densities as $n_r = \epsilon_0 m_e\omega_r^2/e^2$, while charge and current densities are in units of $e\,n_r$ and $e\,c\,n_r$, respectively. Note that this definition of n_r is chosen for best simplification of the Vlasov–Maxwell equations, but does not correspond to the reference distance c/ω_r to the power of -3.

Let us now illustrate by two simple examples this choice of normalization. When dealing with a plasma at constant density n_e, it is convenient to normalize times by introducing the electron plasma frequency $\omega_{pe} = \sqrt{e^2 n_e/(\epsilon_0 m_e)}$. Choosing $\omega_r = \omega_{pe}$, distances are now expressed in units of the electron skin-depth c/ω_{pe}, while number densities are normalized to n_e, and the electric and magnetic fields are in units of $m_e\omega_{pe}c/e$ and $m_e\omega_{pe}/e$, respectively.

In contrast, when considering the irradiation of a plasma by a laser with angular frequency ω_0, it is convenient to use $\omega_r = \omega_0$. From this choice, it follows that distances are measured in units of $k_0^{-1} = c/\omega_0$, while the electric and magnetic fields are in units of $E_c = m_e c\omega_0/e$ and $m_e\omega_0/e$, respectively. Note that E_c is the Compton field, which is widely used to measure the importance of relativistic effects in laser–plasma interaction. In addition, number densities are expressed in units of $n_c = \epsilon_0 m_e\omega_0^2/e^2$, the well-known critical density delimiting plasmas that are transparent or opaque to an electromagnetic radiation with angular frequency ω_0.

Table 1 gives a list of the most common normalizations used in Smilei. In what follows (and if not specified otherwise), all quantities will be expressed in normalized units.

2.3. Quasi-particles and the PIC method

The "Particle-In-Cell" method owes its name to the discretization of the distribution function f_s as a sum of N_s "quasi-particles" (also referred to as "super-particles" or "macro-particles"):

$$f_s(t,\mathbf{x},\mathbf{p}) = \sum_{p=1}^{N_s} w_p\; S\big(\mathbf{x}-\mathbf{x}_p(t)\big)\,\delta\big(\mathbf{p}-\mathbf{p}_p(t)\big), \tag{5}$$

where w_p is a quasi-particle "weight", $\mathbf{x}_p$ is its position, $\mathbf{p}_p$ is its momentum, δ is the Dirac distribution, and $S(\mathbf{x})$ is the shape-function of all quasi-particles. The properties of the shape-function used in Smilei are given in Appendix.

In PIC codes, Vlasov's equation (1) is integrated along the continuous trajectories of these quasi-particles, while Maxwell's equations (3) are solved on a discrete spatial grid, the spaces between consecutive grid points being referred to as "cells". Injecting the discrete distribution function of Eq. (5) in Vlasov's equation (1), multiplying the result by $\mathbf{p}$ and integrating over all $\mathbf{p}$ leads to:

$$\sum_{p=1}^{N_s} w_p\,\mathbf{p}_p\cdot\left[\partial_{\mathbf{x}_p}S(\mathbf{x}-\mathbf{x}_p)+\partial_{\mathbf{x}}S(\mathbf{x}-\mathbf{x}_p)\right]\mathbf{v}_p + \sum_{p=1}^{N_s} w_p\,S(\mathbf{x}-\mathbf{x}_p)\left[\partial_t\mathbf{p}_p - q_s\left(\mathbf{E}+\mathbf{v}_p\times\mathbf{B}\right)\right] = 0, \tag{6}$$

where we have introduced $\mathbf{v}_p = \mathbf{p}_p/(m_s\gamma_p) = d\mathbf{x}_p/dt$ the pth quasi-particle velocity, and $\gamma_p = \sqrt{1+\mathbf{p}_p^2/(m_s^2)}$ its Lorentz factor. Considering all p quasi-particles independently, and integrating over all (real) space $\mathbf{x}$, the first term in Eq. (6) vanishes due to the properties of the shape-function (see Appendix) and one obtains that all quasi-particles satisfy the relativistic equations of motion:

$$\frac{d\mathbf{x}_p}{dt} = \frac{\mathbf{u}_p}{\gamma_p} \tag{7}$$

$$\frac{d\mathbf{u}_p}{dt} = r_s\left(\mathbf{E}_p + \frac{\mathbf{u}_p}{\gamma_p}\times\mathbf{B}_p\right), \tag{8}$$

where we have introduced $r_s = q_s/m_s$ the charge-over-mass ratio (for species s), $\mathbf{u}_p = \mathbf{p}_p/m_s$ the quasi-particle reduced momentum, and the fields interpolated at the particle position:

$$\mathbf{E}_p = \int d\mathbf{x}\,S(\mathbf{x}-\mathbf{x}_p)\,\mathbf{E}(\mathbf{x}), \tag{9}$$

$$\mathbf{B}_p = \int d\mathbf{x}\,S(\mathbf{x}-\mathbf{x}_p)\,\mathbf{B}(\mathbf{x}). \tag{10}$$

Note that, because of the finite (non-zero) spatial extension of the quasi-particles (also referred to as quasi-particle size, Appendix), additional cells (called *ghost cells*, see Section 4) have to be added at the border of the simulation domain to ensure that the full quasi-particle charge and/or current densities are correctly projected onto the simulation grid.

In this Section, we present the general PIC algorithm, starting with the simulation initialization and then going through the PIC loop itself (see Table 2).

2.4. Time- and space-centered discretization

As will be discussed in Section 2.6.4, Maxwell's equations are solved here using the Finite Difference Time Domain (FDTD) approach [14] as well as refined methods based on this algorithm (for a review of these methods see [15]). In these methods, the electromagnetic fields are discretized onto a staggered grid, the Yee-grid, that allows for spatial-centering of the discretized curl operators in Maxwell's equations (3c) and (3d). Fig. 1 summarizes at which points of the Yee-grid the electromagnetic fields, as well as charge and density currents, are defined. Similarly, the time-centering of the time-derivative in Maxwell's equations (3c) and (3d) is ensured by considering the electric fields as defined at integer time-steps (n) and magnetic fields at half-integer time-steps ($n+\frac{1}{2}$). Time-centering of the magnetic fields is however necessary for diagnostic purposes, and most importantly when computing the Lorentz force acting on the quasi-particles. It should also be noted, as will be discussed in Section 2.6.2, that a *leap-frog* scheme is used to advance the particles in time, so that their positions and velocities are defined at integer (n) and half-integer ($n-\frac{1}{2}$) time-steps, respectively.

2.5. Initialization of the simulation

The initialization of a PIC simulation is a three-step process consisting in: (i) loading particles, (ii) computing the initial total charge and current densities onto the grid, and (iii) computing the

Table 2
Summary of SMILEI's PIC algorithm.

Initialization	time step $n = 0$, time $t = 0$
Particle loading	$\forall p$, define $(\mathbf{x}_p)^{n=0}, (\mathbf{u}_p)^{n=-\frac{1}{2}}$
Charge projection on grid	$\left[\forall p, (\mathbf{x}_p)^{n=0}\right] \rightarrow \rho^{(n=0)}(\mathbf{x})$
Compute initial fields	- solve Poisson on grid: $\left[\rho^{(n=0)}(\mathbf{x})\right] \rightarrow \mathbf{E}_{\text{stat}}^{(n=0)}(\mathbf{x})$
	- add external fields: $\mathbf{E}^{(n=0)}(\mathbf{x}) = \mathbf{E}_{\text{stat}}^{(n=0)}(\mathbf{x}) + \mathbf{E}_{\text{ext}}^{(n=0)}(\mathbf{x})$
	$\mathbf{B}^{(n=\frac{1}{2})}(\mathbf{x}) = \mathbf{B}_{\text{ext}}^{(n=\frac{1}{2})}(\mathbf{x})$
PIC loop: from time step n to $n+1$, time $t = (n+1)\,\Delta t$	
Restart charge & current densities	
Save magnetic fields value (used to center magnetic fields)	
Interpolate fields at particle positions	$\forall p, [\mathbf{x}_p, \mathbf{E}^{(n)}(\mathbf{x}), \mathbf{B}^{(n)}(\mathbf{x})] \rightarrow \mathbf{E}_p^{(n)}, \mathbf{B}_p^{(n)}$
Push particles	- compute new velocity $\forall p, \mathbf{p}_p^{(n-\frac{1}{2})}\ \left[\mathbf{E}_p^{(n)}, \mathbf{B}_p^{(n)}\right]\ \mathbf{p}_p^{(n+\frac{1}{2})}$
	- compute new position $\forall p, \mathbf{x}_p^{(n)}\ \left[\mathbf{p}_p^{(n+\frac{1}{2})}\right]\ \mathbf{x}_p^{(n+1)}$
Project current onto the grid	using a charge-conserving scheme
	$\left[\forall p\ \ \mathbf{x}_p^{(n)}, \mathbf{x}_p^{(n+1)}, \mathbf{p}_p^{(n+\frac{1}{2})}\right] \rightarrow \mathbf{J}^{(n+\frac{1}{2})}(\mathbf{x})$
Solve Maxwell's equations	
	- solve Maxwell–Faraday: $\mathbf{E}^{(n)}(\mathbf{x})\ \left[\mathbf{J}^{(n+\frac{1}{2})(\mathbf{x})}\right]\ \mathbf{E}^{(n+1)}(\mathbf{x})$
	- solve Maxwell–Ampère: $\mathbf{B}^{(n+\frac{1}{2})}(\mathbf{x})\ \left[\mathbf{E}^{(n+1)}(\mathbf{x})\right]\ \mathbf{B}^{(n+\frac{3}{2})}(\mathbf{x})$
	- center magnetic fields: $\mathbf{B}^{(n+1)}(\mathbf{x}) = \frac{1}{2}\left(\mathbf{B}^{(n+\frac{1}{2})}(\mathbf{x}) + \mathbf{B}^{(n+\frac{3}{2})}(\mathbf{x})\right)$

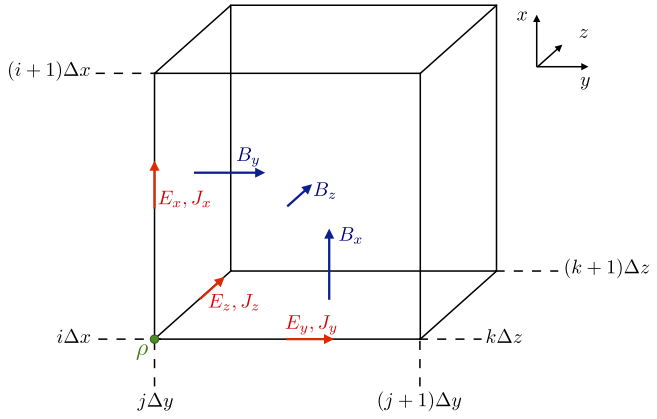

Fig. 1. Representation of the staggered Yee-grid. The location of all fields and current densities follows from the (rather standard) convention to define charge densities at the cell nodes.

initial electric and magnetic field at the grid points. In SMILEI, all three steps can be done either as a restart of a previous simulation (in which case the particles, charge and current densities and electromagnetic fields are directly copied from a file generated at the end of a previous simulation), or from a user-defined input file. In that case, the user defines the initial conditions of the particle, charge and current densities as well as the initial electromagnetic fields over the whole simulation domain.

In particular, the user prescribes spatial profiles for the number density n_s, the number of particle per cell N_s, the mean velocity $\mathbf{v}_s$ and the temperature T_s of each species s at time $t = 0$. The particle loading then consists in creating, in each cell, N_s particles with positions $\mathbf{x}_p$ uniformly distributed within the cell (either randomly or regularly spaced), and with momenta $\mathbf{p}_p$ randomly sampled from a requested distribution.[4] In SMILEI, a given numerical weight w_p is assigned to each particle depending on the density associated to the cell it originates from:

$$w_p = \frac{n_s\big(\mathbf{x}_p(t=0)\big)}{N_s\big(\mathbf{x}_p(t=0)\big)}. \tag{11}$$

This variable weighting is particularly beneficial when considering initially highly inhomogeneous density distributions.

Once all particles in the simulation domain have been created, the total charge and current densities $\rho(t = 0, \mathbf{x})$ and $\mathbf{J}(t = 0, \mathbf{x})$ are computed onto the grid using a direct projection technique (see Appendix for more details) that assigns to a grid point located at $\mathbf{x}_i$ the total charge and or current contained in the cell surrounding it:

$$\rho(t=0,\mathbf{x}) = \sum_s q_s \sum_p w_p \int d\mathbf{x}\, S\big(\mathbf{x} - \mathbf{x}_p(t=0)\big) P_D(\mathbf{x} - \mathbf{x}_i), \tag{12}$$

where $P_D(\mathbf{x}) = \Pi_{\mu=1}^{D} P(x^\mu)$ (D referring to the number of spatial dimensions) with $P(x)$ the crenel function such that $P(x^\mu) = 1$ if $|x^\mu| < \Delta\mu/2$ and $P(x^\mu) = 0$ otherwise, and $\Delta\mu$ is the cell length in the $\mu = (x, y, z)$-direction.

Then, the initial electric fields are computed from $\rho(t = 0, \mathbf{x})$ by solving Poisson's equation (3b). In SMILEI, this is done using the conjugate gradient method [18]. This iterative method is particularly interesting: it is easily implemented on massively parallel computers as it requires mainly local information exchange between adjacent domains (see Section 4 for more information on domain decomposition for parallelization).

External (divergence-free) electric and/or magnetic fields can then be added to the resulting electrostatic fields, provided they fulfill Maxwell's equations (3), and in particular Gauss and Poisson equations (3a) and (3b).

2.6. The PIC loop

At the end of the initialization stage [time-step ($n = 0$)], all quasi-particles in the simulation have been loaded and the electromagnetic fields have been computed over the whole simulation grid. The PIC loop is then started over N time-steps each consisting in (i) interpolating the electromagnetic fields at the particle positions, (ii) computing the new particle velocities and positions,

[4] The user may select a zero-temperature distribution, a Maxwellian distribution, or Maxwell–Jüttner distribution, i.e. the relativistic generalization of the Maxwellian distribution [16]. In the latter case, the method proposed in Ref. [17] is used to ensure a correct loading of particles with a relativistic drift velocity.

(iii) projecting the new charge and current densities on the grid, and (iv) computing the new electromagnetic fields on the grid. In this section, we describe these four steps taken to advance from time-step (n) to time-step $(n+1)$.

2.6.1. Field interpolation at the particle

At the beginning of time-step (n), the particles velocities and positions are known at time-step $(n-\frac{1}{2})$ and (n), respectively. For each particle p, the electromagnetic fields [at time-step (n)] are computed at the particle position using a simple interpolation technique:

$$\mathbf{E}_p^{(n)} = \int d\mathbf{x}\, S\left(\mathbf{x}-\mathbf{x}_p^{(n)}\right)\mathbf{E}^{(n)}(\mathbf{x}), \tag{13}$$

$$\mathbf{B}_p^{(n)} = \int d\mathbf{x}\, S\left(\mathbf{x}-\mathbf{x}_p^{(n)}\right)\mathbf{B}^{(n)}(\mathbf{x}), \tag{14}$$

where we have used the time-centered magnetic fields $\mathbf{B}^{(n)} = \frac{1}{2}[\mathbf{B}^{(n+1/2)} + \mathbf{B}^{(n-1/2)}]$. Additional information on the field interpolation are given in Appendix.

2.6.2. Particle pusher

Knowing, for each quasi-particle, the electromagnetic fields at its position, the new particle momentum and position are computed using a (second order) leap-frog integrator. In Smilei, two different schemes have been implemented, the well-known Boris pusher [19] and the one developed by J.-L. Vay [20]. Both schemes compute the new particle momentum according to:

$$\mathbf{u}_p^{(n+\frac{1}{2})} = \mathbf{u}_p^{(n-\frac{1}{2})} + r_s \Delta t \left[E_p^{(n)} + \frac{\mathbf{v}_p^{(n+\frac{1}{2})} + \mathbf{v}_p^{(n-\frac{1}{2})}}{2} \times B_p^{(n)} \right], \tag{15}$$

as well as the new particle position:

$$\mathbf{x}_p^{(n+1)} = \mathbf{x}_p^{(n)} + \Delta t\, \frac{\mathbf{u}_p^{(n+\frac{1}{2})}}{\gamma_p}, \tag{16}$$

where Δt denotes the duration of a time-step.

The Boris pusher is a widely-used second-order leap-frog solver. However, Ref. [20] shows that it introduces errors when calculating the orbits of relativistic particles in special electromagnetic field configurations (e.g. when the electric and magnetic contributions cancel each other in the Lorentz force). Vay's solver proposes an alternative formulation of the leap-frog solver that prevents such problems with an additional (albeit not large) computational cost.

2.6.3. Charge conserving current deposition

Charge deposition (i.e. charge and current density projection onto the grid) is then performed using the charge-conserving algorithm proposed by Esirkepov [21]. The current densities in the dimensions of the grid (i.e., the x-direction for 1-dimensional simulations, both x- and y-directions for 2-dimensional simulations, and all three x-, y- and z-directions for 3-dimensional simulations) are computed from the charge flux through the cell borders (hence ensuring charge conservation) while the current densities along the other dimensions are performed using a simple projection. To illustrate this point, we take the example of current deposition in a 2-dimensional simulation. The current densities in the x- and y-directions associated to a particle with charge q are computed as:

$$(J_{x,p})_{i+\frac{1}{2},j}^{(n+\frac{1}{2})} = (J_{x,p})_{i-\frac{1}{2},j}^{(n+\frac{1}{2})} + q\, w_p\, \frac{\Delta x}{\Delta t}\, (W_x)_{i+\frac{1}{2},j}^{(n+\frac{1}{2})} \tag{17}$$

$$(J_{y,p})_{i,j+\frac{1}{2}}^{(n+\frac{1}{2})} = (J_{y,p})_{i,j-\frac{1}{2}}^{(n+\frac{1}{2})} + q\, w_p\, \frac{\Delta y}{\Delta t}\, (W_y)_{j,i+\frac{1}{2}}^{(n+\frac{1}{2})} \tag{18}$$

where $(W_x)^{(n+\frac{1}{2})}$ and $(W_y)^{(n+\frac{1}{2})}$ are computed from the particle present and former positions $x_p^{(n+1)}$ and $x_p^{(n)}$, respectively, using the method developed by Esirkepov. The particle current in the z-direction (not a dimension of the grid) is, in this geometry, computed using the direct projection technique described in Appendix:

$$(J_{z,p})_{i,j} = q w_r \mathbf{v}_p \int d\mathbf{x}\, S(\mathbf{x}-\mathbf{x}_p)\, P_D(\mathbf{x}-\mathbf{x}_{i,j}). \tag{19}$$

The charge density deposited by the particle can be obtained, if required e.g. for diagnostic purpose, using a similar direct projection.

The total charge and current densities henceforth gather the contributions of all quasi-particles of all species. It is worth noting that, within a charge-conserving framework, charge densities are only projected on the grid for diagnostics purposes (as we will see in next paragraph, it is not used to advance the electromagnetic fields).

2.6.4. Maxwell solvers

Now that the currents are known at time-step $(n+\frac{1}{2})$, the electromagnetic fields can be advanced solving Maxwell's equations (3). First, Maxwell–Ampère equation (3c) is solved, giving the advanced electric fields:

$$\mathbf{E}^{(n+1)} = \mathbf{E}^{(n)} + \Delta t \left[(\nabla \times \mathbf{B})^{(n+\frac{1}{2})} - \mathbf{J}^{(n+\frac{1}{2})} \right]. \tag{20}$$

Then, Maxwell–Faraday equation (3d) is computed, leading to the advanced magnetic fields:

$$\mathbf{B}^{(n+\frac{3}{2})} = \mathbf{B}^{(n+\frac{1}{2})} - \Delta t\, (\nabla \times \mathbf{E})^{(n+1)}. \tag{21}$$

Before discussing the discretization of the curl-operator in more details, it is worth noting that solving Eqs. (3c) and (3d) is sufficient to get a complete description of the new electromagnetic fields. Indeed, it can be shown that this conserves a divergence-free magnetic field if Gauss' equation (3a) is satisfied at time $t=0$. Similarly, Poisson's equation (3b) is verified as long as it is satisfied at time $t=0$ as long as the charge deposition algorithm fulfills the charge conservation equation:

$$\partial_t \rho + \nabla \cdot \mathbf{J} = 0. \tag{22}$$

This motivated the use of Esirkepov's projection scheme discussed in the previous paragraph.

We conclude this Section by discussing in more details the discretization of the curl-operators in Eqs. (3c) and (3d). To do so, let us focus on the equations for the electric and magnetic fields E_x and B_x discretized on the (staggered) Yee-grid:

$$\frac{(E_x)_{i+\frac{1}{2},j,k}^{(n+1)} - (E_x)_{i+\frac{1}{2},j,k}^{(n)}}{\Delta t} = (J_x)_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}} + \left(\partial_y B_z\right)_{i+\frac{1}{2},j,k}^{(n+\frac{1}{2})} - \left(\partial_z B_y\right)_{i+\frac{1}{2},j,k}^{(n+\frac{1}{2})}, \tag{23}$$

$$\frac{(B_x)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+\frac{3}{2})} - (B_x)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+\frac{1}{2})}}{\Delta t} = \left(\partial_z^* E_y\right)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+1)} - \left(\partial_y^* E_z\right)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+1)}. \tag{24}$$

The partial derivatives in space in both equations are discretized as follows. In the Maxwell–Ampère equation, the partial derivative in x (similarly in y and z) reads:

$$(\partial_x F)_{i,j,k} = \frac{F_{i+\frac{1}{2},j,k} - F_{i-\frac{1}{2},j,k}}{\Delta x}, \tag{25}$$

and corresponds to the usual curl-operator discretization used in the FDTD method. In the Maxwell–Faraday equation, the partial

derivatives can be modified using an extended stencil (see Ref. [15] for a comparative study of different solvers). The spatial derivative in the x-direction (similarly in the y and z directions) reads:

$$\begin{aligned}&\left(\partial_x^* F\right)_{i,j,k}\\&= \alpha_x \frac{F_{i+\frac{1}{2},j,k} - F_{i-\frac{1}{2},j,k}}{\Delta x} + \eta_x \frac{F_{i+\frac{3}{2},j,k} - F_{i-\frac{3}{2},j,k}}{\Delta x}\\&+\beta_{xy}\left[\frac{F_{i+\frac{1}{2},j+1,k} - F_{i-\frac{1}{2},j+1,k}}{\Delta x} + \frac{F_{i+\frac{1}{2},j-1,k} - F_{i-\frac{1}{2},j-1,k}}{\Delta x}\right]\\&+\beta_{xz}\left[\frac{F_{i+\frac{1}{2},j,k+1} - F_{i-\frac{1}{2},j,k+1}}{\Delta x} + \frac{F_{i+\frac{1}{2},j,k-1} - F_{i-\frac{1}{2},j,k-1}}{\Delta x}\right],\end{aligned} \tag{26}$$

the set of parameters α_x, η_x, β_{xy} and β_{xz} depending of the type of solver used [15], and the standard FDTD solver is recovered for $\alpha_x = 1$, $\eta_x = \beta_{xy} = \beta_{xz} = 0$.

Note that the FDTD solvers are subject to a Courant–Friedrich–Lewy (CFL) condition. For the standard solver, the CFL condition requires the time-step to be smaller than:

$$\Delta t_{\text{CFL}} = \left[\sum_{\mu} \Delta\mu^{-2}\right]^{-1/2}, \tag{27}$$

$\mu = (x, y, z)$ standing for the different spatial directions resolved in the simulation.

2.6.5. Boundary conditions

After having computed new quasi-particle positions and velocities, boundary conditions (BCs) are applied to each quasi-particle that may be located in a ghost cell, i.e. outside of the 'real' grid. Quasi-particle species may have a different BC for each boundary of the simulation box: the quasi-particles can either loop around the box (periodic), be stopped (momentum set to zero), suppressed (removed from memory), reflected (momentum and position follow specular reflection rules) or thermalized. In the latter case, the quasi-particle is set back inside the simulation box, and its new momentum is randomly sampled in a Maxwellian distribution [22] with a given temperature and drift velocity, both specified by the user.

BCs are applied to the electromagnetic fields after Maxwell's equations have been solved. Each boundary of the simulation box can feature a different BC. First, injecting/absorbing BCs inspired from the "Silver–Müller" BC [23] are able to inject an electromagnetic wave (e.g. a laser) and/or to absorb outgoing electromagnetic waves.[5] In contrast, the reflective electromagnetic BC will reflect any outgoing electromagnetic wave reaching the simulation boundary. Lastly, periodic BCs are also available.

3. An evolutive, multi-purpose code

Smilei's objectives are high performances, a large user community and support for a variety of applications. Its C++ approach reflects these goals, providing structure to separate physics from computing aspects, to encourage their progress, to facilitate their maintainability and to ensure a multi-purpose capability.

[5] In addition, Perfectly Matched Layers [24] and advanced BCs to model arbitrarily shaped, tightly focused laser pulses [25] are currently being considered for implementation in Smilei.

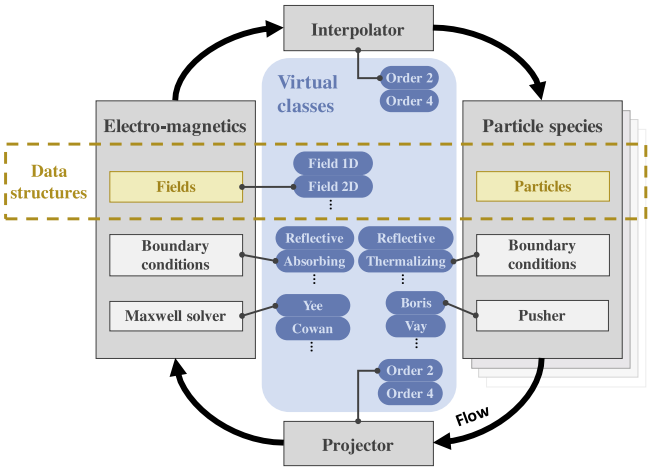

Fig. 2. C++ flow, classes and data structure in Smilei.

3.1. C++ elements and flow

Smilei's core program is written in the C++ language. Its multi-purpose and mature technology ensures great flexibility and strong support for the new HPC machines. Moreover, C++'s object-oriented programming provides an efficient way of structuring the code. Importantly, this eliminates a few bad habits such as passing large lists of parameters through functions, or usage of global variables, inefficient in parallel computing. Components can be constructed almost independently. It offers a good separation between the purely computing/performance aspects and the physics calculations.

Fig. 2 shows the various elements of Smilei's main code: C++ classes, data structure, and the program flow. The main classes, namely "particle species" and "electromagnetics", are the counterparts of *particle* and *cell* in *Particle-in-cell*, respectively. The particle species class hold the particle object, which is the data structure for the quasi-particles positions and momenta. It also contains operators on the quasi-particles such as the boundary conditions and the pusher. On the other side, the electromagnetics class contains the fields, i.e. the data structure for the electric and magnetic fields. Note that these fields also describe the charge and current densities projected onto the grid. Electromagnetics also includes operators such as the Maxwell solver and the boundary conditions for the fields.

Two additional operators are external to those structures because they operate between particles and fields. The interpolator takes the field data and interpolates it at the particles positions. The projector takes the particle data and projects it at the grid points.

3.2. Polymorphism

The C++ language supports the definition of *polymorphic* classes. These classes contain functions, called *virtual functions*, that are selected at runtime among several options. In other words, the behavior of an object is not decided *a priori*, but may be defined during the simulation.

Smilei relies on C++ polymorphism to handle its multi-purpose ambition. For instance, the basic polymorphic `Field` class may be derived into different classes such as `Field1D`, `Field2D`, etc. All these derived classes inherit their functions from the base class, but they include different data structures. In Fig. 2, examples of polymorphic (*virtual*) classes are highlighted. Note that, in Smilei, selecting the class from which each object will be created is ensured by a "factory design pattern".

There are several advantages to polymorphism. First, it allows for straightforward inheritance of properties between objects of similar structures. It also improves the readability of the code by removing the complexity of all the multi-purpose capabilities from the program flow. Lastly, it standardizes the form of the objects for easier maintenance. In these conditions, a single executable file can perform simulations in various dimensions, interpolation orders, or physics components, without the complexity of many code versions.

However, an excess of virtualization, or a large number of objects layers could have a significant computational cost. For instance, the use of a virtual method to access a single data element (e.g., a single particle property) would have an unacceptable data access overhead. This pitfall is avoided by passing the whole data structures to computational operators. They are passed in their virtual form, then cast to the required class by the operator itself.

3.3. Uncoupling operators from data

An other fundamental ambition of the project is to provide an efficient tool of simulation on current and future supercomputers whose architectures are in permanent evolution. For instance, they may have complex memory hierarchy, whether distributed or shared between several processors. For ideal performances, the code must be adapted to these specific architectures. Besides this multi-machine aspect, a multi-purpose code (able to simulate various physical scenarii) may require a different optimization strategy depending on the subject of each simulation.

For these two challenges, Smilei's solution is based on its object-oriented design: it consists in uncoupling the computing algorithms from the data formalism. In all operators (solvers, interpolators, projectors, etc.), algorithms do not rely on raw data but on wrappers (`Field` and `Particles`) which encapsulate and provide access to the data. Operators can thus be defined independently from the chosen data structure, provided the "protocol" for accessing to the data is respected. As a consequence, performances can be optimized separately in operators and in the data structures.

Along the same principle, parallelism management tends to be decoupled from the physics calculations by implementing different levels of parallelism, as detailed in Section 4.2.2.

3.4. HDF5 data management

A significant amount of output data is generated by PIC simulations. We examine here the representation of these data, focusing on the data access convenience and performances on a large supercomputer.

Classical output management would simply consist in gathering data on a "master" processor which writes everything out, or in generating one file for each processor. The former technique is limited by the cost of communicating data and its memory overhead, while the latter requires heavy post-processing. In both cases, the larger the simulation, the more expensive the overhead.

Parallel I/O libraries are optimized to avoid these pitfalls, and their development continuously improves their performances. They can share and write data in parallel to a single file. Famous examples are *MPI-IO*,[6] *HDF5* (*Hierarchical Data Format*[7]) and *NetCDF* (*Network Common Data Form*[8]). Although no parallel I/O library is yet fully optimized for the most recent parallelism techniques, they greatly enhance the simulations efficiency.

MPI-IO has demonstrated good performances, but it generates unformatted data, thus requiring an additional effort from the user to access and analyze the simulation data. In contrast, both *HDF5* and *NetCDF* rely on a structured data model, which is also open-source and widely used. *HDF5* also benefits from a large panel of open-source software for post-processing and visualization. To sustain the required level of performance while maintaining its user-friendly and open-source approach Smilei currently uses *HDF5*.

During preliminary studies done for the IDRIS *Grand Challenge* (see in Section 7.2), Smilei achieved a write bandwidth of 2.6 Gb/s on the Turing (BlueGene/Q) GPFS file system. The simulation domain consisted in a grid of size 30,720 x 15,360 cells, and 18 fields were written every 1755 timesteps (for a total of 135,000 timesteps). The amount of 60 Gb of data was written in 24 s for each of the selected timesteps.

4. Parallelization

As high-performance computing (HPC) systems are evolving toward the exascale, there is an admitted risk that today's algorithms and softwares will be subpar, at best, for the upcoming architectures. Manufacturers have been unable to improve the existing "standard" microprocessor technologies for the last decade. Instead, the trend is oriented toward the multiplication of the number of computing units by several orders of magnitude. This is achieved either using co-processors or massively multi-core processors. In order to face this emerging complexity, codes must expose a tremendous amount of parallelism while conserving data locality and minimizing load imbalance. In this Section, we first present the overall parallelization strategy chosen for Smilei, and follow with accurate descriptions of its elements.

4.1. Strategy

For the sake of generality, all fundamental computing items (cores, MPI processes, openMP threads, cuda threads, openCL work items, etc.) will be referred to as computing elements (CE) in this subsection.

The difficulty in parallelizing a PIC code lies in the coupling between the grid and particle aspects of the code. In a typical run, most of the load is carried by the particles. It is therefore very tempting to distribute particles equally between CEs: benefits would be huge. First, simplicity. No particle communications are required because particles only interact with fields and are independent from each other. Second, an almost perfect load balance is maintained at all times. The drawback of this approach is that it implies that all CEs have access to a shared global array of grid quantities (fields and currents). These accesses must be synchronized and require frequent global communications which, in practice, prevent any form of scalability above a couple hundreds of CEs.

A purely particle-based decomposition being impossible, we must apply a grid-based decomposition technique. Domain decomposition is the technique used in all state-of-the-art PIC codes such as Osiris [26] or Calder-Circ [27] in laser–plasma interaction or Photon–Plasma [28] in astrophysics.

It has shown very good scalability but comes with a cost. As most of the computational load is carried by particles, having a grid-based decomposition is inconvenient. Its efficient implementation is more involved, and load balance is very difficult to achieve. The biggest issue is that particles are volatile objects traveling throughout the entire domain, forcing (i) communications between CEs when particles cross their local domain boundary, and (ii) random access to the grid at every interpolation and projection phases. Communications are limited to neighbor domains and are not a fundamental threat to performance or scalability. In contrast, the randomness of the particle positions is much more

[6] IBM Knowledge center at http://www.goo.gl/XjUXzu.
[7] https://www.hdfgroup.org/HDF5.
[8] http://www.unidata.ucar.edu/software/netcdf/docs/index.html.

problematic. Random access to the grid arrays breaks the principle of data locality paramount to the performance via a good cache use. Conversely, a proper access to the data avoids multiple load operations when the same data is used several times. And on top of that, if the access is well organized, *Single Instruction Multiple Data* (SIMD) operations can be executed thus accelerating the computation by a significant amount.

Most of the time, this issue is addressed by sorting particles. Different kind of algorithms can ensure that particles close to each other in space are also well clustered in memory. Particles can be sorted at the cell level by a full count-sort algorithm every now and then during the simulation, or they can be subject to a more lax but more frequent sorting as proposed in Ref. [29]. Note that the domain decomposition technique is already a form of sorting. Particles of a given sub-domain are naturally stored in a compact array of memory and attached to the grid portion they can interact with. If each sub-domain is sufficiently small to fit in the cache, very good performances can be achieved. This approach was suggested in Refs. [30,31] and is the one used in Smilei. It consists in a very fine-grain domain decomposition referred to as "patch-based" decomposition where patches denote the very small sub-domains. In addition, Smilei still performs a very lightweight particle sorting within the patches, as in Ref. [29], in order to minimize cache misses. It brings a convenient flexibility in the patches size without loss of performances as quasi-particles remain well sorted even if the patches are large.

4.2. A patch-based MPI + openMP implementation

Smilei uses the Message Passing Interface (MPI) to communicate data between distinct nodes of the distributed-memory architecture, and the Open Multi-Processing (openMP) interface to harmonize the computational load within each node with a reduced programming complexity.

This section shows that this hybrid MPI + openMP implementation of a patch-based decomposition naturally extends the pure MPI one described in Ref. [31]. It provides both scalability and dynamic load balancing.

4.2.1. Patches distribution between MPI processes

The first layer of parallelism in Smilei is similar to the standard domain decomposition: the simulation box is divided into sub-domains that can be treated in parallel. In a standard "traditional" MPI approach, each MPI process handles one sub-domain. But in Smilei, the simulation box is divided into many more sub-domains than there are MPI processes. They are called "patches" specifically to make this distinction: each MPI process handles many patches. Note that the content of a patch is not different than that of a sub-domain: particles and a portion of the grid.

The obvious cost of this fine-grain domain decomposition is an additional, but necessary, synchronization between patches. Synchronization between patches belonging to the same MPI process is very cheap. It consists in a simple copy of a relatively small amount of ghost cells and exchange of particles in a shared memory system. Synchronization becomes more expensive when it occurs between patches belonging to different MPI processes. In that case, data has to be exchanged through the network between distributed memory systems via costly calls to the MPI library. In order to limit this cost, we need a distribution policy of the patches between the different MPI processes which minimizes MPI calls. This is achieved by grouping patches in compact clusters that reduce the interface between MPI sub-domains as much as possible. In addition, this policy must be flexible enough to support an arbitrary number of MPI processes and varying number of patches per process. In order to satisfy both compactness and flexibility, patches are ordered along a Hilbert space-filling curve [32]. An example of the Hilbert curve is given in Fig. 3. This curve is divided into as many segments as MPI processes and each process handles one of these segments. The mathematical properties of the Hilbert curve guarantee that these segments form compact clusters of patches in space (see Fig. 3) independently of their number or length.

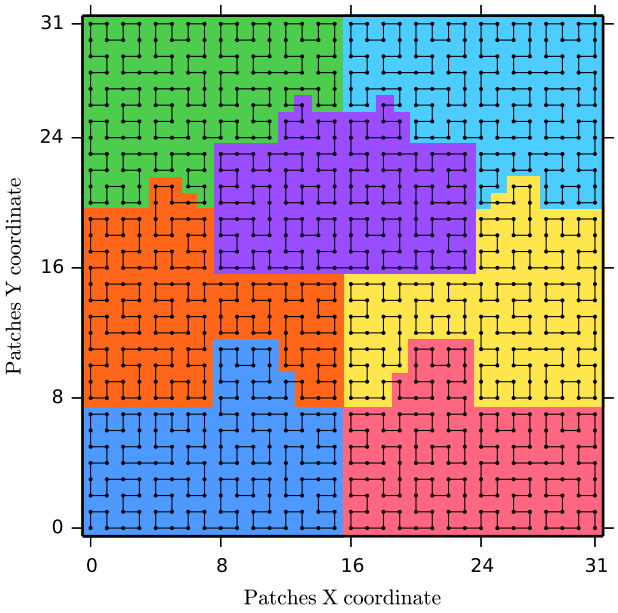

Fig. 3. Example of a 32×32 patches domain decomposition, shared between 7 MPI processes. MPI domains are delimited by different colors. The Hilbert curve (black line) passes through all the patch centers (black dots). It starts from the patch with coordinates $(0, 0)$ and ends at the patch with coordinates $(31, 0)$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.2.2. OpenMP parallelization and load balancing

Patch-based decomposition, in addition to its cache efficiency, is a very convenient way to expose a lot of local (inside MPI sub-domains) parallelism. Each patch being independent, they can be easily treated in parallel by the threads owned by the MPI process. Without this structure, the projection of particles might result in race conditions (threads overwriting each other's computation) and would require costly atomic operations.

In Smilei, patches are treated by openMP threads. In practice, this allows the user to start the simulation with less (but larger) MPI domains than in a pure MPI implementation. A similar level of computational performance is retained while decreasing the global amount of communications. The number of macro-particles per patch may differ significantly and so does the computational load associated to each patch. The use of the openMP dynamic scheduler therefore provides local load balancing at a reasonable cost. If a thread is busy treating a patch with a lot of macro-particles, other threads will be able to handle the remaining lighter patches thus avoiding idle time (see performance results in Section 4.3.2 and Fig. 5).

Patches also act as sorting structures. Indeed, quasi-particles of a given patch only interact with this patch's local grid. Small patches therefore provide a finer-grain load balancing and optimized cache use at the cost of more inter-patch synchronization. This cost is assessed in Section 4.3.5.

4.2.3. Load management

The objective of load management is to harmonize the computational workload between CEs as homogeneously as possible, in order to avoid idle, underloaded CEs waiting for overloaded

CEs. In Smilei, the load is dynamically balanced. Note that load balancing is not the only approach for load management: it can also involve load-limiting techniques such as the k-means particle-merging algorithm implemented in Photon–Plasma [33].

We have seen in Section 4.2.2 that openMP already provides some amount of load balancing at the node level, but it does not help managing the load between MPI processes. Smilei balances the load between MPI processes by exchanging patches (defined in Section 4.2.1). This technique is efficient because a single patch workload is much smaller than the total workload of a process. The patch size defines the balance grain and the smaller the patches the smoother the balance.

This is yet another argument in favor of using patches as small as possible. At this point, it becomes interesting to understand what limits the patch size. The minimum size of a patch is dictated by the number of ghost cells used. We consider reasonable that a patch must have more cells than ghost cells. The number of ghost cells is defined by the order of Maxwell's equations discretization scheme, and by the shape function of the macro-particles. A standard second-order Yee scheme, for instance, uses 4 ghost cells per dimension (2 on each side). The minimum patch size in that case is therefore 5 cells per dimension. This criteria also guarantees that ghost cells from non-neighbor patches do not overlap, which is convenient for the synchronization phases. The influence of the patch size is illustrated in Section 4.3.

We have seen in Section 4.2.1 that patches are organized along a Hilbert space-filling curve divided into as many segments of similar length as there are MPI processes. Each process handles the patches located in its segment of the Hilbert curve. Dynamically balancing the load simply consists in exchanging patches between neighbor MPI processes along the curve. That is to lengthen or shorten the segments depending on how loaded they are. When an MPI process is overloaded, it sends patches to its neighbors along the Hilbert curve; therefore its segment becomes shorter. Inversely, an underloaded process will receive patches from its neighbors; its segment becomes longer.

The following describes the dynamic load-balancing algorithm (it is summarized in Table 3). First, the computational load $P[p]$ of each patch p is evaluated as

$$P[p] = N_{\text{part}} + C_{\text{cell}} \times N_{\text{cells}} + C_{\text{frozen}} \times N_{\text{frozen}} \tag{28}$$

where N_{part} is the number of active particles in the patch, N_{cells} is the number of cells in the patch, N_{frozen} is the number of frozen (immobile) particles in the patch, and C_{cell} and C_{frozen} are user-defined coefficients representing the computational cost of cells (mostly solving Maxwell equation) and frozen particles. In most cases, the active particles are the major source of computational load. By default Smilei uses $C_{\text{cell}} = 1$ and $C_{\text{frozen}} = 0.1$. The total computational load of MPI rank r is $L[r] = \Sigma_p P[p]$ (sum over all the patches owned by r) and the total computational load of the simulation is $L_{\text{tot}} = \Sigma_r L[r]$. The optimal computational load per process $L_{\text{opt}} = L_{\text{tot}}/N_{\text{MPI}}$, where N_{MPI} is the number of MPI processes. The balancing algorithm proceeds to a new decomposition of the Hilbert curve so that each segment carries a load as close to L_{opt} as possible. This balancing process is typically done every 20 iterations in order to follow the dynamics of the simulation. Frequent and small corrections give superior performance than rare and dramatic adjustments (see Fig. 8). The three global communications required involve only a single number by MPI rank thus minimizing the total volume of data exchanged and memory occupation. Sharing the details of the computational load of each patch is needed only between two consecutive MPI ranks and is done via local communications.

The amplitude of the readjustment is limited: each MPI process keeps at least one of its original patches. This reduces the performance impact of strong, high-frequency, oscillatory variations of the load observed in some cases. Once the segments are defined, the actual exchange of data is performed if necessary.

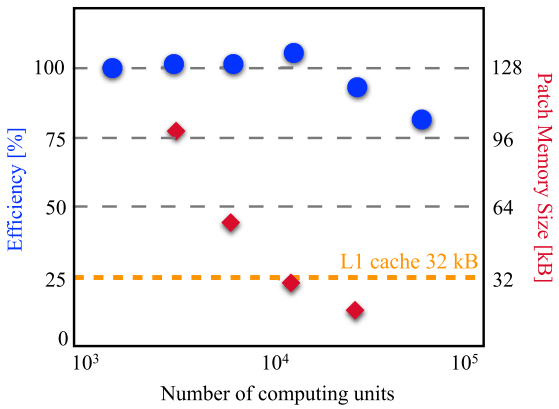

Fig. 4. Pure MPI strong scaling of Smilei in a homogeneous plasma case on the CINES/Occigen system. For this specific test case, the MPI domain size becomes smaller than the L1 cache around 20,000 cores. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.3. Performances and scaling

This section illustrates the efficiency of the chosen parallelization strategy and gives some insight on the optimization of the numerical parameters available to the user.

4.3.1. MPI

We study here the case of an MPI-only parallelization. The series of simulations presented here were performed on the CINES/Occigen system (Bull) and focused a physics study devoted to Brillouin amplification of short laser pulses (see Section 7.2), for which the plasma remains rather homogeneous throughout the simulation. Fig. 4 displays Smilei's strong scaling for a pure MPI parallelization. The same simulation is run on different number of cores and a single MPI process is attached to each core. As the number of cores increases, the size of the data handled by each core, or "domain size", decreases because the global domain is divided between all cores. The efficiency remains close to 100% as long as the domain size remains larger or equal to the L1 cache size. For the specific global domain size used in this test, this occurs around 20,000 cores. As the domain size approaches the L1 size, an improved cache-use slightly improves the performances. Using a larger number of MPI processes then decreases the efficiency as the domain size becomes significantly smaller than the cache. At this point, the system computing units occupation is too small to deliver proper performances, and the cost of additional MPI communications starts being significant. Fig. 4 illustrates the fact that the pure MPI decomposition performs well in Smilei up to the optimal (for this given simulation set-up) number of MPI domains. There is no significant overhead due to MPI computations, their costs being much smaller than the computation in a standard case.

In summary, MPI parallelization is good at handling homogeneous plasmas as long as the MPI domain sizes are not too small with respect to the L1 cache.

4.3.2. MPI + openMP

In this section we present the performances achieved with the hybrid MPI+openMP parallelization described in Section 4.2, when the plasma does not remain homogeneous.

The case study is now, and until the end of the section, an ultra-high-intensity laser propagating in a plasma. It is a typical laser wakefield acceleration case, well known for being strongly impacted by load imbalance [34]. It is a two-dimensional simulation

Table 3
Load balancing algorithm used in Smilei. After initialization, a segment of patches along the space filling curve is attributed to each MPI rank. The number of patches handled by each MPI process is then periodically reevaluated according to the following algorithm. The table shows the list of operations executed by MPI rank R_{MPI} as it updates its number of patches N. (*) shows operations requiring global communications.

Initialization	
N_{MPI} = total number of MPI processes.	
R_{MPI} = my MPI rank.	
$P[p]$ = computational load of my p^{th} patch	see eq. 28
$L[R_{\mathrm{MPI}}]$ = my total computational load	$L[R_{\mathrm{MPI}}] = \sum_p P[p]$
Compute the simulation total computational load*	$L_{\mathrm{tot}} = \sum_{r=0}^{r=N_{\mathrm{MPI}}-1} L[r]$
Compute the optimal load	$L_{\mathrm{opt}} = L_{\mathrm{tot}}/N_{\mathrm{MPI}}$
Set number of patches in my current segment	$N = \mathrm{size(P)}$
If $R_{\mathrm{MPI}} > 0$: Check exchanges with MPI rank $R_{\mathrm{MPI}} - 1$	
$P_{\mathrm{left}}[p]$ = computational load of the p^{th} patch of $R_{\mathrm{MPI}} - 1$	
Set target T to optimal begining for my rank	$T = L_{\mathrm{opt}} \times R_{\mathrm{MPI}}$
Set current load C_L to actual begining of my rank*	$C_L = \sum_{r=0}^{r=R_{\mathrm{MPI}}-1} L[r]$
If $C_L > T$, I may have to take patches from $R_{\mathrm{MPI}} - 1$	
Set index j to index of last patch of $R_{\mathrm{MPI}} - 1$	$j = \mathrm{size(P_{left})} - 1$
While $\lvert C_L - T\rvert > \lvert C_L - P_{\mathrm{left}}[j] - T\rvert$ and $j > 0$:	
I take a patch from $R_{\mathrm{MPI}} - 1$	$C_L = C_L - P_{\mathrm{left}}[j]$
I own one more patch	$N = N + 1$
Scan next patch	$j = j - 1$
Else, I may have to give patches to $R_{\mathrm{MPI}} - 1$	
Set index j to index of my first patch	$j = 0$
While $\lvert C_L - T\rvert > \lvert C_L + P[j] - T\rvert$ and $j < \mathrm{size(P)} - 1$:	
I give a patch to $R_{\mathrm{MPI}} - 1$	$C_L = C_L + P[j]$
I own one less patch	$N = N - 1$
Scan next patch	$j = j + 1$
If $R_{\mathrm{MPI}} < N_{\mathrm{MPI}} - 1$: Check exchanges with MPI rank $R_{\mathrm{MPI}} + 1$	
$P_{\mathrm{right}}[p]$ = computational load of the p^{th} patch of $R_{\mathrm{MPI}} + 1$	
Set target T to optimal end for my rank	$T = L_{\mathrm{opt}} \times (R_{\mathrm{MPI}} + 1)$
Set current load C_L to actual end of my rank	$C_L = \sum_{r=0}^{r=R_{\mathrm{MPI}}} L[r]$
If $C_L < T$, I may have to take patches from $R_{\mathrm{MPI}} + 1$	
Set index j to index of first patch of $R_{\mathrm{MPI}} + 1$	$j = 0$
While $\lvert C_L - T\rvert > \lvert C_L + P_{\mathrm{right}}[j] - T\rvert$ and $j < \mathrm{size(P_{right})} - 1$:	
I take a patch from $R_{\mathrm{MPI}} + 1$	$C_L = C_L + P_{\mathrm{right}}[j]$
I own one more patch	$N = N + 1$
Scan next patch	$j = j + 1$
Else, I may have to give patches to $R_{\mathrm{MPI}} + 1$	
Set index j to index of my last patch	$j = \mathrm{size(P)} - 1$
While $\lvert C_L - T\rvert > \lvert C_L - P[j] - T\rvert$ and $j > 0$:	
I give a patch to $R_{\mathrm{MPI}} + 1$	$C_L = C_L - P[j]$
I own one less patch	$N = N - 1$
Scan next patch	$j = j - 1$
Communicate my new number of patches to all other MPI ranks*	Broadcast(N)

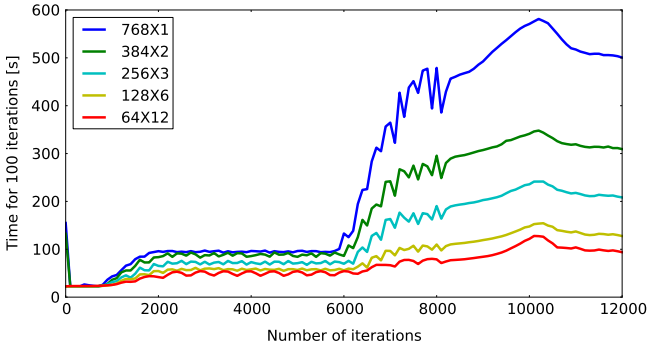

Fig. 5. OpenMP load balancing effect. The plot displays the evolution of the wall-clock time necessary to complete 100 iterations as a function of the number of iterations already completed. The legend shows the total number of MPI processes and number of openMP threads per MPI process in the format MPI × openMP.

consisting of 1024 × 128 patches (except in Section 4.3.5 where this parameter varies), each having 8 × 5 cells and 196 particles per cell. The global box size is therefore 8192 × 640 cells for a total of more than a billion particles. The plasma is homogeneous with a density of $4.94 \times 10^{-4} n_c$, where n_c is the critical density. The laser has both transverse and longitudinal Gaussian profiles and its strength parameter is $a_0 = 8$. The scale of the simulation is based on the laser's inverse wavenumber [$k_0^{-1} = \lambda_0/(2\pi)$, see Section 2.2]. For laser wakefield electron acceleration, a standard value for the laser wavelength is $\lambda_0 = 0.8$ μm. In that case, the electron number density is 8.6×10^{17} cm^{-3}, the waist is $w_0 = 18$ μm and the pulse full width half-maximum is $\tau_0 = 35.3$ fs. The longitudinal cell size is $\Delta x = 0.125/k_0$ and the transverse cell size is $\Delta y = 1.5/k_0$. The timestep is $\Delta t = 0.124/\omega_0$, where $\omega_0 = ck_0$ is the laser angular frequency.

Each run ran on 32 nodes of the OCCIGEN system. This represents 64 processors of 12 cores each for a total of 768 cores. The plasma is initially homogeneous but load imbalance gradually builds up, then rises quickly after 6000 iterations before stabilizing.

Fig. 5 shows the evolution of the wall-clock time necessary to complete 100 iterations as a function of the number of iterations already completed for different numerical settings. The runs only differ by the number of openMP threads per MPI process and total number of MPI processes. The total number of threads is kept constant and equal to 768 in order to have 1 thread per core. The openMP dynamic scheduler is used in all cases.

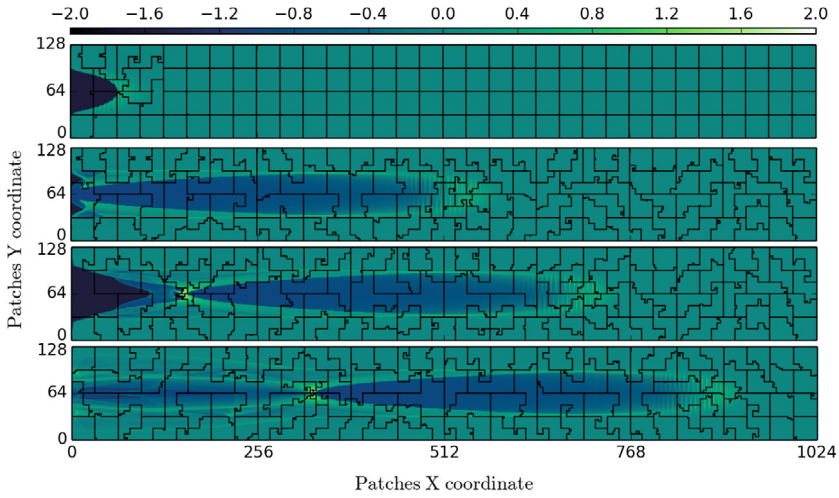

Fig. 6. Evolution of the MPI domains shapes with respect to the computational load distribution. The colormap indicates the local imbalance $I_{loc} = \log_{10}(L_{loc}/L_{av})$ where L_{loc} is the local patch computational load and L_{av} the average computational load. Black lines delimit the different MPI domains. The laser enters an initially homogeneous plasma from the left side of the box and propagates toward the right. The 4 panels show the entire simulation domain after 1600, 5480, 6820, 9080 iterations from top to bottom. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Several interesting features can be noticed on Fig. 5. First, as long as the plasma is relatively homogeneous (first 1000 iterations), all runs perform similarly. It means that the overhead for having a hybrid parallelization is negligible in this situation. Later in the simulation, the pure-MPI case shows an extreme sensitivity to the load imbalance. The wall-clock time spent to perform 100 iterations is almost multiplied by 20 with respect to the initial homogeneous plasma. Cases using more than one openMP thread per MPI process are much less sensitive to this effect. And the more threads per MPI process, the smoother the performances. This is perfectly in line with the local load balancing analysis given in Section 4.2.2. Nevertheless, even in the best case 64 × 12, a performance loss of a factor superior to 4 is still impacting the simulation. This is explained by the fact that openMP can only balance the load within a given MPI domain. Imbalance across MPI domains will keep slowing the simulation down.

Using more openMP threads, or equivalently more cores, per MPI process allows the use of larger MPI domains and therefore provides a better load balancing. But the number of openMP threads is limited to the number of cores accessible on the shared memory system. In our case, this is a single OCCIGEN node made of two processors of 12 cores each so up to 24 openMP threads could be used. But going from 12 to 24 openMP threads per MPI process results in a drop of the performances because of the synchronization required between the two processors of the node. The best performances are achieved when a single MPI process is given to each processor and when all cores of the processor are managed by the openMP scheduler. The quality of the load balancing via the openMP dynamic scheduler thus directly depends on the size (in number of cores) of the processors composing the nodes.

4.3.3. MPI + openMP + dynamic load balancing

This section presents results obtained with the dynamic load balancing (DLB) algorithm described in Section 4.2.3. With DLB activated, the MPI domains are now capable of exchanging patches and therefore their shape evolves with respect to the computational load distribution. Fig. 6 shows this distribution and the corresponding evolution of the shape of the MPI domains. As expected, they tend to become smaller in areas where the computational load is high and, reversely, larger where the patches are underloaded. The least loaded patches have approximately 1% of the average patch load. These under loaded patches are empty of particles and their computational load is limited to solving the maxwell equations. On the opposite, the most loaded patches have almost 100 times as much computational load as the average and their load is completely dominated by particles. Note that the computational load map looks very much like the density map for the simple reason that the computational load is mostly carried by the macro-particles as in most PIC simulations.

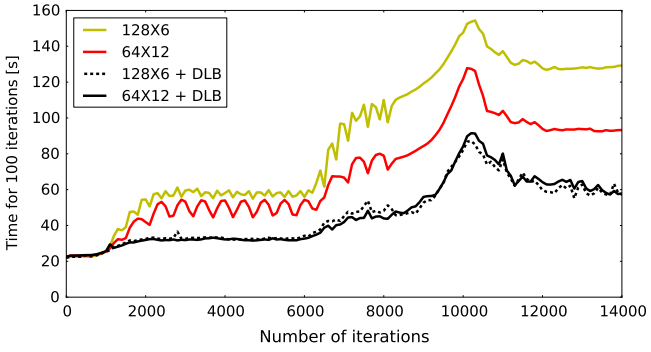

Fig. 7. Dynamic load balancing (DLB) algorithm effect. The plot displays the evolution of the wall-clock time necessary to complete 100 iterations as a function of the number of iterations already completed. The legend shows the total number of MPI processes and number of openMP threads per MPI process in the format MPI × openMP. The red and yellow curves are replicas of Fig. 5. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 7 shows a performance comparison between the two best cases obtained in the previous section (without DLB) and the same cases with DLB activated.

The balancing here is done every 20 iterations and $C_{cell} = 2$. No difference is observed during the balanced stage of the run (first 1000 iterations). As expected, the cost of the balancing is negligible when actual balancing is not required. In the imbalanced stage of the run, DLB provides an additional gain of almost 40% with respect to the previous best case "64 × 12". A side benefit is also to reduce the dependency on the large number of openMP threads. Indeed, it appears that almost similar results are obtained with only 6 openMP threads when DLB is active. As DLB balances the load between MPI processes, the local balancing via openMP becomes much less critical than before. Note that the openMP parallelization remains necessary for an efficient fine grain balancing but it can be achieved with only a limited number of threads thus removing the dependency on a large shared memory hardware.

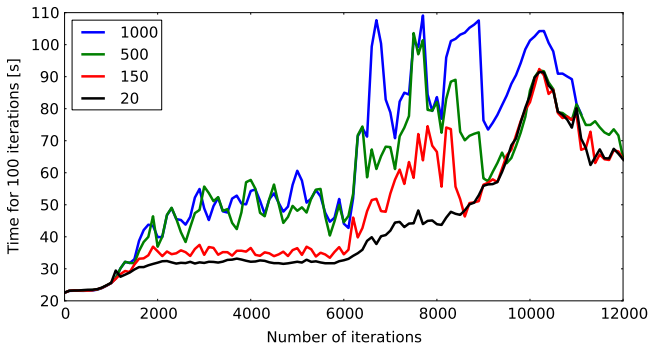

Fig. 8. Evolution of the wall-clock time necessary to complete 100 iterations, as a function of the number of completed iterations, for four different values of N_b (the number of iterations between two load-balancing events). The black curve is a replica of Fig. 7. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

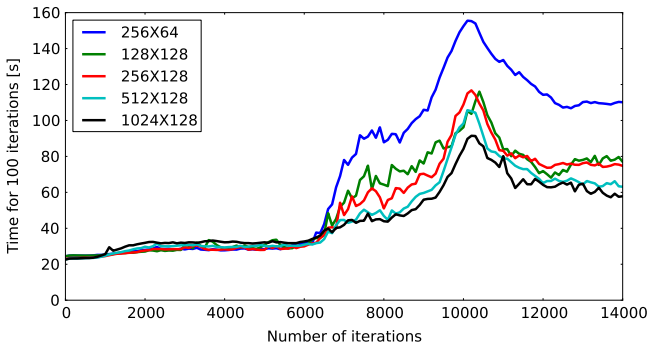

Fig. 9. Evolution of the wall-clock time necessary to complete 100 iterations as a function of the number of iterations already completed, for various number of patches. The legend shows N_{patches} in the format $N_x \times N_y$ where N_x and N_y are respectively the number of patches in the x and y directions. The black curve is a replica of Fig. 7. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Note also that the cost of the imbalance is still significant in spite of all the efforts to balance the load. The additional cost is mainly due to the imbalance of the particles communication cost which is not as well balanced as the computational cost of particles.

4.3.4. Balancing frequency

The load is balanced every N_b iterations. Fig. 8 shows the influence of this parameter. As expected, as N_b decreases, the load balance gets more accurate and the performances increase. A more frequent balancing also means smaller adjustments each time. Consequently, the overhead of load balancing remains low, even for low N_b. The cost of load balancing has a negative impact on performances only when N_b is much lower than the number of iterations over which imbalance builds up.

4.3.5. Number of patches

The number of patches is an important parameter. It influences the quality of both the openMP implementation and DLB. A large number of patches allows for finer-grain openMP parallelization and DLB, but also implies more ghost cells and synchronization costs.

Fig. 9 shows several interesting features. First, while imbalance is weak (between iterations 2000 and 6000), having more patches noticeably costs additional synchronization. The cost of particles dynamics far outweighs the cost of synchronization, but this overhead is measurable. On the other hand, in the second stage of the simulation where a strong imbalance kicks in (after iteration 6000), having smaller and more numerous patches is clearly beneficial.

Another interesting result comes from the comparison between the 256×64 and 128×128 cases. Despite an equal number of patches, much better performances are achieved in the latter topology. This can be explained by the way the Hilbert curve is generated. The case of a "square" topology ($N_x = N_y$, N_x being a power of 2) corresponds to the usual Hilbert curve. If the number of patches is larger in one direction, the Hilbert curve is generated over a square of the smaller dimension's size, then repeated along the longer dimension. The constraint on N_x and N_y being powers of 2 remains but they can be different. The cost of this generalization is that the resulting space-filling curve loses some of its compactness. This translates into additional synchronization cost.

5. Additional modules

To answer to the users various needs, additional modules have been implemented in Smilei.

5.1. Electric field and current density filters

Particle-in-Cell codes relying on the FDTD Maxwell solvers are known to face serious problems when dealing with relativistic beams of particles and/or relativistically drifting plasmas [35] such as encountered in laser wakefield acceleration [36] or in relativistic astrophysics simulation [37]. Numerical dispersion indeed results in a spurious, direction-dependent reduction of the light waves velocity. As a result, ultra-relativistic particles may artificially catch up with the light waves giving rise to the grid-Cerenkov instability [35]. Various methods have been proposed to deal with this instability: in particular, time-filtering of the electric fields [38] and spatial-filtering of the current density [39].

Smilei specifically uses the Friedman time filter on the electric fields [38]. If required by the user, this filter consists in replacing the electric field in the Maxwell–Faraday solver by a time-filtered field:

$$\mathbf{E}^{(n)} = \left(1+\frac{\theta}{2}\right)\mathbf{E}^{(n)} - \left(1-\frac{\theta}{2}\right)\mathbf{E}^{(n-1)} + \frac{1}{2}(1-\theta)^2\,\bar{\mathbf{E}}^{(n-2)}, \quad (29)$$

where $\bar{\mathbf{E}}^{(n-2)} = \mathbf{E}^{(n-2)} + \theta\,\bar{\mathbf{E}}^{(n-3)}$, and the filtering parameter $\theta \in [0, 1]$ is an input parameter defined by the user.

A multi-pass bilinear filter on the current density has also been implemented [39]. Each pass consists in a 3-points spatial averaging (in all spatial dimensions) of the current, so that the filtered current density (here defined at location i on a one-dimensional grid) is recomputed as:

$$J_i^f = \frac{1}{2}J_i + \frac{J_{i+1}+J_{i-1}}{4}. \quad (30)$$

Current filtering, if required by the user, is applied before solving Maxwell's equation, and the number of passes is an input parameter defined by the user.

Both methods can be used together or separately and have allowed to satisfactorily reduce the numerical grid-Cerenkov instability when dealing with relativistically drifting electron–positron plasmas in the framework of collisionless shock studies (see Section 7.4).

5.2. Antennas

After the particle projection and before the Maxwell solver execution, custom additional currents can be introduced. These additional, user-defined, currents are referred to as *antennas* in Smilei. The user provides both spatial and temporal profiles for chosen currents J_x, J_y and/or J_z. Antennas may be used, for instance,

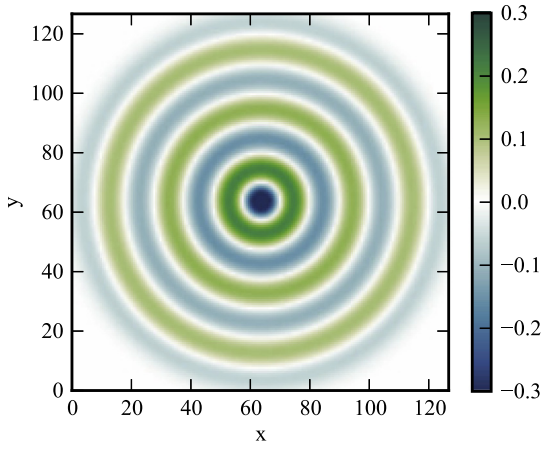

Fig. 10. E_z electric field (at $t = 60$) generated by an oscillating J_z source placed at the center of an empty box.

to apply external electromagnetic sources anywhere in the box. An example is provided in Fig. 10, showing the electric field induced by an oscillating J_z current applied within a small circular region in the center of an empty box. A circular wave is generated from the antenna and propagates outwards.

5.3. Field ionization

Field ionization is a process of particular importance for laser–plasma interaction in the ultra-high intensity regime. It can affect ion acceleration driven by irradiating a solid target with an ultra-intense laser [40], or can be used to inject electrons through the accelerating field in laser wakefield acceleration [41]. This process is not described in the standard PIC (Vlasov–Maxwell) formulation, and an *ad hoc* description needs to be implemented. A Monte-Carlo module for field ionization has thus been developed in Smilei, closely following the method proposed by Nuter et al. [40].

5.3.1. Physical model

This scheme relies on the quasi-static rate for tunnel ionization derived in Refs. [42–44]. Considering an ion with atomic number Z being ionized from charge state $Z^\star$ to $Z^\star + 1 \leq Z$ in an electric field $\mathbf{E}$ of magnitude $|E|$, the ionization rate reads:

$$\Gamma_{Z^\star} = A_{n^\star,l^\star}\, B_{l,|m|}\, I_{Z^\star} \left(\frac{2(2I_{Z^\star})^{3/2}}{|E|}\right)^{2n^\star-|m|-1} \times \exp\left(-\frac{2(2I_{Z^\star})^{3/2}}{3|E|}\right), \tag{31}$$

where $I_{Z^\star}$ is the $Z^\star$ ionization potential of the ion, $n^\star = (Z^\star + 1)/\sqrt{2I_{Z^\star}}$ and $l^\star = n^\star - 1$ denote the effective principal quantum number and angular momentum, and l and m denote the angular momentum and its projection on the laser polarization direction, respectively. $\Gamma_{Z^\star}$, $I_{Z^\star}$ and E are here expressed in atomic units.[9] The coefficients $A_{n^\star,l^\star}$ and $B_{l,|m|}$ are given by:

$$A_{n^\star,l^\star} = \frac{2^{2n^\star}}{n^\star\,\Gamma(n^\star + l^\star + 1)\,\Gamma(n^\star - l^\star)}, \tag{32a}$$

$$B_{l,|m|} = \frac{(2l+1)(l+|m|)!}{2^{|m|}|m|!(l-|m|)!}, \tag{32b}$$

[9] $\Gamma_{\rm qs}$ is in units of $\hbar/(\alpha^2 m_e c^2)$ with $\hbar$ the Planck constant and α the fine-structure constant. $I_{Z^\star}$ is in units of $\alpha^2 m_e c^2$ (also referred to as Hartree energy) and E is in unit of $\alpha^3 m_e^2 c^3/(e\hbar)$.

where $\Gamma(x)$ is the gamma function. Note that considering an electric field $E = |E|\cos(\omega t)$ oscillating in time at the frequency ω, averaging Eq. (31) over a period $2\pi/\omega$ leads to the well-known cycle-averaged ionization rate:

$$\Gamma_{\rm ADK} = \sqrt{\frac{6}{\pi}} A_{n^\star,l^\star}\, B_{l,|m|}\, I_{Z^\star} \left(\frac{2(2I_{Z^\star})^{3/2}}{|E|}\right)^{2n^\star-|m|-3/2} \times \exp\left(-\frac{2(2I_{Z^\star})^{3/2}}{3|E|}\right). \tag{33}$$

In Smilei, following Ref. [40], the ionization rate Eq. (31) is computed for $|m| = 0$ only. Indeed, as shown in Ref. [44], the ratio R of the ionization rate computed for $|m| = 0$ by the rate computed for $|m| = 1$ is:

$$R = \frac{\Gamma_{Z^\star},\, |m|=0}{\Gamma_{Z^\star},\, |m|=1} = 2\frac{(2\,I_{Z^\star})^{3/2}}{|E|} \simeq 7.91\,10^{-3}\,\frac{(I_{Z^\star}[\mathrm{eV}])^{3/2}}{a_0\,\hbar\omega_0[\mathrm{eV}]}, \tag{34}$$

where, in the practical units formulation, we have considered ionization by a laser with normalized vector potential $a_0 = e|E|/(m_e c\omega_0)$, and both the ionization potential $I_{Z^\star}$ and the photon energy $\hbar\omega_0$ in eV. Typically, ionization by a laser with wavelength 1 μm (correspondingly $\hbar\omega_0 \sim 1$ eV) occurs for values of $a_0 \ll 1$ (even for large laser intensities for which ionization would occur during the rising time of the pulse) while the ionization potential ranges from a couple of eV (for electrons on the most external shells) up to a few tens of thousands of eV (for electrons on the internal shell of high-Z atoms). As a consequence, $R \gg 1$, and the probability of ionization of an electron with magnetic quantum number $|m| = 0$ greatly exceeds that of an electron with $|m| = 1$.

Finally, it should be stressed that simulations involving field ionization (the same is true for those involving binary collisions and/or collisional ionization as detailed in the next two Sections 5.4 and 5.5) cannot be arbitrarily scaled. The reference time normalization ω_r^{-1} needs to be prescribed. In Smilei, this is done at initialization, the user having to define the reference angular frequency in SI units whenever one of this additional module is used.

5.3.2. Monte-Carlo procedure

In Smilei, tunnel ionization is treated for each species (defined by the user as subject to field ionization) right after field interpolation and before applying the pusher. For all quasi-particles (henceforth referred to as quasi-ions) of the considered species, a Monte-Carlo procedure has been implemented that allows to treat multiple ionization events in a single timestep. It relies on the cumulative probability derived in Ref. [40]:

$$F_k^{Z^\star} = \sum_{j=0}^{k} P_j^{Z^\star}, \tag{35}$$

to ionize from 0 to k times a quasi-ion with initial charge state $Z^\star$ during a simulation timestep Δt, $P_j^{Z^\star}$ being the probability to ionize exactly j times this ion given by:

$$P_k^i = \begin{cases} \bar{P}^i \quad \text{if} \quad k = 0 \\ \displaystyle\sum_{p=0}^{k-1} R_{i+p}^{i+k}\left(\bar{P}^{i+k} - \bar{P}^{i+p}\right) \prod_{j=0,j\neq p}^{k-1} R_{i+j}^{i+p} \\ \quad \text{if} \quad 0 < k < k_{\max} \\ \displaystyle\sum_{p=0}^{k-1}\left[1 + R_{i+p}^{i+k}\left(\frac{\Gamma_{i+k}}{\Gamma_{i+p}}\bar{P}^{i+p} - \bar{P}^{i+k}\right)\right] \prod_{j=0,j\neq p}^{k-1} R_{i+j}^{i+p} \\ \quad \text{if} \quad k = k_{\max}, \end{cases} \tag{36}$$

with $\bar{P}^i = \exp(-\Gamma_i \Delta t)$ the probability to *not* ionize an ion in initial charge state i, and $R_\alpha^\beta = (1 - \Gamma_\beta/\Gamma_\alpha)^{-1}$ with Γ_i the ith ionization rate given by Eq. (31).

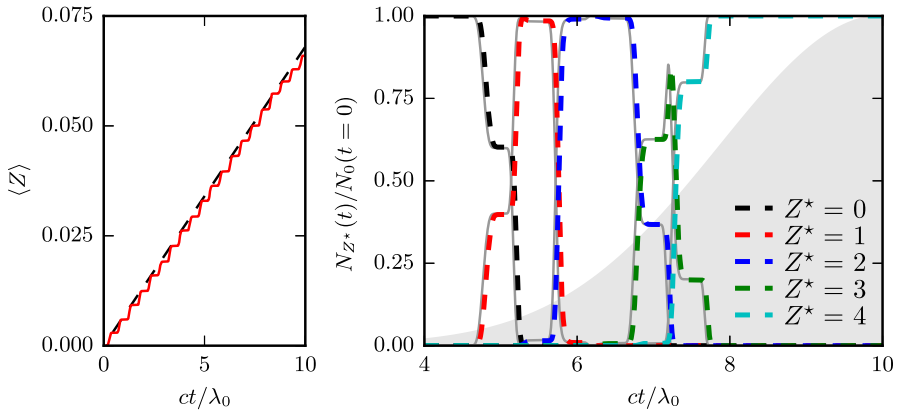

Fig. 11. Results of two benchmarks for the field ionization model. Left: Average charge state of hydrogen ions as a function of time when irradiated by a laser. The red solid line corresponds to PIC results, the dashed line corresponds to theoretical predictions using the cycle-averaged ADK growth rate of Eq. (33). Right: Relative distribution of carbon ions for different charge states as a function of time. Dashed lines correspond to PIC results, thin gray lines correspond to theoretical predictions obtained from Eq. (38). The Gaussian gray shape indicates the laser electric field envelope. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The Monte-Carlo scheme proceeds as follows. A random number r with uniform distribution between 0 and 1 is picked. If r is smaller than the probability $P_0^{Z^\star}$ to not ionize the quasi-ion, then the quasi-ion is not ionized during this time step. Otherwise, we loop over the number of ionization events k, from $k = 1$ to $k_{\max} = Z - Z^\star$ (for which $F_{k_{\max}}^{Z^\star} = 1$ by construction), until $r < F_k^{Z^\star}$. At that point, k is the number of ionization events for the quasi-ion. A quasi-electron is created with the numerical weight equal to k times that of the quasi-ion, and with the same velocity as this quasi-ion. The quasi-ion charge is also increased by k.

Finally, to account for the loss of electromagnetic energy during ionization, an *ionization current* $\mathbf{J}_{\text{ion}}$ is projected onto the simulation grid [40,45] such that

$$\mathbf{J}_{\text{ion}} \cdot \mathbf{E} = \Delta t^{-1} \sum_{j=1}^{k} I_{Z^\star+k-1}. \tag{37}$$

5.3.3. Benchmarks

In what follows, we present two benchmarks of the field ionization model implemented in Smilei. Both benchmarks consist in irradiating a thin (one cell long) neutral material (hydrogen or carbon) with a short (few optical-cycle long) laser with wavelength $\lambda_0 = 0.8\,\mu\text{m}$.

In the first benchmark, featuring hydrogen, the laser intensity is kept constant at $I_L = 10^{14}\ \text{W/cm}^2$, corresponding to a normalized vector potential $a_0 \simeq 6.81 \times 10^{-3}$, over 10 optical cycles. The resulting averaged ion charge in the simulation is presented as a function of time in Fig. 11, left panel. It is found to be in excellent agreement with the theoretical prediction (dashed in Fig. 11, left panel) considering the cycle averaged ionization rate $\Gamma_{\text{ADK}} \simeq 2.55 \times 10^{12}\ \text{s}^{-1}$ computed from Eq. (33).

The second benchmark features a carbon slab. The laser has a peak intensity $I_L = 5 \times 10^{16}\ \text{W/cm}^2$, corresponding to a normalized vector potential $a_0 \simeq 1.52 \times 10^{-1}$, and a Gaussian time profile with full-width-at-half-maximum (FWHM) $\tau_L = 5\ \lambda_0/c$ (in terms of electric field). Fig. 11, right panel shows, as function of time, the relative distribution of carbon ions for different charge states (from 0 to +4). These numerical results are shown to be in excellent agreement with theoretical predictions obtained by numerically solving the coupled rate equations on the population N_i of each level i:

$$\frac{dN_i}{dt} = (1 - \delta_{i,0})\,\Gamma_{i-1}\,N_{i-1} - (1 - \delta_{i,Z})\,\Gamma_i\,N_i, \tag{38}$$

with $\delta_{i,j}$ the Kronecker delta, and Γ_i the ionization rate of level i. Note also that, for this configuration, $\Delta t \simeq 0.04$ fs is about ten times larger than the characteristic time $\Gamma_{\text{ADK}}^{-1} \simeq 0.006$ fs to ionize C^{2+} and C^{3+} so that multiple ionization from C^{2+} to C^{4+} during a single timestep does occur and is found to be correctly accounted for in our simulations.

5.4. Binary collisions

As detailed in Section 2, the PIC method aims at describing the self-consistent evolution of a collisionless plasma by solving the coupled system of Vlasov–Maxwell equations (1)–(3). Consequently, PIC codes must introduce additional modules to account for collisions. In Smilei, the effects of relativistic collisions have been implemented following the scheme described in Ref. [46]. It is based on Nanbu's approach [47], with the addition of a few enhancements: relativistic particles, low-temperature correction to the collision rate, and variable Coulomb logarithm. We briefly review this scheme here and illustrate it with typical applications.

Nanbu's theory first considers real particles, assuming that collisions occur many times during one time-step, and that each collision introduces a deflection angle $\theta \ll 1$ (although the total deflection angle may be large). By simulating a large number of Coulomb collisions, he finds that the total deflection angle $\langle\chi\rangle$ is well described by a unique function of $s = \langle\theta^2\rangle N/2$, where $\langle\theta^2\rangle$ is the expectation of θ^2 and N is the number of collisions during one time-step ($N \gg 1$). He also provides a probability density function $f(\chi)$ to pick randomly the deflection angle χ accumulated during one time-step.

Colliding each quasi-particle with all other quasi-particles nearby would be time-consuming. Instead, quasi-particles are randomly paired so that each collides with only one other quasi-particle at a given time-step. After many time-steps, each of them will have sampled the overall distribution of target particles. This pairing follows Ref. [48]. It is split in two cases: *intra*-collisions, when a group of particles collides with itself, and *inter*-collisions, when two distinct groups of particles collide. Intra-collisions can occur, for example, within all the electrons in the plasma. In this case, the group is split in two halves and one half is randomly shuffled to provide random pairs. In the case of inter-collisions, the two halves are simply the two groups: only one group is randomly shuffled. When the two halves do not contain the same number of particles, the extra particles (not paired yet) are randomly assigned a companion particle from those which have already been paired. Thus, one particle may participate in several pairs.

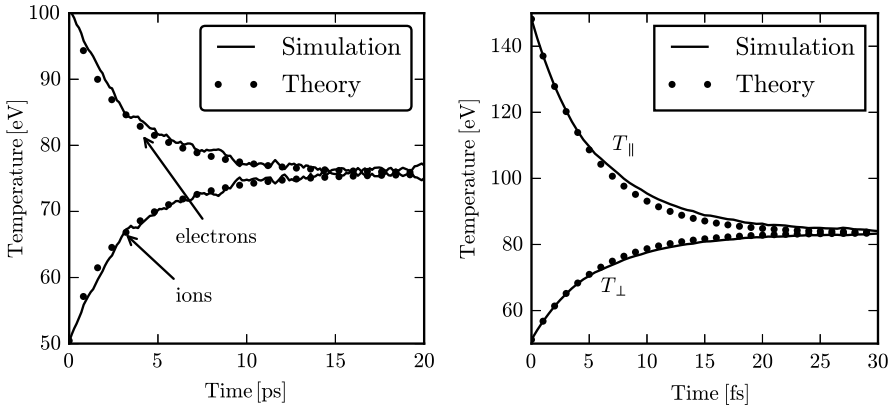

Fig. 12. Left: thermalization by collisions between ions and electrons of an hydrogen plasma. Right: temperature isotropization of an electron plasma.

Whereas many codes naturally make quasi-particles collide within their own cell, Smilei makes them collide with all those in the same patch. This is only accurate when the plasma parameters do not vary significantly within one patch, but it greatly reduces the amount quasi-particle sorting required.

The parameter s is normally calculated from the point of view of one particle traversing a cloud of numerous target particles. However, this picture is broken by the quasi-particles of the PIC code having variable weights. To ensure a deflection angle common to both quasi-particles (in the center-of-mass frame), thus momentum conservation, the parameter s must be the same from both quasi-particles' point of views. Unfortunately, this condition is not fulfilled when the weights or densities are different. Ref. [48] provides a detailed solution consisting in modifying s by a factor which makes it symmetric when exchanging the quasi-particles in a pair. This modification is later compensated by randomly picking quasi-particles which will not actually undergo a deflection, so that energy and momentum are conserved in average.

In addition to these considerations, Ref. [46] provides the relativistic expressions of s and χ, specifying the relativistic changes of frames, and gives corrections for low-temperature plasmas and a varying Coulomb logarithm. Note that these expressions, just like those of the field ionization module, cannot be normalized to dimension-less equations when using Smilei's units: the value of the reference frequency ω_r must be specified in the SI system of units.

As a first example of the possible effects of collisions, let us consider the thermalization between ions and electrons: a fully-ionized hydrogen plasma of density 10^{22} cm^{-3} is set with an ion temperature of 50 eV and an electron temperature of 100 eV. The left panel of Fig. 12 shows the evolution of both temperatures due to the e–i collisions, well matched by the theoretical solution taken from Ref. [49]. Note that, for a simpler comparison between simulation and theory, the Coulomb logarithm was set to 5 and e–e and i–i collisions were also applied to ensure Maxwellian distributions of each species.

This example of the effect of e–i (*inter*-) collisions has the following counterpart for e–e (*intra*-) collisions. We set an hydrogen plasma of the same density with an anisotropic electron temperature: $T_\parallel = 150$ eV and $T_\perp = 50$ eV. The right panel of Fig. 12 shows the evolution of both temperatures due to the e–e collisions, again well matched by the theoretical solution in Ref. [49].

Another important consequence of Coulomb collisions is the slowing down of high-energy electrons passing through an ionized plasma (due to e–e collisions). We simulated this situation for various electron energies traversing a fully-ionized hydrogen plasma, and present the resulting stopping power in Fig. 13. It is in good agreement with theoretical calculations from Ref. [50].

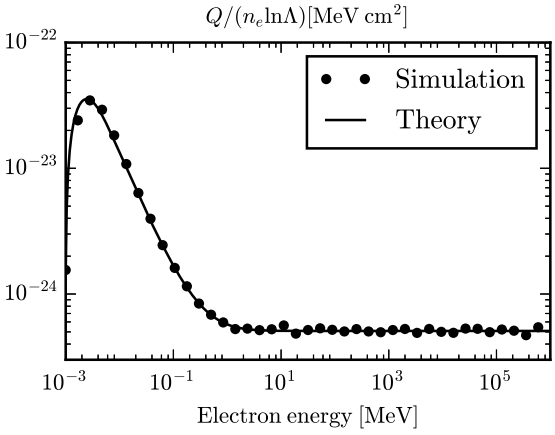

Fig. 13. Stopping power Q of a fully-ionized hydrogen plasma of density $n_e = 10^{22}$ cm^{-3} and temperature 1 keV, divided by n_e and by the Coulomb logarithm $\ln\Lambda$, as a function of the incident electron energy.

5.5. Collisional ionization

The collision module described in Section 5.4 hosts an electron–ion impact-ionization model that makes use of the particle pairing to compute the ionization probability of each pair. The scheme is identical to that of Ref. [46] with the exception of a few improvements detailed in the following.

The overall approach consists in calculating quantities averaged over all orbitals of a given ion with atomic number Z and charge $Z^\star$, instead of dealing with each orbital individually. This greatly reduces the amount of random numbers to generate. In this regard, this scheme is partially deterministic.

At the beginning of the simulation, the cross-section formulae from Ref. [51] are averaged over all the ions orbitals for each value of $Z^\star$ and for a given set of incident electron energies. In addition to these tabulated average cross-sections $\overline{\sigma}$, the average energy $\overline{e}$ lost by the incident electron, and the average energy $\overline{w}$ transferred to the secondary electron, are tabulated at the same time. For each particle pair that collides during the simulation, these tables are interpolated, providing an ionization probability. When an ionization occurs, the incident electron energy is reduced by $\overline{e}$, a new electron with energy $\overline{w}$ is created, and $Z^\star$ is incremented.

In Ref. [46], the ionization probabilities and the energy transfers assume that the ion frame is the laboratory frame. To overcome this limitation, Smilei introduces the following changes. The electron Lorentz factor in the ion frame is calculated using the relativistic

transformation $\gamma_e^\star = \gamma_e\gamma_i - \mathbf{p}_e \cdot \mathbf{p}_i/(m_e m_i)$ and the probability for ionization can be expressed as:

$$P = 1 - \exp\left(-v_e\overline{\sigma}n\Delta t\right) = 1 - \exp\left(-V^\star\overline{\sigma}^\star n\Delta t\right) \tag{39}$$

where v_e is the electron velocity in the laboratory frame, n is the particle density in the laboratory frame, $\overline{\sigma}^\star$ is the cross-section in the ion frame, and $V^\star = \sqrt{\gamma_e^{\star\,2} - 1}/(\gamma_e\gamma_i)$. If ionization occurs, the loss of energy $\overline{e}$ of the incident electron translates into a change in momentum $p_e^{\star\prime} = \alpha_e p_e^\star$ in the ion frame, with $\alpha_e = \sqrt{(\gamma_e^\star - \overline{e})^2 - 1}/\sqrt{\gamma_e^{\star 2} - 1}$. To calculate this energy loss in the laboratory frame, we apply the relativistic transformation:

$$\mathbf{p}_e' = \alpha_e\mathbf{p}_e + ((1-\alpha_e)\gamma_e^\star - \overline{e})\frac{m_e}{m_i}\mathbf{p}_i. \tag{40}$$

A similar operation is done for calculating the momentum of the new electron in the laboratory frame: it is created with energy $\overline{w}$ and its momentum is $p_w^\star = \alpha_w p_e^\star$ in the ion frame, with $\alpha_w = \sqrt{(\overline{w}+1)^2 - 1}/\sqrt{\gamma_e^{\star 2} - 1}$. In the laboratory frame, it becomes:

$$\mathbf{p}_w = \alpha_w\mathbf{p}_e + (\overline{w} + 1 - \alpha_w\gamma_e^\star)\frac{m_e}{m_i}\mathbf{p}_i. \tag{41}$$

Finally, Eqs. (39)–(41) ensure that all quantities are correctly expressed in the laboratory frame.

To test this first improvement, let us consider the inelastic stopping power caused by e–i collisions when a test electron beam is injected in a cold, non-ionized Al plasma of ion density 10^{21} cm^{-3}. Electrons of various initial velocities are slowed down by the ionizing collisions and their energy loss is recorded as a function of time. The left panel of Fig. 14 provides the corresponding stopping power, compared to the theory of Rohrlich and Carlson [52]. Knowing that this theory is valid only well above the average ionization energy (here $\sim$ 200 eV), the agreement is satisfactory. At energies above 10^7 keV (the ion rest mass), the center-of-mass frame is not that of the ions, thus the agreement with the theory confirms the validity of our correction.

Another modification has been added to the theory of Ref. [46] in order to account for multiple ionization in a single time-step. This approach closely follows that presented for field ionization in Section 5.3, the only difference being in the computation of the ionization rates (described above). It was tested and validated for a wide range of materials and incident electron energies. An example is given in the right panel of Fig. 14, where a fast electron beam ionizes a zinc plasma. The ionized electrons' density is plotted against time, for two vastly different time-steps. With these parameters, the multiple-ionization scheme matches better the well-resolved case than the single-ionization scheme. We found that it takes a reduction of an order of magnitude in the time-step for the single-ionization approach to work as well as the multiple-ionization scheme. It therefore brings a significant accuracy improvement.

6. User interface

6.1. Python input file

End-users only need to know how to write an input file, or *namelist*. Although the core of Smilei is written in C++, the namelist is written in the *python* language. This has many advantages over the typical text-only inputs. Indeed, *python* can process complex operations that may be necessary to initialize the simulation. It can generate arbitrary numbers of simulation elements at runtime, without the help of an external script (which would have to be pre-processed). It supports thousands of additional packages, often helpful for specific physics calculations. It is widely used and becoming a reference for all sorts of applications. Very importantly, *python* functions can be passed as arguments to Smilei. For instance, a density profile can be directly defined as a function of the coordinates.

When Smilei is run, it starts a *python* interpreter that parses the namelist line-by-line, and executes all the *python* commands. Throughout the initialization of the simulation elements (particles, fields, diagnostics, etc.) the interpreter stays active. Smilei gathers required data from it, processes all required initialization steps, and finally closes the interpreter. Note that, if a *python* function needs to be evaluated throughout the simulation, the interpreter is kept active at all times. This happens, for instance, when defining a custom temporal profile for a laser envelope.

6.2. Diagnostics

Data collection and analysis are performed by *diagnostics*. They are not post-processing modules, but are part of the main code and executed at runtime. All of these diagnostics have the capability of being performed only at user-defined times during the simulation.

Scalar diagnostic – The simplest diagnostic is called *scalars*: it processes a large set of field and particle data, and combines the results from all processors before writing out scalar quantities in a dedicated file. Among these quantities, one can find the overall energy balance (with contributions from the different fields, particles, and losses at the boundaries), averaged particle quantities (charge, energy, number of particles), and global field information (minima, maxima and Poynting flux through boundaries).

Fields diagnostic – The diagnostic *fields* provides a direct copy of all the arrays in the code, after concatenating them from all the processors. Note that, in addition of the **E** and **B** fields, the particle densities and currents are also written as they are projected on arrays at each time-step. Moreover, these data may be temporally averaged over a number of time-steps requested by the user.

Probe diagnostics – The drawback of the diagnostic *fields* is that the whole arrays are written out. To reduce the file space footprint, the *probes* have been implemented: one *probe* corresponds to a series of points at which locations the fields are interpolated and written in a dedicated file. This series of points can be either regularly arranged in a line, in a rectangle (for a two-dimensional simulation), or in a parallelepiped (for a three-dimensional simulation). The spatial separations between consecutive points is defined by the user. Note that several *probes* can be added to a single simulation.

Trajectory diagnostics – Histories of individual quasi-particles are stored by the *tracking* diagnostic. Each species of particles may be *tracked* independently, with custom output frequencies. In order to follow individual particles, each tracked particle is assigned a unique number which is transported throughout the simulation.

Particle distribution diagnostics – Tracking the position of all quasi-particles with a high frequency would be time- and memory-consuming. To obtain digested data with flexible capabilities, the *particle diagnostic* has been implemented. One diagnostic is defined by an arbitrary number of *axes*, which overall define a grid: all the quasi-particles in the selected species deposit their weight in the grid cell they belong to (the cell size is unrelated to the PIC grid). These *axes* are not necessarily spatial (x, y or z), but can also be one of p_x, p_y, p_z, p, γ, v_x, v_y, v_z, v or the particle charge q. A large number of combinations can thus be designed. For instance, using one axis [x] will provide the density distribution *vs.* x; using two axes [x, y] will provide the two-dimensional density distribution *vs.* x and y; using one axis [p_x] will provide the x-momentum distribution; using two axes [x, p_x] provides the phase-space along x; using three axes [x, y, γ] provides density maps at different energies; using one axis [q] provides the charge distribution. Further versatility is possible by choosing which piece of data is deposited in each cell instead of the quasi-particle weight w. For instance, depositing the product $w\,q\,v_x$ results in the j_x current density and depositing $w\,v_x\,p_x$ results in a component of the pressure tensor. A final feature of these *particle diagnostics* is the capability for temporal averaging over an arbitrary number of time-steps.

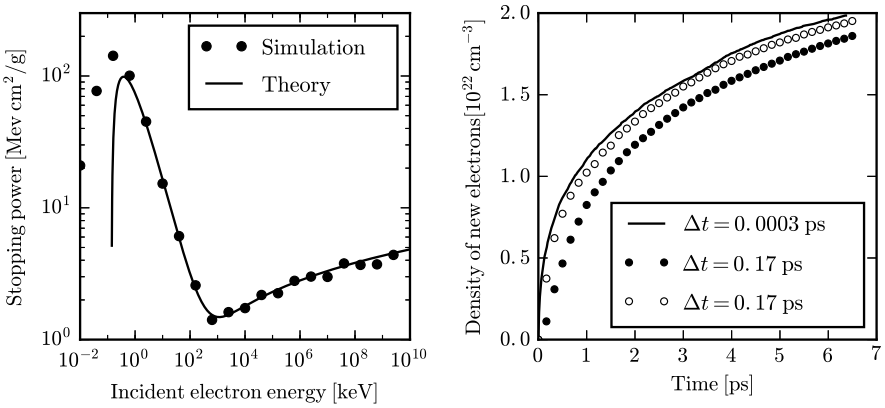

Fig. 14. Left: inelastic stopping power of a cold aluminum plasma of density 10^{21} cm^{-3} as a function of the incident electron energy. Right: evolution of the secondary electron density caused by a beam of 1 MeV electrons traversing a cold zinc gas (both electrons and target have a density of 10^{21} cm^{-3}), for various simulation time-steps. The open circles correspond to the multiple-ionization scheme.

7. Physics highlights

In this section, we present a few examples of simulations highlighting physics studies relying on Smilei. The first two are related to laser–plasma interaction studies, the latter two to astrophysics.

7.1. High-harmonic generation and electron acceleration from intense femtosecond laser interaction with dense plasmas

The interaction between an ultra-intense ($I > 10^{18}$ W/cm^2) femtosecond laser pulse with a solid target generates a dense "plasma mirror" at its surface that reflects the laser in a strongly non-linear manner. The temporal distortion of the reflected wave creates a train of ultra-short attosecond pulses, associated, in the frequency domain, to a comb of high-order harmonics. This scheme is considered as one of the best candidates for attosecond light sources [9]. Recent experiments have shown that it also produces high-energy (relativistic) ultra-short and very-high-charge (nC) electron bunches [53], of interest for electron injectors.

In what follows, we present a 2-dimensional Smilei simulation of laser–solid interaction, in conditions relevant to experiments at the UHI 100 laser facility.[10] The laser pulse with wavelength $\lambda_0 = 0.8\,\mu$m has a peak intensity $I \simeq 2\times10^{19}$ W/cm^2 (normalized vector potential $a_0 = 3$) when focused to a $4\lambda_0$ waist, at 45°-incidence with p-polarization, onto an overdense plasma slab. This overdense plasma mimics the solid target considered fully ionized with a constant electron density $n_0 = 200\,n_c$ ($n_c \simeq 1.7\times10^{21}$ cm^{-3} being the critical density), $5\lambda_0$-thick, with an exponential pre-plasma of gradient length $0.1\,\lambda_0$ down to a cut-off density $n_{c-off} = 0.05\,n_c$. The full box size is $80\,\lambda_0 \times 60\lambda_0$ and the simulation time $150\,\lambda_0/c$. The cell size is $\Delta x = \Delta y = \lambda_0/256$ (for a total of 25,600 × 26,880 cells) and the timestep is $c\Delta t = \lambda_0/384 \simeq 0.95\,\Delta t_{CFL}$. Eight to 49 quasi-particles are set in each cell, for a total of ~ 1.4 billions of quasi-particles in the entire box. They are frozen (not moved) until $t = 50\,\lambda_0/c$, i.e. until the laser pulse reaches the target.

Fig. 15 presents the simulation set-up and a summary of the results obtained. The top panel represents half of the simulation box in the y-direction, and the laser field is reported at three different times. The reflected laser pulse (at time t_2) shows a different spectral content than the incident pulse (at time t_0). The plasma electron density is shown in black. A close-up view of the interaction region is given in the bottom panel, illustrating the electron bunches being pulled out from the plasma surface.

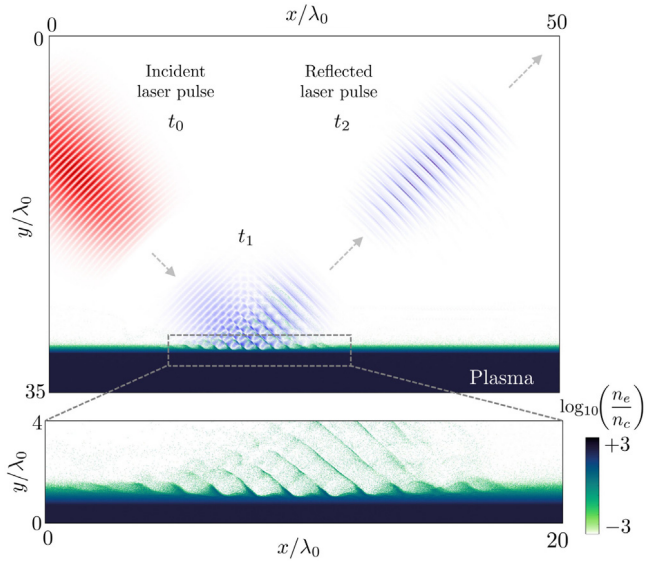

Fig. 15. Setup and results of a laser–solid interaction simulation. Top: laser magnetic field B_z snapshots at three different times: t_0 before interaction with the plasma, t_1 during interaction and t_2 after reflection by the plasma mirror. The dark-scale region represents the plasma electron density at time t_1. Bottom: close-up of the interaction region showing the plasma electron density at t_1, during interaction.

Fourier analysis of the reflected laser magnetic field B_z in space and time provides the angular distribution of the frequency spectrum of the reflected light. High harmonics (up to order 16) are observed as seen in the top panel of Fig. 16. In addition, electron acceleration was observed, as highlighted in the bottom panel of Fig. 16, showing the trajectories of electrons ejected from the target. The most energetic electrons (with energies up to 10 MeV) are found to propagate in the direction of the reflected light pulse. The angular histogram also shows that the momenta of the escaping energetic electrons are mostly directed along two directions which are close to the reflected laser direction. This is consistent with vacuum electron acceleration suggested in Ref. [53].

This simulation was run on the CINES/Occigen (Bullx) machine using 256 MPI × 14 OpenMP threads for about 10,700 CPU-hours. Considering only the simulation time during which particles are not frozen, the characteristic time to push a particle (complete time to run one full PIC loop divided by the product of the number of particles by the number of timesteps) is of the order of 0.717 μs, 25% of which were devoted to diagnostics.

10 http://iramis.cea.fr/slic/UHI100.php.

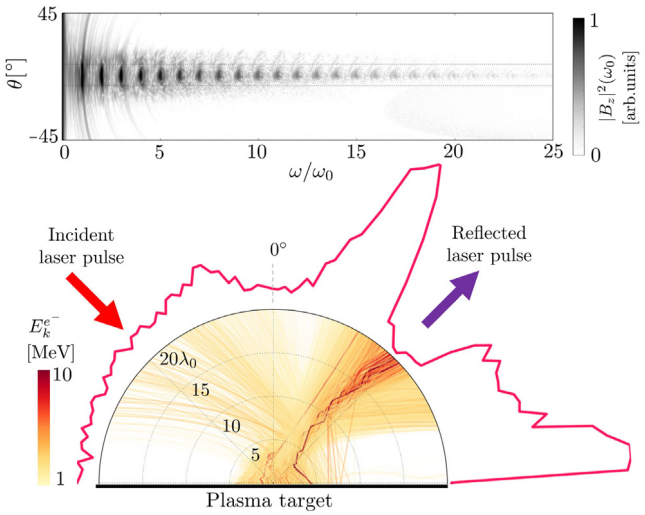

Fig. 16. Top: angular distribution of high-harmonics generated in short-pulse laser–plasma interaction. Bottom: typical trajectories of electrons ejected from the target and accelerated in the reflected laser field. The color scale denotes the electron kinetic energy. The pink curve is a histogram of the electron momentum angle when they reach a "detector" located at a distance of 20 λ_0 from the focal spot. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7.2. Short laser pulse amplification by stimulated Brillouin scattering

The generation of short high-intensity laser pulses is limited by the damage threshold of solid optics materials [54,55], but such limitations could be overcome using a plasma as an amplifying medium. This can be achieved by coupling, in a plasma, a long energetic "pump" pulse of moderate intensity and a short counter-propagating "seed" pulse of initially low intensity. Energy transfer from the pump to the seed thanks to the excitation of a plasma wave can then be obtained [56,57]. In what follows, we focus on stimulated Brillouin scattering (SBS) amplification, where the excited waves are ion-acoustic waves.

In the case of a pump with intensity $I_p \gtrsim 10^{15}\mathrm{W/cm^2}$ (with the laser wavelength $\lambda_0 = 1\ \mu\mathrm{m}$), SBS amplification operates in its "strong-coupling" regime [58–61]. This scheme is particularly robust with respect to plasma inhomogeneities and does not require any frequency shift of the seed pulse.

Multi-dimensional kinetic simulations are required to describe the competing processes (spontaneous Raman scattering, filamentation, saturation), to study the non-linearities intervening in the amplification mechanism, and to optimize the resulting phase front (for later focusing [62,63]), but they appear very challenging as inherently multi-scale. We present here two 2-dimensional Smilei simulation of short-pulse SBS amplification in conditions close to actual experiments [64,65]. The simulation box size is $1024\ \mu\mathrm{m} \times 512\ \mu\mathrm{m}$ and the grid cells are 33nm in both directions, resulting in 30,720 × 15,360 cells. The simulation lasts 10 ps with a timestep of 7.3×10^{-2} fs (over 135,000 timesteps in total). Respectively 25 and 16 billions of quasi-particles have been set in each simulation.

The first simulation corresponds to typical present-day experiments. The pump has a $\cos^2$-temporal profile with duration 4.2 ps FWHM and maximum intensity $I_p = 10^{15}\ \mathrm{W/cm^2}$ and propagates along the x-direction toward $x > 0$. The counter-propagating seed has a $\cos^2$-temporal profile with duration 0.5 ps FWHM and initial intensity $I_s = 10^{15}\mathrm{W/cm^2}$. Both the seed and pump lasers have a transverse Gaussian profile with 130 μm FWHM (in terms of intensity). The plasma has a Gaussian density profile over all the simulation box, with a maximum (central) electron density $n = 0.1\ n_c$, where n_c the critical density for both the laser pump and seed ($n_c \simeq 1.1 \times 10^{21}\ \mathrm{cm^{-3}}$ at $\lambda_0 = 1\ \mu\mathrm{m}$).

Typical simulation results are presented in Fig. 17 showing pump and seed intensities at three different amplification stages. At $t = 5.8$ ps (panel a), the seed starts interacting with the pump. At $t = 7.6$ ps (panel b), the seed reaches the middle of the simulation box. At that time, the seed is still in the linear amplification regime, and the pump is not depleted yet. At $t = 9.7$ ps (panel c), the seed has traveled through the entire simulation box and the pump is depleted. The final intensity of the seed is $I_s^{\mathrm{out}} \simeq 4.6 \times 10^{15}\ \mathrm{W/cm^2}$, i.e. nearly 5× its initial intensity. The spot size and phase front are also well conserved, suggesting that such a beam could be further focused using plasma mirrors to reach even larger intensities.

The second simulation deals with an innovative plasma–laser configuration to further optimize SBS amplification. The seed pulse is now interacting with two pump lasers, both with a $\cos^2$-temporal shape with duration 4.2 ps FWHM, and top intensity $I_p = 10^{15}\ \mathrm{W/cm^2}$ that are propagating with an angle of $\pm 6°$ degrees with respect to the x-axis. Taking two pump pulses is an experimentally convenient configuration that has the advantage to increase the pump intensity in the 3-pulse-interaction region while keeping a relatively low pump intensity during propagation in the non-overlapping region (thus reducing spurious Raman losses). Moreover, this laser–plasma configuration allows to separate the Raman backscattering of the pump from the amplified signal (as will be shown in what follows). The transverse size of the pump pulses is, for this simulation, reduced to 30 μm FWHM and the plasma has a constant density profile with electron density $n = 0.05\ n_c$.

The typical interaction set-up and simulation results are shown in Fig. 18(a). The vertical white-dashed lines delimit the constant plasma, and the amplified seed exiting the simulation box at $t = 10$ ps reaches a final intensity $I_s^{\mathrm{out}} \simeq 3 \times 10^{15}$ (3× the initial intensity). Of outmost interest is the spatio-temporal (ω, k) spectrum of the light recorded on the left-boundary of the simulation box presented in Fig. 18(b). As expected, this set-up allows the Raman signal (at $\omega \simeq 0.76\,\omega_0$ with $\omega = 2\pi c/\lambda_0$ the laser angular frequency) originating from the backscattering of the pump to propagate mostly in the opposite pump directions (the signal is mainly at $k \simeq \pm 0.11\,k_0$, with $k_0 = 2\pi/\lambda_0$), thus angularly separating its contribution from the seed. Furthermore, this

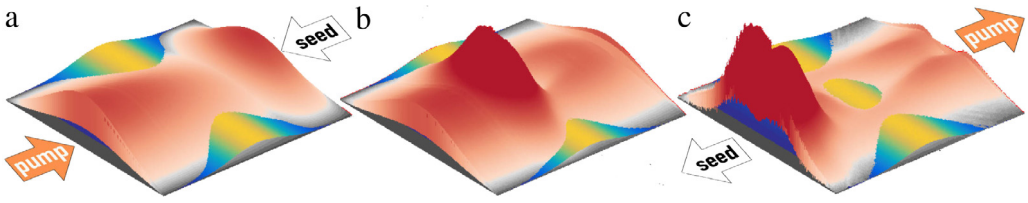

Fig. 17. Evolution of the pump and seed intensities in the case of 2 pulse head-on collision at: (a) $t = 5.8$ ps, (b) $t = 7.6$ ps and (c) $t = 9.6$ ps. The blue–yellow maps correspond to the plasma density while the white–red maps correspond to the lasers intensity. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

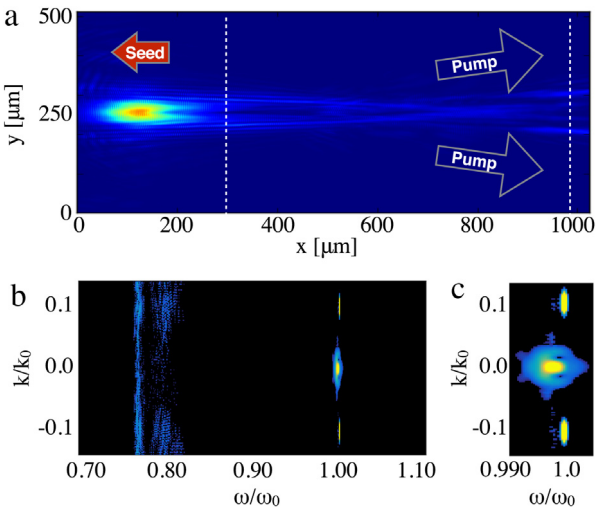

Fig. 18. (a) Pump and seed intensity at the end of amplification at $t = 10$ ps. The final intensity of the seed is $I_s^{\text{out}} \simeq 3 \times 10^{15}$ W/cm^2 (3× its initial intensity). (b) Spectrum (in terms of wave number k/k_0 and frequency ω/ω_0, where k_0 and ω_0 are the nominal wavenumber and frequency of the pump lasers) of the electric field recorded on the entire length of the left side of the simulation box. (c) Zoom of the spectrum for $\omega/\omega_0 = [0.98, 1.02]$.

spectrum confirms the dominant role of SBS amplification in the seed amplification. Indeed, both broadening and red-shift (toward small temporal frequencies $\omega < \omega_0$) shown in Fig. 18(b) [and insert (c)] are signatures of SBS amplification. The signal at $\omega \simeq \omega_0$ and $k \simeq \pm 0.11\, k_0$ correspond to the (forward-propagating) pump lasers. Notice that, at the end of the amplification, the transverse focal spot size of the seed at FWHM in intensity is of the order of 28 μm, i.e. of the same order than the initial one.

Both simulations have been performed on the IDRIS/Turing (BlueGene/Q) super-computer using 1.8 million CPU-hours on 32,768 MPI processes, and 4 OpenMP threads per core to take best advantage of the architecture. The average time to push a particle was ∼ 1.9 μs, 5% of which were devoted to diagnostics. The typical memory footprint for these simulations was of the order of 1 Tb, and each simulation generated over 1 Tb of output data. On the CINES/Occigen (Bullx) machine, we obtained an average time of 0.43 μs to push one particle (without diagnostics).

7.3. Magnetic reconnection at the Earth magnetopause

Magnetic reconnection at the Earth magnetopause regulates the transport of matter, momentum and energy from the solar wind to the internal magnetosphere. Because of their different origins, the properties of the plasma and magnetic field on both side of the magnetopause are quite different. The solar wind plasma temperature is typically one tenth that of the magnetospheric plasma, but its density is about ten times larger. The magnetic field is typically 2–3 times larger on the magnetospheric side than on the solar wind side. This asymmetry makes the reconnection dynamics vastly more complex than in symmetric environments, and has only been studied for a decade via numerical simulations and spacecraft observations [66,67]. Among all possible asymmetries, those in the particle density and magnetic field amplitude have by far the most important impact on the reconnection rate.

Following times of strong magnetospheric activity, very dense and cold plasma from the plasmasphere can be transported all the way up to the Earth magnetopause, forming an elongated tongue of dense material. As it impacts the magnetopause, it drastically changes the asymmetry described above. If it reaches the magnetopause at a location where magnetic reconnection is already ongoing with a typical asymmetry, the filling of the reconnection site with cold plasma, which density can even exceed the solar wind density, should affect importantly the reconnection dynamics, first by lowering the reconnection rate.

Studying the impact of a plasmaspheric plume on magnetopause reconnection via kinetic numerical simulation is difficult. Indeed, the simulation first needs to reach a quasi-steady state reconnection with a typical magnetopause asymmetry, see the arrival of the plume and then last longer for a quasi-steady state plume reconnection regime to settle. Due to the large particle density of plumes, the transition and last phases have substantially longer time scales than the early phase, which makes the simulation heavy. The domain must be long enough in the downstream direction for the plasma, expelled during the early and transition phases, to be evacuated from the reconnection region. Otherwise, upstream plasma would not inflow, thereby stopping reconnection.

We designed a simulation so that typical magnetopause reconnection can proceed and form a reconnection exhaust of about $100\, c/\omega_{pi}$, where ω_{pi} is the ion plasma frequency corresponding to the reference (solar wind) density n_0, long before the plume reaches the reconnection site. Using the Cassak–Shay estimate of the inflow velocity [68], we need to position the plume on the magnetospheric side at about $20\, c/\omega_{pi}$ from the initial magnetopause position. Three ion populations are present. The solar wind and magnetospheric populations have densities equal to n_0 and $n_0/10$, respectively, on their side of the current sheet, and fall to zero on the other side. The plume population increases from 0 to $2\, n_0$ at $20\, c/\omega_{pi}$ from the initial current sheet on the magnetospheric side. The magnetic field amplitude goes from $2\, B_0$ in the magnetosphere to $B_0 = m_e\omega_{pe}/e$ in the solar wind and is totally in the simulation plane. The temperature is initially isotropic and its profile is calculated to balance the total pressure.

The domain size is $1280\, c/\omega_{pi} \times 256\, c/\omega_{pi}$ for 25,600 × 10,240 cells, in the x (downstream) and y (upstream) directions. The total simulation time is $800\, \Omega_{ci}^{-1}$ with a time step $0.00084\, \Omega_{ci}^{-1}$, where $\Omega_{ci} = eB_0/m_i$ is the ion gyrofrequency. We used a reduced ion to electron mass ratio $m_i/m_e = 25$, and a ratio $c/V_A = 50$ of the speed of light by the Alfvén velocity. There are initially 8.6 billion quasi-protons for the three populations, and 13 billion electrons.

Fig. 19 presents some of the simulation results: the electron density at three different times. In the top panel, reconnection is in steady state between the solar wind plasma of density $\simeq n_0$ and the magnetosphere plasma of density $\simeq 0.1\, n_0$. At this time, the exhaust is filled with mixed solar wind/hot magnetospheric plasma as the plume (of density $\simeq 2\, n_0$) is still located at $\simeq 10\, c/\omega_{pi}$ from the magnetospheric separatrix. The reconnection rate during this period has a typical value around $0.1\, \Omega_{ci}^{-1}$, with important fluctuations caused by plasmoid formation. The plume, originally at $20\, c/\omega_{pi}$ from the magnetopause, is slowly advected toward the magnetosphere separatrix and finally touches the reconnection site at about $t = 300\, \Omega_{ci}^{-1}$. The second panel at $t = 370\, \Omega_{ci}^{-1}$ shows the plume starting to fill the exhaust after reaching the reconnection site and mixing with solar wind plasma. At this time, the reconnection rate collapses to about half its previous value. The transition phase lasts for about $100\, \Omega_{ci}^{-1}$ before a plume reconnection regime reaches a quasi-steady state. The third panel shows the electron density at the end of the simulation, where the exhaust is filled with plume and solar wind plasma.

This large-scale simulation has run for a total of 14 million CPU-hours on 16,384 cores of the CINES/Occigen (Bullx) supercomputer within a GENCI-CINES special call. Overall, the characteristic (full) push-time for a single particle was of the order of 1.6 μs, 31% of which were devoted to diagnostics. Note that no dynamic load balancing was used for this simulation.

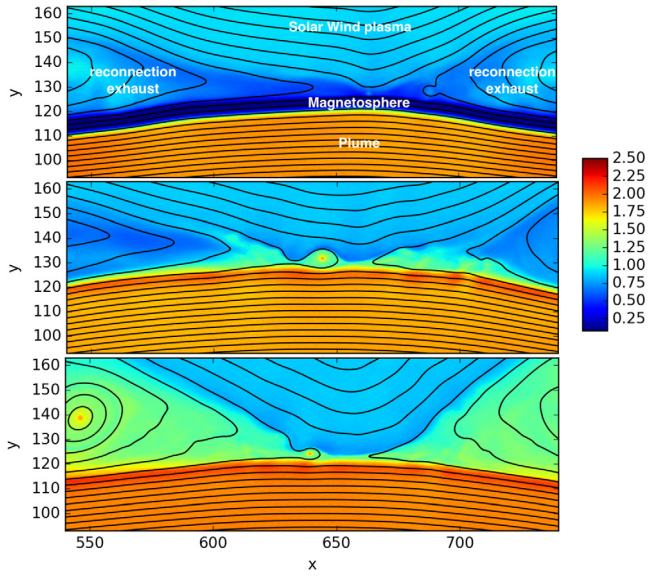

Fig. 19. Magnetopause reconnection simulation results: electron density color coded at different times ($t = 220$, 370 and $800\Omega_{ci}^{-1}$ from top to bottom) in a region zoomed around the reconnection site. Solid black lines are in-plane magnetic field lines.

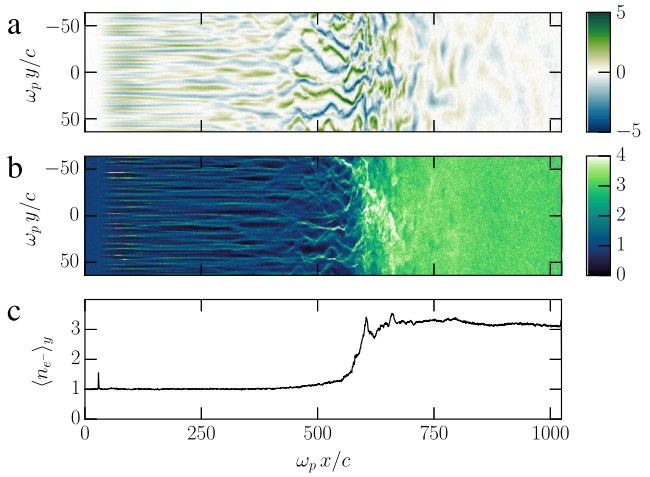

Fig. 20. Snapshot at $t = 1000\,\omega_p^{-1}$. (a) Weibel generated magnetic field B_z in units of $B_0 = m_e\,\omega_p/e$. (b) Electron density in units of n_0. (c) Electron density (in units of n_0) averaged along the y-direction.

7.4. Collisionless shock in pair plasmas

Relativistic collisionless shocks play a fundamental role in various astrophysical scenarios (active galactic nuclei, micro-quasars, pulsar wind nebulae and gamma-ray bursts) where they cause high-energy radiation and particle acceleration related to the cosmic-ray spectrum [69]. The long-standing problem of describing collisionless shock formation has gained renewed interest as PIC simulations provide insight into the micro-physics of these non-linear structures [70–72].

In the absence of particle collisions, the shock is mediated by collective plasma processes, produced by electromagnetic plasma instabilities, taking place at the shock front. In particular, we study the Weibel (or current filamentation) instability [73–75] that is observed in most of the astrophysical relativistic outflows interacting with the interstellar medium. It can be excited by counter-streaming unmagnetized relativistic flows, and it has been shown to dominate the instability spectrum for a wide range of parameters [74]. It converts part of the kinetic energy of the counter-propagating flows into small-scale magnetic fields, which are then amplified up to sub-equipartition levels of the total energy. The resulting strong magnetic turbulence can isotropize the incoming flow (in the center-of-mass frame), hence stopping it and leading to compression of the downstream (shocked plasma) and shock formation.

The density compression ratio between the upstream relativistic flow (with density n_0) and the downstream (with density n_d) plasma can be derived from macroscopic conservation laws giving the Rankine–Hugoniot (RH) jump conditions [76]. The shock is considered formed when the density jump becomes $n_d/n_0 = 1 + (\gamma_0 + 1)/[\gamma_0(\Gamma_{ad} - 1)]$. Considering an ultra-relativistic incoming flow $\gamma_0 \gg 1$ and adiabatic index $\Gamma_{ad} = 3/2$ for a 2-dimensional downstream plasma at ultra-relativistic temperature, we expect a compression factor $n_d/n_0 = 3$. Another clear signature of the shock formation is the isotropization of the downstream plasma. This physical picture has been confirmed by various PIC simulations using counter-penetrating relativistic flows [70–72].

In what follows, we present a 2-dimensional PIC simulation of a Weibel-mediated collisionless shock driven in an initially unmagnetized electron–positron plasma. The simulation relies on the "piston" method that consists in initializing the simulation with a single cold electron–positron plasma drifting in the $+x$-direction at a relativistic velocity $v_0 \simeq 0.995\,c$ ($\gamma_0 = 10$). A reflecting (for both fields and particles) boundary condition is applied at the right border of the simulation box, hence creating a counter-penetrating (reflected) flow, the reflected beam mimicking a flow with velocity $-v_0$.

The simulation box size is $2048\,\delta_e \times 128\,\delta_e$, $\delta_e = c/\omega_p$ being the (non-relativistic) electron skin-depth of the initial flow. The spatial resolution is set to $\Delta x = \Delta y = \delta_e/16$, the timestep to $c\Delta t = \Delta x/2$ and 16 particles-per-cell were used for each species leading to a total of $\simeq 2.15 \times 10^9$ quasi-particles. Temporal Friedman filtering (with $\theta = 0.1$) and binomial current filtering (using 3 passes) have been applied in order to avoid spurious effects (e.g. upstream heating) due to the grid-Cerenkov numerical instability (see Section 5.1).

Fig. 20 presents the characteristic simulation results at time $t = 1000\,\omega_p^{-1}$. The overlapping region of incoming and reflected flows is Weibel-unstable which results in the creation, before the shock ($50\,\delta_e < x < 400\,\delta_e$), of filamentary structures in both the magnetic field (panel a) and the total plasma density (panel b). The magnetic field at the shock front ($400\,\delta_e < x < 600\,\delta_e$) becomes turbulent and it is strong enough to stop the incoming particles leading to a pile-up of the plasma density up to $n_d \simeq 3.2\,n_0$ (panel c), as predicted by the RH conditions. The simulation also indicates that the shock propagates toward the left with a velocity $v_{sh} \simeq (0.46 \pm 0.01)\,c$. The RH conditions predict a shock velocity $v_{sh} = c\,(\Gamma_{ad} - 1)(\gamma_0 - 1)/(\gamma_0 v_0) \simeq 0.452\,c$, in excellent agreement with the value observed in the simulation. Isotropization and thermalization of the downstream distribution function was also observed (not shown), with a typical temperature close to that predicted from the RH conditions $T_d = \frac{1}{2}(\gamma_0 - 1)\,m_e c^2 \simeq 4.5\,m_e c^2$.

Finally, the simulation also demonstrates the build-up of a supra-thermal tail in the downstream particle energy distribution, as shown in Fig. 21. At an early time after shock formation, $t \simeq 500\,\omega_p^{-1}$, the particle distribution in the downstream (shocked plasma, here taken at $800\,c/\omega_p < x < 900\,c/\omega_p$) is found to have relaxed to an isotropic, quasi-thermal, distribution with a temperature initially only slightly larger than that predicted from RH conditions $T_d \simeq 4.5\,m_e c^2$ (dashed line). At later times ($t \to 2000\,\omega_p^{-1}$), a supra-thermal tail of energetic particles has built up, even more pronounced for particles located in the downstream plasma closer to the shock front (here taken in a region $500\,c/\omega_p <$

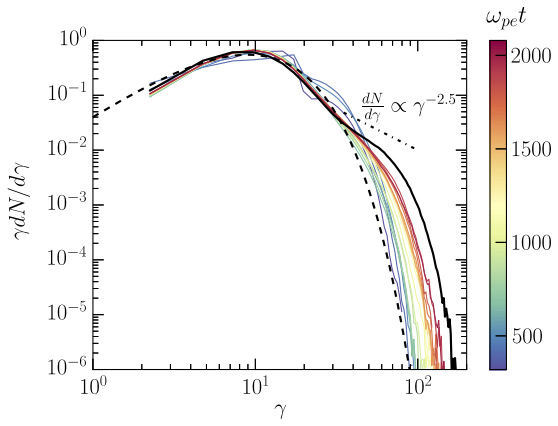

Fig. 21. Electron energy distribution in the downstream region ($800\,c/\omega_p < x < 900\,c/\omega_p$) for different times $t > 450\,\omega_p^{-1}$. At time $t \simeq 500\,\omega_p^{-1}$, the shock is already formed and the energy distribution closely follows the thermal (2-dimensional) Maxwell–Jüttner distribution with temperature $T_d = 4.5\,m_e c^2$ expected from RH conditions (black dashed line). At later times, a supra-thermal tail appears, even more visible for particles located closer to the shock front ($500\,c/\omega_p < x < 600\,c/\omega_p$, solid black line). The dot-dashed guide line indicates the power law $dN/d\gamma \propto \gamma^{-2.5}$.

$x < 600\,c/\omega_p$, solid black line). This tail is characteristic of first order Fermi acceleration at the shock front, and appears following a $dN/d\gamma \propto \gamma^{-s}$ scaling with $s = 2.5$, a power index consistent with previous PIC simulations [70,71] and theoretical investigations [77].

This simulation run on the TGCC/Curie machine using 512 MPI $\times$ 8 OpenMP threads over a total of 26,100 CPU-hours for 65,536 timesteps. For this simulation, the characteristic (total) push time for a single quasi-particle was of 0.66 μs, 20% of which were devoted to diagnostics.

8. Conclusions and perspectives

To summarize the capabilities of the open-source PIC code Smilei, we emphasize on its object-oriented (C++) structure complemented by its user-friendly Python interface, making it a versatile, multi-purpose tool for plasma simulation.

Co-developed by both physicists and HPC experts, Smilei benefits from a state-of-the-art parallelization technique relying on a patch-based super-decomposition strategy. This approach allows for an improved cache use, and provides a straightforward implementation of dynamic load balancing. This strategy is shown to manage efficiently any load imbalance and to scale well over a very large number of computing elements (up to several hundred of thousands). The code was tested on various super-computers, architectures (Bullx and BlueGene/Q in particular) and processors (Intel Sandy Bridge, Broadwell and Haswell, and IBM Power A2).

Still a young project (development started in 2013), Smilei benefits from a wide range of additional modules, including a Monte-Carlo treatment of binary collisions as well as collisional and field ionization. Smilei is currently used by a growing community, as illustrated by the presented applications to both laser–plasma interaction and astrophysics.

The code is under active development and numerous enhancements are being incorporated collaboratively. Even though the simulations presented in this article are two-dimensional, the three-dimensional version of the code is now available for production (see, e.g., Refs. [78] and [79]).

On the physics side, Monte-Carlo modules are being developed to account for various quantum-electrodynamics effects, from high-energy photon to electron–positron pair production (see, e.g., Ref. [80] and references therein). The implementation of HPC-relevant spectral solvers for Maxwell's equations in Smilei, although challenging, is also under study. This is done through a collaboration with Lawrence Berkeley National Laboratory and the open-source PICSAR (PXR) project [81,82].

On the performance side, the incorporation of Single Instruction Multiple Data (SIMD), also known as *vectorization*, is in progress. A joint effort with *Intel*, aiming at optimizing the code for the Xeon Phi architectures, was initiated by the *Grand Equipement National de Calcul Intensif* (GENCI). To this day, two approaches are considered. The first one is based on a fine-grain particle sorting. The other relies on the integration of the vectorized PXR library in Smilei.

Finally the collaborative aspects are central to the Smilei project, and important efforts are undertaken: a complete documentation is under construction (a large portion of which is already available on Smilei's website), a set of automated benchmarks (*continuous integration*) is currently used and under constant development, the *openPMD* standard for I/O formatting [83] is also being implemented. Furthermore, user/training workshops are currently in preparation.

Acknowledgments

The authors are grateful to L. Gremillet, M. Lobet, R. Nuter and A. Sgattoni for fruitful discussions, and Ph. Savoini for feedback on the code. MG and GB thank F. Quéré and H. Vincenti for sharing physics insights. Financial support from the *Investissements d'Avenir* of the PALM LabEx (ANR-10-LABX-0039-PALM, Junior Chair SimPLE) and from the Plas@Par LabEx (ANR-11-IDEX-0004–02) are acknowledged. AG acknowledges financial support from the *Université Franco-Italienne* through the Vinci program (Grant No. C2-133). NA and JDa thank the ANR (project ANR-13-PDOC-0027) for funding their research. MG personally thanks the collaboration federated around the ANR MACH project (ANR-14-CE33–0019 MACH). This work was performed using HPC resources from GENCI-IDRIS *Grands Challenges 2015*, GENCI-IDRIS/TGCC (Grants 2016-x2016057678 and 2017-x2016057678), GENCI-CINES (Grant 2016-c2016067484) and GENCI-CINES (Special allocation n° t201604s020).

Appendix. Quasi-particles shape functions and interpolation/projection order

The quasi-particles shape function $S(\mathbf{x})$ has the following properties: (i) it is symmetric with respect to its argument $\mathbf{x}$, (ii) it is non-zero in a region centered around $\mathbf{x} = 0$ that extends over a distance $n\,\Delta x^\mu$ in $x^\mu = (x, y, z)$-direction, with Δx^μ the size of a cell in this direction, hence defining a so-called quasi-particle volume $V_p = \Pi_\mu n \Delta\mu$, where the integer n, henceforth referred to as the interpolation/projection order is discussed in what follows, and (iii) it is normalized so that $\int d\mathbf{x} S(\mathbf{x}) = 1$.

In what follows, we will consider different shape functions all of which can be written as a product over the D spatial dimensions of the simulation:

$$S(\mathbf{x}) = \prod_{\mu=1}^{D} s^{(n)}(x^\mu), \tag{A.1}$$

where n denotes the previously introduced interpolation/projection order. The one-dimensional shape functions $s^{(n)}(x)$ used in Smilei can be written in a recursive way. The interpolation/projection order $n = 0$ corresponds to a point-like quasi-particle and $s^{(0)}(x) = \delta(x)$, with $\delta(x)$ the Dirac distribution. The shape functions of higher order $n > 0$ are then obtained recursively:

$$\begin{aligned} s^{(n)}(x) &= \Delta x^{-1} P(x) \otimes s^{(n-1)}(x) \\ &\equiv \Delta x^{-1} \int_{-\infty}^{\infty} dx'\, P(x' - x)\, s^{(n-1)}(x'), \end{aligned} \tag{A.2}$$

with the crenel function $P(x) = 1$ if $|x| \leq \Delta x/2$ and $P(x) = 0$ otherwise. In what follows, we write explicitly the shape-functions $\hat{s}^{(n)} = \Delta x\, s^{(n)}$ for n up to 4 (i.e. up to fourth order):

$$\hat{s}^{(0)}(x) = \Delta x\, \delta(x), \tag{A.3}$$

$$\hat{s}^{(1)}(x) = \begin{cases} 1 \text{ if } |x| \leq \dfrac{1}{2}\,\Delta x, \\ 0 \text{ otherwise,} \end{cases} \tag{A.4}$$

$$\hat{s}^{(2)}(x) = \begin{cases} \left(1 - \left|\dfrac{x}{\Delta x}\right|\right) \text{ if } |x| \leq \Delta x, \\ 0 \qquad\qquad \text{otherwise,} \end{cases} \tag{A.5}$$

$$\hat{s}^{(3)}(x) = \begin{cases} \dfrac{3}{4}\left[1 - \dfrac{4}{3}\left(\dfrac{x}{\Delta x}\right)^2\right] \text{ if } |x| \leq \dfrac{1}{2}\,\Delta x, \\ \dfrac{9}{8}\left(1 - \dfrac{2}{3}\left|\dfrac{x}{\Delta x}\right|\right)^2 \text{ if } \dfrac{1}{2}\,\Delta x < |x| \leq \dfrac{3}{2}\,\Delta x, \\ 0 \qquad\qquad\qquad\qquad \text{otherwise,} \end{cases} \tag{A.6}$$

$$\hat{s}^{(4)}(x) = \begin{cases} \dfrac{2}{3}\left[1 - \dfrac{3}{2}\left(\dfrac{x}{\Delta x}\right)^2 + \dfrac{3}{4}\left|\dfrac{x}{\Delta x}\right|^3\right] \text{ if } |x| \leq \Delta x, \\ \dfrac{4}{3}\left(1 - \dfrac{1}{2}\left|\dfrac{x}{\Delta x}\right|\right)^3 \qquad\qquad \text{if } \Delta x < |x| \leq 2\,\Delta x, \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{A.7}$$

Field interpolation at the particle position

In Section 2.3, it is shown that the quasi-particles are subject to the electric and magnetic fields interpolated at their positions, these interpolated fields being given by Eqs. (9) and (10), respectively. For the simplest case of a one-dimensional grid, the field (either electric or magnetic) seen by the quasi-particle at position $\mathbf{x}_p = x_p\,\hat{\mathbf{x}}$ can thus be written in the form:

$$F_p = \int dx\, s^{(n)}(x - x_p)\, F(x), \tag{A.8}$$

where field $F(x)$ can be reconstructed from the grid as:

$$F(x) = \sum_i F_i\, P(x - x_i), \tag{A.9}$$

i denoting the grid point index and x_i the location of the ith grid point. Injecting Eq. (A.9) in Eq. (A.8), and using the recursive definition of the shape-function Eq. (A.2), one obtains a simple way to interpolate the field at the quasi-particle position:

$$F_p = \sum_i F_i\, \hat{s}^{(n+1)}(x_p - x_i). \tag{A.10}$$

The generalization to an arbitrary number of spatial dimension is straightforward.

Direct projection of the charge and current densities onto the grid

Direct projection of the charge and/or current densities onto a grid point x_i can be performed considering the projected quantity $[Q = (\rho, J)]$ as the amount of charge and/or current contained in the cell located around this grid point:

$$Q_i = \int dx\, Q_p\, s^{(n)}(x - x_p)\, P(x - x_i). \tag{A.11}$$

Using the recursive definition of the shape-function Eq. (A.2), one obtains:

$$Q_i = Q_p\, \hat{s}^{(n+1)}(x_i - x_p). \tag{A.12}$$

For the sake of completeness, it is worth noting that using the same shape-function for both interpolation and projection is mandatory to avoid unphysical self-force acting on the quasi-particles.

References

[1] F.H. Harlow, A Machine Calculation for Hydrodynamic Problems. Technical Report, Los Alamos Scientific Laboratory Report LAMS-1956, 1956.
[2] C.K. Birsall, A.B. Langdon, Plasma Physics Via Computer Simulation, McGraw-Hill, New York, 1985.
[3] T. Tajima, J.M. Dawson, Phys. Rev. Lett. 43 (1979) 267–270.
[4] A. Pukhov, J. Meyer-ter Vehn, Appl. Phys. B 74 (4) (2002) 355–361.
[5] S.P.D. Mangles, C.D. Murphy, Z. Najmudin, A.G.R. Thomas, J.L. Collier, A.E. Dangor, E.J. Divall, P.S. Foster, J.G. Gallacher, C.J. Hooker, D.A. Jaroszynski, A.J. Langley, W.B. Mori, P.A. Norreys, F.S. Tsung, R. Viskup, B.R. Walton, K. Krushelnick, Nature 431 (7008) (2004) 535–538.
[6] C.G.R. Geddes, Cs. Toth, J. van Tilborg, E. Esarey, C.B. Schroeder, D. Bruhwiler, C. Nieter, J. Cary, W.P. Leemans, Nature 431 (7008) (2004) 538–541.
[7] J. Faure, Y. Glinec, A. Pukhov, S. Kiselev, S. Gordienko, E. Lefebvre, J.P. Rousseau, F. Burgy, V. Malka, Nature 431 (7008) (2004) 541–544.
[8] A. Macchi, M. Borghesi, M. Passoni, Rev. Modern Phys. 85 (2013) 751–793.
[9] C. Thaury, F. Quéré, J. Phys. B: At. Mol. Opt. Phys. 43 (21) (2010) 213001.
[10] Hui Chen, Scott C. Wilks, James D. Bonlie, Edison P. Liang, Jason Myatt, Dwight F. Price, David D. Meyerhofer, Peter Beiersdorfer, Phys. Rev. Lett. 102 (2009) 105001.
[11] G. Sarri, K. Poder, J.M. Cole, W. Schumaker, A. Di Piazza, B. Reville, T. Dzelzainis, D. Doria, L.A. Gizzi, G. Grittani, S. Kar, C.H. Keitel, K. Krushelnick, S. Kuschel, S.P.D. Mangles, Z. Najmudin, N. Shukla, L.O. Silva, D. Symes, A.G.R. Thomas, M. Vargas, J. Vieira, M. Zepf, Nature Commun. 6 (2015) 6747.
[12] B. Cros, B.S. Paradkar, X. Davoine, A. Chancé, F.G. Desforges, S. Dobosz-Dufrénoy, N. Delerue, J. Ju, T.L. Audet, G. Maynard, M. Lobet, L. Gremillet, P. Mora, J. Schwindling, O. Delferrière, C. Bruni, C. Rimbault, T. Vinatier, A. Di Piazza, M. Grech, C. Riconda, J.R. Marquès, A. Beck, A. Specka, Ph. Martin, P. Monot, D. Normand, F. Mathieu, P. Audebert, F. Amiranoff, Nucl. Instrum. Methods Phys. Res. A 740 (2014) 27–33. Proceedings of the first European Advanced Accelerator Concepts Workshop 2013.
[13] A. Di Piazza, C. Müller, K.Z. Hatsagortsyan, C.H. Keitel, Rev. Modern Phys. 84 (2012) 1177–1228.
[14] Allen Taflove, Computation Electrodynamics: the Finite-Difference Time-Domain Method, third ed., Artech House, Norwood, 2005.
[15] Rachel Nuter, Mickael Grech, Pedro Gonzalez de Alaiza Martinez, Guy Bonnaud, Emmanuel d'Humières, Eur. Phys. J. D 68 (6) (2014) 177.
[16] Thomas P. Wright, G. Ronald Hadley, Phys. Rev. A 12 (1975) 686–697.
[17] Seiji Zenitani, Phys. Plasmas 22 (4) (2015) 042116.
[18] Brian P. Flannery, Saul Teukolsky, William H. Press, William T. Vetterling, Numerical Recipies, third ed., Cambridge University Press, 2007.
[19] J.P. Boris, Proceeding of the 4th Conference on Numerical Simulation of Plasmas, 1970, pp. 3–67.
[20] J.L. Vay, Phys. Plasmas 15 (5) (2008) 056701.
[21] T. Zh. Esirkepov, Comput. Phys. Comm. 135 (2) (2001) 144–153.
[22] H. Spohn, Large Scale Dynamics of Interacting Particles, Springer-Verlag, Berlin Heidelberg, 1991.
[23] H. Barucq, B. Hanouzet, Asymptot. Anal. 15 (1) (1997) 25.
[24] Jean-Pierre Berenger, J. Comput. Phys. 114 (2) (1994) 185–200.
[25] Illia Thiele, Stefan Skupin, Rachel Nuter, J. Comput. Phys. 321 (2016) 1110–1119.
[26] 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, J.C. Adam, Lecture Notes in Computer Science, Vol. 2331, Springer, Heidelberg, 2002, pp. 342–351.
[27] A.F. Lifschitz, X. Davoine, E. Lefebvre, J. Faure, C. Rechatin, V. Malka, J. Comput. Phys. 228 (5) (2009) 1803–1814.
[28] Troels Haugbølle, Jacob Trier Frederiksen, Åke Nordlund, Phys. Plasmas 20 (6) (2013) 062904.
[29] George Stantchev, William Dorland, Nail Gumerov, J. Parallel Distrib. Comput. 68 (10) (2008) 1339–1349. General-Purpose Processing using Graphics Processing Units.
[30] Viktor K. Decyk, Tajendra V. Singh, Comput. Phys. Comm. 182 (3) (2011) 641–648.
[31] Kai Germaschewski, William Fox, Stephen Abbott, Narges Ahmadi, Kristofor Maynard, Liang Wang, Hartmut Ruhl, Amitava Bhattacharjee, J. Comput. Phys. 318 (2016) 305–326.
[32] D. Hilbert, Math. Ann. 38 (1891) 459.
[33] J. Trier Frederiksen, G. Lapenta, M.E. Pessah, Particle control in phase space by global k-means clustering, 2015. https://arxiv.org/abs/1504.03849.
[34] A. Beck, J.T. Frederiksen, J. Dérouillat, Nucl. Instrum. Methods Phys. Res. A 829 (2016) 418–421. 2nd European Advanced Accelerator Concepts Workshop - {EAAC} 2015.
[35] B.B. Godfrey, J. Comput. Phys. 15 (4) (1974) 504–521.
[36] R. Lehe, Improvement of the Quality of Laser-Wakefield Accelerators: Towards a Compact Free-Electron Laser (Ph.D. thesis), Ecole Polytechnique, 2014.

[37] Rachel Nuter, Vladimir Tikhonchuk, J. Comput. Phys. 305 (2016) 664–676.
[38] Andrew D. Greenwood, Keith L. Cartwright, John W. Luginsland, Ernest A. Baca, J. Comput. Phys. 201 (2) (2004) 665–684.
[39] J.-L. Vay, C.G.R. Geddes, E. Cormier-Michel, D.P. Grote, J. Comput. Phys. 230 (15) (2011) 5908–5929.
[40] R. Nuter, L. Gremillet, E. Lefebvre, A. Lévy, T. Ceccotti, P. Martin, Phys. Plasmas 18 (3) (2011) 033107.
[41] D. Umstadter, J.K. Kim, E. Dodd, Phys. Rev. Lett. 76 (1996) 2073–2076.
[42] A.M. Perelomov, V.S. Popov, M.V. Terent'ev, Sov. Phys.—JETP 23 (1966) 924.
[43] A.M. Perelomov, V.S. Popov, M.V. Terent'ev, Sov. Phys.—JETP 24 (1967) 207.
[44] M.V. Ammosov, N.B. Delone, V.P. Krainov, Sov. Phys.—JETP 64 (1986) 1191.
[45] P. Mulser, F. Cornolti, D. Bauer, Phys. Plasmas 5 (12) (1998) 4466–4475.
[46] F. Pérez, L. Gremillet, A. Decoster, M. Drouin, E. Lefebvre, Phys. Plasmas 19 (8) (2012) 083104.
[47] K. Nanbu, Phys. Rev. E 55 (1997) 4642–4652.
[48] K. Nanbu, S. Yonemura, J. Comput. Phys. 145 (2) (1998) 639–654.
[49] J.D. Huba, NRL Plasma Formulary, Office of Naval Research, Naval Research Laboratory (U.S.), 2013.
[50] N.E. Frankel, K.C. Hines, R.L. Dewar, Phys. Rev. A 20 (1979) 2120–2129.
[51] Yong-Ki Kim, José Paulo Santos, Fernando Parente, Phys. Rev. A 62 (2000) 052710.
[52] F. Rohrlich, B.C. Carlson, Phys. Rev. 93 (1954) 38–44.
[53] M. Thevenet, A. Leblanc, S. Kahaly, H. Vincenti, A. Vernier, F. Quere, J. Faure, Nat. Phys. 12 (4) (2016) 355–360.
[54] B.C. Stuart, M.D. Feit, A.M. Rubenchik, B.W. Shore, M.D. Perry, Phys. Rev. Lett. 74 (1995) 2248–2251.
[55] D. Ristau, Laser-Induced Damage in Dielectrics with Nanosecond to Subpicosecond Pulses, Taylor & Francis Inc., 2014.
[56] D.W. Forslund, J.M. Kindel, E.L. Lindman, Phys. Fluids 18 (8) (1975) 1002–1016.
[57] Bruce I. Cohen, Claire Ellen Max, Phys. Fluids 22 (6) (1979) 1115–1132.
[58] A.A. Andreev, C. Riconda, V.T. Tikhonchuk, S. Weber, Phys. Plasmas 13 (5) (2006) 053110.
[59] S. Weber, C. Riconda, L. Lancia, J.-R. Marquès, G.A. Mourou, J. Fuchs, Phys. Rev. Lett. 111 (2013) 055004.
[60] M. Chiaramello, C. Riconda, F. Amiranoff, J. Fuchs, M. Grech, L. Lancia, J.R. Marquès, T. Vinci, S. Weber, Phys. Plasmas 23 (7) (2016) 072103.
[61] M. Chiaramello, F. Amiranoff, C. Riconda, S. Weber, Phys. Rev. Lett. 117 (2016) 235003.
[62] J. Fuchs, A.A. Gonoskov, M. Nakatsutsumi, W. Nazarov, F. Quéré, A.M. Sergeev, X.Q. Yan, Eur. Phys. J. Spec. Top. 223 (6) (2014) 1169–1173.
[63] R. Wilson, M. King, R.J. Gray, D.C. Carroll, R.J. Dance, C. Armstrong, S.J. Hawkes, R.J. Clarke, D.J. Robertson, D. Neely, P. McKenna, Phys. Plasmas 23 (3) (2016) 033106.
[64] L. Lancia, J.-R. Marquès, M. Nakatsutsumi, C. Riconda, S. Weber, S. Hüller, A. Mančić, P. Antici, V.T. Tikhonchuk, A. Héron, P. Audebert, J. Fuchs, Phys. Rev. Lett. 104 (2010) 025001.
[65] L. Lancia, A. Giribono, L. Vassura, M. Chiaramello, C. Riconda, S. Weber, A. Castan, A. Chatelain, A. Frank, T. Gangolf, M.N. Quinn, J. Fuchs, J.-R. Marquès, Phys. Rev. Lett. 116 (2016) 075001.
[66] Michael Hesse, Nicolas Aunai, Seiji Zenitani, Masha Kuznetsova, Joachim Birn, Phys. Plasmas 20 (6) (2013) 061210.
[67] First results from NASA's Magnetospheric Multiscale (MMS) Mission, 2016. http://agupubs.onlinelibrary.wiley.com/hub/issue/10.1002/(ISSN)1944-8007.NASAMMS1/. (Online; Accessed 14 February 2016).
[68] P.A. Cassak, M.A. Shay, Phys. Plasmas 14 (10) (2007) 102114.
[69] J.G. Kirk, P. Duffy, J. Phys. G: Nucl. Part. Phys. 25 (8) (1999) R163.
[70] Anatoly Spitkovsky, Astrophys. J. Lett. 682 (1) (2008) L5.
[71] L. Sironi, A. Spitkovsky, J. Arons, Astrophys. J. 771 (1) (2013) 54.
[72] Troels Haugbølle, Astrophys. J. Lett. 739 (2) (2011) L42.
[73] Erich S. Weibel, Phys. Rev. Lett. 2 (1959) 83–84.
[74] A. Bret, L. Gremillet, M.E. Dieckmann, Phys. Plasmas 17 (12) (2010) 120501.
[75] A. Grassi, M. Grech, F. Amiranoff, F. Pegoraro, A. Macchi, C. Riconda, Phys. Rev. E 95 (2017) 023203.
[76] R.D. Blandford, C.F. McKee, Phys. Fluids 19 (8) (1976) 1130–1138.
[77] Pasquale Blasi, Astron. Astrophys. Rev. 21 (1) (2013) 70.
[78] A. Grassi, M. Grech, F. Amiranoff, A. Macchi, C. Riconda, Phys. Rev. E 96 (2017) 033204. http://dx.doi.org/10.1103/PhysRevE.96.033204.
[79] A.A. Golovanov, I.Yu. Kostyulov, J. Thomas, A. Pukhov, Phys. Plasmas 24 (2017) 103104. http://dx.doi.org/10.1063/1.4996856.
[80] M. Lobet, E. d'Humières, M. Grech, C. Ruyer, X. Davoine, L. Gremillet, J. Phys. Conf. Ser. 688 (1) (2016) 012058.
[81] H. Vincenti, J.-L. Vay, Comput. Phys. Comm. 200 (2016) 147–167.
[82] H. Vincenti, J.-L. Vay, 2017. https://arxiv.org/abs/1707.08500.
[83] Axel Huebl, Rémi Lehe, Jean-Luc Vay, David P. Grote, Ivo Sbalzarini, Stephan Kuschel, Michael Bussmann, Openpmd 1.0.0: A Meta Data Standard For Particle And Mesh Based Data, Zenodo, 2015. http://dx.doi.org/10.5281/zenodo.33624.