



PAPER • OPEN ACCESS

PICSAR-QED: a Monte Carlo module to simulate strong-field quantum electrodynamics in particle-in-cell codes for exascale architectures

To cite this article: Luca Fedeli *et al* 2022 *New J. Phys.* **24** 025009

View the [article online](#) for updates and enhancements.

You may also like

- [Non-equilibrium correlations and entanglement in a semiconductor hybrid circuit-QED system](#)
L D Contreras-Pulido, C Emary, T Brandes et al.
- [Witnessing topological Weyl semimetal phase in a minimal circuit-QED lattice](#)
Feng Mei, Zheng-Yuan Xue, Dan-Wei Zhang et al.
- [Output field-quadrature measurements and squeezing in ultrastrong cavity-QED](#)
Roberto Stassi, Salvatore Savasta, Luigi Garziano et al.



OPEN ACCESS

RECEIVED
16 September 2021REVISED
28 November 2021ACCEPTED FOR PUBLICATION
25 January 2022PUBLISHED
1 March 2022

Original content from
this work may be used
under the terms of the
[Creative Commons
Attribution 4.0 licence](#).

Any further distribution
of this work must
maintain attribution to
the author(s) and the
title of the work, journal
citation and DOI.



PAPER

PICSAR-QED: a Monte Carlo module to simulate strong-field quantum electrodynamics in particle-in-cell codes for exascale architectures

Luca Fedeli^{1,*} , Neïl Zaïm¹ , Antonin Sainte-Marie¹ , Maxence Thévenet² ,
Axel Huebl³ , Andrew Myers³ , Jean-Luc Vay³  and Henri Vincenti^{1,*} ¹ Université Paris-Saclay, CEA, CNRS, LIDYL, 91191 Gif-sur-Yvette, France² Deutsches Elektronen-Synchrotron, Notkestraße 85, D-22607 Hamburg, Germany³ Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States of America

* Authors to whom any correspondence should be addressed.

E-mail: luca.fedeli@cea.fr and henri.vincenti@cea.fr**Keywords:** strong-field QED, particle-in-cell codes, nonlinear Breit–Wheeler pair production, inverse Compton photon emission, Monte Carlo methods, exascale supercomputing

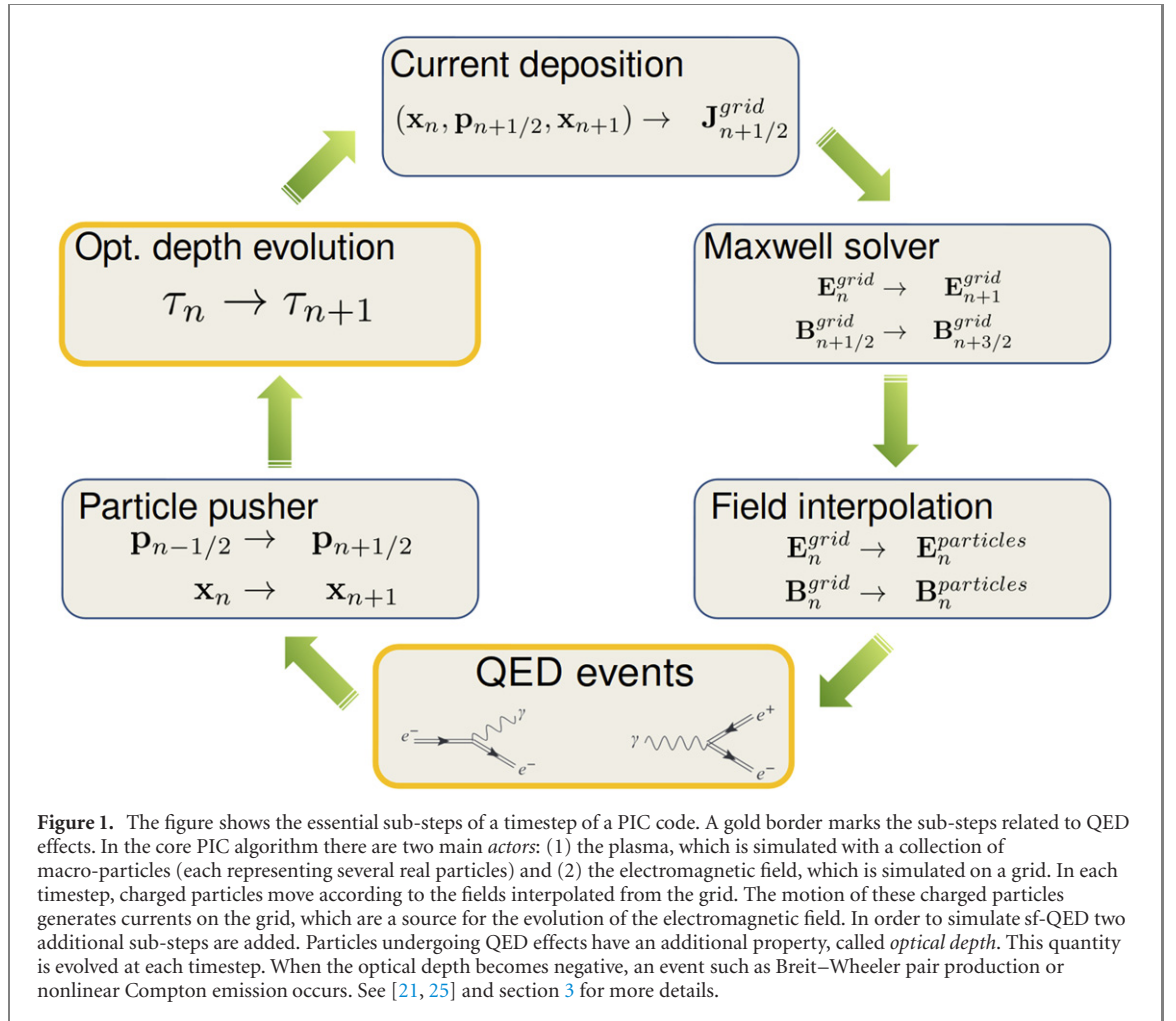
Abstract

Physical scenarios where the electromagnetic fields are so strong that quantum electrodynamics (QED) plays a substantial role are one of the frontiers of contemporary plasma physics research. Investigating those scenarios requires state-of-the-art particle-in-cell (PIC) codes able to run on top high-performance computing (HPC) machines and, at the same time, able to simulate strong-field QED processes. This work presents the PICSAR-QED library, an open-source, portable implementation of a Monte Carlo module designed to provide modern PIC codes with the capability to simulate such processes, and optimized for HPC. Detailed tests and benchmarks are carried out to validate the physical models in PICSAR-QED, to study how numerical parameters affect such models, and to demonstrate its capability to run on different architectures (CPUs and GPUs). Its integration with WarpX, a state-of-the-art PIC code designed to deliver scalable performance on upcoming exascale supercomputers, is also discussed and validated against results from the existing literature.

1. Introduction

One of the frontiers of modern physics research deals with physical scenarios where the electromagnetic fields are so strong that quantum electrodynamics (QED) plays a substantial role, specifically in the so-called *strong-field* regime (sf-QED). These scenarios range from the interaction region of an ultra-intense laser pulse with a plasma [1–11] to extreme astrophysical objects, such as pulsar magnetospheres [12, 15–17], black-holes [13, 14], or gamma-ray bursts [18]. Numerical modeling is essential to gain insights into these scenarios and to assist the experimental investigation of ultra-intense laser–matter interaction. Since a kinetic description of the plasma is usually required, particle-in-cell (PIC) codes [19–21] are often the numerical tool of choice. Moreover, PIC simulations can include the most relevant sf-QED effects in these regimes [22–26], such as the emission of high-energy photons via the *nonlinear Compton* process [27] (also known as *nonlinear synchrotron emission*) and the decay of a high-energy photon into an electron–positron pair via the *nonlinear Breit–Wheeler pair production* process [27–31] (see figure 1 for a scheme showing the core algorithms of a PIC code including sf-QED effects).

Realistic simulations often demand a substantial amount of computational resources, to the point that the most ambitious numerical campaigns can only be performed on the most powerful high-performance computing (HPC) facilities [32]. Most of those machines offload a conspicuous fraction of their calculations to specialized hardware [33] (i.e. graphics processing units, GPUs) or make use of CPUs specifically designed for HPC needs [34]. Few PIC codes in use in the plasma physics community can efficiently take advantage of those machines. Moreover, only a fraction of these codes is distributed as free



and open-source software. Requiring the capability to simulate at least the most relevant QED processes further restricts the choice.

OSIRIS [35], Picador [36], VPIC 2.0 [37], and PConGPU [38] are popular codes able to take advantage of modern, GPU-based supercomputers. However, only PConGPU and VPIC 2.0 are available as free and open-source software. Moreover, while OSIRIS and Picador have very comprehensive QED modules, PConGPU does not implement nonlinear Breit–Wheeler pair production yet, and VPIC 2.0 does not implement QED processes. Smilei [39], EPOCH [21], and Tristan-MP v2 [40] are well-known open-source, massively parallel PIC codes with comprehensive QED modules, but they are not currently designed to take advantage of GPU-based supercomputers. Calder [24, 41] is also well-known for its very comprehensive QED modules [26, 42], but it is not open-source nor optimized for large-scale GPU-based HPC machines.

This paper presents PICSAR-QED [43], a module part of the PICSAR library [44–47], which has been coupled with the WarpX [48, 49] PIC code in order to simulate sf-QED processes relevant for extreme plasma physics scenarios. WarpX is an open-source code developed within the framework of the Exascale Computing Project [50] and designed to provide scalable performance on upcoming exascale.

PICSAR-QED implements primitives designed to be portable across different architectures (CPUs and GPUs). It is conceived to be easily included in existing projects, and is released as a standalone open-source project, contributing a carefully validated module to the open-source PIC ecosystem, of which other PIC codes not currently implementing sf-QED could take advantage.

In this work, we first review the physical models implemented in PICSAR-QED (section 2). Then, we review the numerical methods to model sf-QED processes in PIC codes, we discuss the specific implementation choices made in PICSAR-QED, and we present a detailed validation of the methods provided by the library (section 3). In section 4 we show performance benchmarks on different architectures. Finally, we describe in section 5 the integration of PICSAR-QED in WarpX and present benchmarks with existing results from the literature.

2. Physical processes implemented in PICSAR-QED

The extreme plasma physics scenarios mentioned in the introduction are characterized by electromagnetic fields so strong that relevant QED processes pertain to the strong-field regime of QED. This regime is characterized by the QED critical field scale E_s , also known as *Schwinger field* [51–53]:

$$E_s = \frac{m_e^2 c^3}{q_e \hbar} \approx 1.32 \times 10^{18} \text{ V m}^{-1} \quad (1)$$

where m_e is the electron mass, c is the speed of light, q_e is the elementary charge and \hbar is the reduced Planck constant. Achieving such a tremendous field directly in the laboratory is far beyond current experimental capabilities, being roughly three orders of magnitude higher than the strongest electric fields available on Earth [54]. However, it can be approached in the rest frame of ultra-relativistic particles. Indeed, for a particle with four-momentum p^μ propagating in a region where the electromagnetic field tensor is $F^{\mu\nu}$, the actual parameter of interest for sf-QED is the parameter χ defined as follows [30]:

$$\chi = \frac{|F_{\mu\nu} p^\nu|}{E_s m_e c} = \frac{\gamma}{E_s} \sqrt{(\mathbf{E} + \mathbf{v} \times \mathbf{B})^2 - \left(\frac{\mathbf{v} \cdot \mathbf{E}}{c}\right)^2} \quad (2)$$

where \mathbf{E} is the electric field, \mathbf{B} is the magnetic field, \mathbf{v} is the velocity of the particle, and $\gamma = \varepsilon/(m_e c^2)$ is its energy normalized to the electron mass energy. χ is Lorentz-invariant and is called *quantum non-linearity parameter*. For a general particle and generic configurations such that \mathbf{B} is not aligned with \mathbf{E} , it takes a simple interpretation in the ultra-relativistic limit $\varepsilon \simeq \mathbf{c} \mathbf{p} \gg m_e c^2$ as the electric field value in units of E_s in the instantaneous rest frame of a (fictitious) particle with three-momentum \mathbf{p} and mass m_e . For electrons and positrons, χ equals this ratio exactly and without restrictions. For high-energy photons with four-momentum p^μ , the quantum non-linearity parameter, which we will refer to as χ_γ , reads as follows:

$$\chi_\gamma = \frac{|F_{\mu\nu} p^\nu|}{E_s m_e c} = \frac{\gamma_\gamma}{E_s} \sqrt{(\mathbf{E} + \mathbf{c} \times \mathbf{B})^2 - \left(\frac{\mathbf{c} \cdot \mathbf{E}}{c}\right)^2} \quad (3)$$

where γ_γ is the photon energy normalized to $m_e c^2$ and \mathbf{c} is a velocity vector with a magnitude equal to the speed of light.

We attain the so-called *full quantum regime* of sf-QED when $\chi > 1$, while strong-field QED effects rapidly vanish for $\chi < 1$. The physical models used to include the most relevant sf-QED processes in PIC codes—nonlinear Breit–Wheeler pair production and nonlinear Compton emission—are well known from the literature [22, 23, 25–29, 31, 55] and are briefly reviewed below. Schwinger pair production, which is a particularly extreme physical process where the electromagnetic field is strong enough to generate electron–positron pairs from the quantum fluctuations of the vacuum, is also briefly discussed.

Following the common practices in the existing literature, the theoretical framework presented in this section is restricted to the validity of several approximations. First, we assume that electrons and positrons are ultra-relativistic, and that high-energy photons have significantly more energy than the rest mass of an electron: $\varepsilon \gg m_e c^2$. This greatly simplifies the determination of the properties of the generated particles, since their momenta can be safely assumed to be collinear with that of the emitting particle. Then, all the reaction rates are calculated within the so-called *locally constant field approximation* [4, 10, 30, 56], which requires that $a_0 \gg 1$ and $a_0^3 \gg \chi$ ($a_0 = q_e \sqrt{(F_{\mu\nu} p^\nu)^2} / (k_\mu p^\mu m_e c^2)$ is the normalized amplitude of an electromagnetic wave with wave-vector k_μ and field tensor $F_{\mu\nu}$), and in the crossed-field approximation. The latter is generally well satisfied, since any macroscopic field appears like a plane-wave in the rest frame of an ultra-relativistic particle [57], and since in a sufficiently intense field charged particles are quickly accelerated up to relativistic energies. In addition, the field Lorentz-invariants, which vanish for a plane wave, must remain small, which is ensured in particular if the field amplitude in the laboratory frame is significantly smaller than the Schwinger field [58] (i.e. $|\mathbf{E}| \ll E_s$ and $|\mathbf{B}| \ll E_s/c$). Finally, we assume to be far from the so-called *fully nonperturbative regime* of sf-QED, a regime that still defies the formulation of a complete theory [59, 60]. Therefore we require that $\chi^{2/3} \ll 1/\alpha_f \approx 137$ (where α_f is the fine structure constant).

2.1. Nonlinear Breit–Wheeler pair production

Nonlinear Breit–Wheeler pair production is the decay of a high-energy photon propagating in a strong field into an electron–positron pair. The differential pair production rate for a photon with quantum parameter

χ_γ reads as follows [24, 28]:

$$\frac{d^2N}{dt d\chi_-} = \frac{\alpha_f m_e c^2}{\pi \sqrt{3} \hbar \gamma_\gamma \chi_\gamma} \left[\int_{X(\chi_\gamma, \chi_-)}^{+\infty} \sqrt{s} K_{1/3} \left(\frac{2}{3} s^{3/2} \right) ds - \left(2 - \chi_\gamma X^{3/2}(\chi_\gamma, \chi_-) \right) K_{2/3} \left(\frac{2}{3} X^{3/2}(\chi_\gamma, \chi_-) \right) \right] \quad (4)$$

where χ_+ and χ_- are respectively the quantum parameter of the emitted positron and of the emitted electron, K_α are the modified Bessel functions of the second kind of order α , α_f is the fine structure constant, and

$$X(\chi_\gamma, \chi_-) = \left(\frac{\chi_\gamma}{\chi_+ \chi_-} \right)^{2/3}. \quad (5)$$

Since particles in very intense electromagnetic fields are usually ultra-relativistic (i.e. $\gamma \gg 1$), the so-called *ultra-relativistic* approximation can be used. In this approximation, the three-momenta of the product particles are aligned with that of the high-energy photon. Moreover, since $|\mathbf{v}| \approx c$, the square root terms in the expressions for χ and χ_γ become almost identical. Finally, since $\gamma \approx |\mathbf{p}|/(m_e c)$, using total momentum conservation we can write:

$$\chi_+ + \chi_- = \chi_\gamma. \quad (6)$$

This result is used to replace χ_+ with $\chi_\gamma - \chi_-$, so that the differential pair production rate and the total pair production rate can be rewritten as follows:

$$\frac{d^2N}{dt d\chi_-} = \frac{\alpha_f m_e c^2}{\hbar \gamma_\gamma} \chi_\gamma F(\chi_\gamma, \chi_-) \quad (7)$$

$$\frac{dN}{dt} = \frac{\alpha_f m_e c^2}{\hbar \gamma_\gamma} \chi_\gamma T(\chi_\gamma) \quad (8)$$

where

$$T(\chi_\gamma) = \int_0^{\chi_\gamma} F(\chi_\gamma, u) du = \frac{1}{\pi \sqrt{3} \chi_\gamma^2} \int_0^{\chi_\gamma} \left[\int_{X(\chi_\gamma, u)}^{+\infty} \sqrt{s} K_{1/3} \left(\frac{2}{3} s^{3/2} \right) ds - \left(2 - \chi_\gamma X^{3/2}(\chi_\gamma, u) \right) K_{2/3} \left(\frac{2}{3} X^{3/2}(\chi_\gamma, u) \right) \right] du. \quad (9)$$

It is noteworthy that very good and simple asymptotic approximations exist for $T(\chi_\gamma)$ [28]:

$$T(\chi_\gamma) \sim 0.16 \frac{K_{1/3}^2 \left(\frac{2}{3\chi_\gamma} \right)}{\chi_\gamma} \sim \begin{cases} \exp \left(-\frac{2}{3\chi_\gamma} \right) & \chi_\gamma \ll 1 \\ \chi_\gamma^{-1/3} & \chi_\gamma \gg 1 \end{cases}. \quad (10)$$

Equation (8) allows determining the probability of a photon to decay into an electron–positron pair within a given time interval. Indeed, if the timestep of the simulation is Δt , this probability is

$p_{\text{decay}} = 1 - \exp(-\Delta t dN/dt)$. In a Monte Carlo approach, a random number r_{decay} from a uniform probability distribution between zero and one can be drawn and compared with p_{decay} : if $r_{\text{decay}} < p_{\text{decay}}$ Breit–Wheeler pair production occurs for the given photon (or, in a PIC simulation, a macro-photon, that is a numerical particle representing several real photons). In practice, however, a different approach is typically followed [23], in order to avoid a random number extraction per particle at each iteration. In this approach, each macro-photon has a randomly initialized quantity, called *optical depth*, which is reduced at each time-step according to the total pair production rate. As soon as this quantity reaches zero, Breit–Wheeler pair production occurs (see section 3 for a more in-depth discussion).

The energy of the generated particles can be determined using equation (7), by calculating the cumulative probability distribution with respect to χ_- :

$$P(\chi_\gamma, \chi_-) = \frac{\int_0^{\chi_-} F(\chi_\gamma, u) du}{\int_0^{\chi_\gamma} F(\chi_\gamma, u) du}. \quad (11)$$

The quantum parameter of the electron χ_- can be sampled by solving $P(\chi_\gamma, \chi_-) = r$, where r is a random number drawn from a uniform probability distribution in the range $[0, 1]$. The quantum parameter of the positron is then simply $\chi_+ = \chi_\gamma - \chi_-$. In the ultra-relativistic limit, determining the energy and momenta of the generated particles is straightforward if the quantum parameters χ_\pm are known. Indeed, their kinetic energy can be calculated as $K_\pm = (\gamma_\gamma - 2)\chi_\pm/\chi_\gamma$.

2.2. Nonlinear Compton photon emission

The nonlinear Compton photon emission is the emission of a high-energy photon from a charged particle (e.g. an electron or a positron) propagating in a strong electromagnetic field. PICSAR-QED implements the model described in [23], which is summarized here for completeness (it is worth noting that a slightly different notation is adopted: in [23] χ is replaced with η and χ_γ is replaced with 2χ).

The differential emission rate for the nonlinear Compton scattering process reads as follows:

$$\frac{d^2N}{dt d\chi_\gamma} = \frac{2}{3} \frac{\alpha m_e c^2}{\hbar} \frac{1}{\gamma} \frac{\frac{\sqrt{3}}{2\pi} (\chi_\gamma/\chi) \left[\int_{Y(\chi, \chi_\gamma/\chi)}^{\infty} K_{5/3}(s) ds + \frac{(\chi_\gamma/\chi)^2}{1-(\chi_\gamma/\chi)} K_{2/3}(Y(\chi, \chi_\gamma/\chi)) \right]}{\chi_\gamma} \quad (12)$$

where

$$Y(\chi, \chi_\gamma/\chi) = \frac{2}{3} \frac{\chi_\gamma/\chi}{\chi(1 - \chi_\gamma/\chi)}. \quad (13)$$

Equation (12) can be re-written as

$$\frac{d^2N}{dt d\chi_\gamma} = \frac{2}{3} \frac{\alpha m_e c^2}{\hbar} \frac{1}{\gamma} \frac{S(\chi, \xi)}{\xi \chi} \quad (14)$$

where we introduced $\xi = \chi_\gamma/\chi$. As for Breit–Wheeler pair production, the ultra-relativistic approximation applies. Therefore, since the photon is emitted within a cone of amplitude $\alpha \approx 1/\gamma$, for $\gamma \gg 1$ the photon can be safely considered to be emitted along the direction of the momentum of the emitting particle.

Within the ultra-relativistic approximation, it is also trivial to show that $\xi < 1$.

The total emission rate is obtained by integrating equation (14) over χ_γ from 0 up to χ :

$$\frac{dN}{dt} = \frac{2}{3} \frac{\alpha m_e c^2}{\hbar} \frac{1}{\gamma} G(\chi) \quad (15)$$

where

$$G(\chi) = \int_0^\chi \frac{S(\chi, u/\chi)}{u} du = \int_0^1 \frac{S(\chi, \xi)}{\xi} d\xi = \int_0^1 \frac{\sqrt{3}}{2\pi} \left[\left(\int_{Y(\chi, \xi)}^{\infty} K_{5/3}(s) ds \right) + \frac{\xi^2}{1-\xi} K_{2/3}(Y(\chi, \xi)) \right] d\xi \quad (16)$$

Equation (16) allows determining the probability of an electron or a positron to emit a high-energy photon via nonlinear Compton emission, with a procedure identical to that described for Breit–Wheeler pair production. As for Breit–Wheeler pair production, the quantum parameter χ_γ of the generated photon is determined using the cumulative probability distribution:

$$P(\chi, \xi) = \frac{\int_0^\xi S(\chi, u) du}{\int_0^1 S(\chi, u) du}. \quad (17)$$

Once χ_γ is known, the energy of the generated photons can be determined trivially using the ultra-relativistic approximation: $\gamma_\gamma = (\gamma - 1)\xi$. Finally, the kinetic energy of the emitting particle must be reduced by $\gamma_\gamma m_e c^2$. Using $(\gamma - 1)$ instead of γ to calculate γ_γ ensures that the total energy is conserved.

2.3. Schwinger pair production

Schwinger pair production is the generation of electron–positron pairs from the fluctuations of the quantum vacuum in the presence of a sufficiently strong electromagnetic field. An expression for the Schwinger pair production rate per unit volume can be obtained within the locally constant field approximation (i.e. by assuming that the field is constant during a timestep inside a simulation cell) and can be found in [61]:

$$\frac{d^2N}{dt dV} = \frac{q_e^2 E_s^2}{4\pi^2 \hbar^2 c} \epsilon \eta \coth\left(\frac{\pi \eta}{\epsilon}\right) \exp\left(-\frac{\pi}{\epsilon}\right) \quad (18)$$

where $\epsilon = \mathcal{E}/E_s$ and $\eta = \mathcal{H}/E_s$. \mathcal{E} and \mathcal{H} are given by

$$\mathcal{E} = \sqrt{\sqrt{\mathcal{F}^2 + \mathcal{G}^2} + \mathcal{F}} \quad \mathcal{H} = \sqrt{\sqrt{\mathcal{F}^2 + \mathcal{G}^2} - \mathcal{F}} \quad (19)$$

where \mathcal{F} and \mathcal{G} are the invariants of the electromagnetic field and are equal to

$$\mathcal{F} = (\mathbf{E}^2 - c^2 \mathbf{B}^2)/2 \quad (20)$$

$$\mathcal{G} = c \mathbf{E} \cdot \mathbf{B}. \quad (21)$$

Electron–positron pairs generated via the Schwinger pair production process can be initialized at rest. In principle, the electromagnetic field should lose an amount of energy equal to $2m_e c^2$ when pairs are created via the Schwinger process. However, since the field loses significantly more energy while accelerating these particles to relativistic velocities immediately after their creation, the small energy loss due to the rest-mass energy of the pair can be safely disregarded.

It may be noted that the Schwinger process becomes relevant when the field invariants acquire a significant value (i.e. they start to approach E_s^2). This corresponds to electromagnetic fields for which the crossed-field approximation, which is assumed in the nonlinear Breit–Wheeler and nonlinear Compton modules, starts to break. As such, there may be situations where nonlinear Breit–Wheeler and nonlinear Compton emission are no longer accurately modeled at the positions in spacetime for which Schwinger pair production is significant (this is especially true if the field invariants become equal or greater than the χ parameter). Let us note that this does not *a priori* preclude the accurate modeling of physical configurations where both Schwinger and Breit–Wheeler processes play a role, e.g. if Schwinger pair creation happens in spite of small field invariants (the local rate suppression being overcome by the large interaction volume [61]) or if field invariants become significant in a spacetime region so small compared to the overall strong-field region that the inaccuracy due to the use of crossed-field amplitudes for other processes at those points can be safely neglected. However, a detailed discussion of the validity of the nonlinear Breit–Wheeler and nonlinear Compton modules in the presence of strong invariants goes beyond the scope of this paper and should be the subject of a dedicated publication.

Schwinger pair production is implemented in PICSAR-QED. However, since the implementation is relatively simple with respect to the other sf-QED process (the pair production rate is not very expensive to compute and product particles are initially at rest), it will not be mentioned in the following section. Its coupling to the PIC code WarpX will be briefly discussed in section 5.

3. Numerical implementation

The total production rates for Breit–Wheeler pair production and nonlinear Compton emission have quite complex expressions, featuring special functions and multiple integrals, which would be too computationally expensive to evaluate for each particle at each time step. Indeed, in the standard PIC algorithm, the number of operations per particle per timestep is relatively small if compared with what would be required to compute QED rates. Therefore, their evaluation at runtime would largely dominate the simulation time, unacceptably slowing down the simulation. For this reason, as documented in the literature [23, 24, 26], the standard approach is to reformulate the total production rates as a product between simple numerical factors and a numerically expensive function, which is pre-computed and stored in a one-dimensional lookup table. For instance, the right-hand side of equation (8) is the product of a constant ($\alpha_f m_e c^2 / \hbar$), the normalized photon energy γ_γ (which is a simple function of the photon momentum), the quantum parameter χ_γ (which is a simple function of the photon momentum and of the electromagnetic field), and the function $T(\chi_\gamma)$, which contains all the other terms of the total cross section (see equation (9)). Similarly, for the nonlinear Compton emission total production rate, all the numerically expensive terms can be absorbed into the $G(\chi)$ function (see equations (15) and (16)). The cumulative probability distributions—required to determine the properties of the product particles—are also impractical to compute at runtime. Therefore, equations (11) and (17) are pre-computed over a finite set of parameters and the result is stored in two-dimensional lookup tables.

As mentioned in section 2, in principle a random number per particle at each timestep should be drawn in order to determine if a sf-QED process occurs (two underlying assumptions are that the QED production rates do not vary significantly over one timestep and that the probability of a QED process to occur during a timestep is significantly smaller than one). However, generating pseudo-random numbers can have a significant numerical cost, depending on the algorithm. Therefore, as documented in the literature [23], the preferred approach is to assign a quantity τ , called *optical depth*, to each particle which may undergo a sf-QED process. τ is extracted from an exponential probability distribution $dp/ds = \exp(-s)$, and at each iteration it is updated as $\tau_{n+1} = \tau_n - dN/dt \Delta t$, where dN/dt is the total production rate of either Breit–Wheeler pair production or nonlinear Compton photon emission. In the case of the nonlinear Compton process, τ is reinitialized with an exponential probability distribution after each photon emission

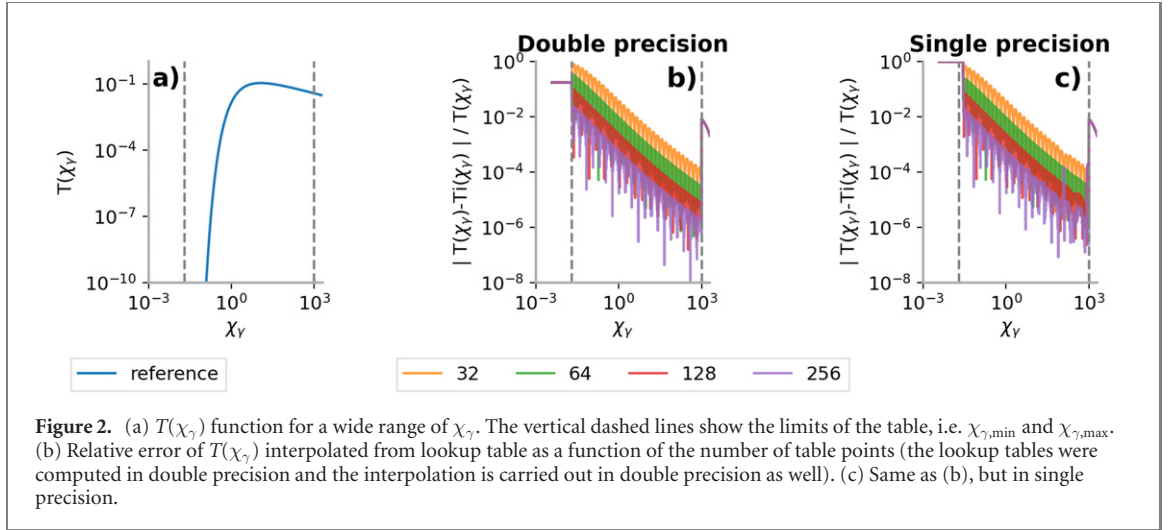


Figure 2. (a) $T(\chi_\gamma)$ function for a wide range of χ_γ . The vertical dashed lines show the limits of the table, i.e. $\chi_{\gamma,\min}$ and $\chi_{\gamma,\max}$. (b) Relative error of $T(\chi_\gamma)$ interpolated from lookup table as a function of the number of table points (the lookup tables were computed in double precision and the interpolation is carried out in double precision as well). (c) Same as (b), but in single precision.

event. Under the hypothesis that the probability of multiple events occurring in a single time-step is negligible, this second approach is equivalent to the former, but with a reduced computational cost. Indeed, if k_i is the total production rate during step i and Δt is the time-step, the probability of an event after n steps is $P(n) = \exp(-\sum_{i=1}^{n-1} k_i \Delta t) [1 - \exp(-k_n \Delta t)]$. Using the optical depth, for this to occur the initial optical depth τ_0 must be $\sum_{i=1}^{n-1} k_i \Delta t < \tau_0 < \sum_{i=1}^n k_i \Delta t$. Since τ_0 is drawn from a probability distribution $dp/ds = \exp(-s)$, the probability of initializing τ_0 within this range is $\int_A^B \exp(-s) ds$, where $A = \sum_{i=1}^{n-1} k_i \Delta t$ and $B = \sum_{i=1}^n k_i \Delta t$. The resulting probability is exactly equal to $P(n)$. From a numerical point of view, the simpler loop on the particles to update the optical depth offers more opportunities for the compiler to optimize the code (e.g. exploiting *single instruction on multiple data* parallelization on CPU architectures).

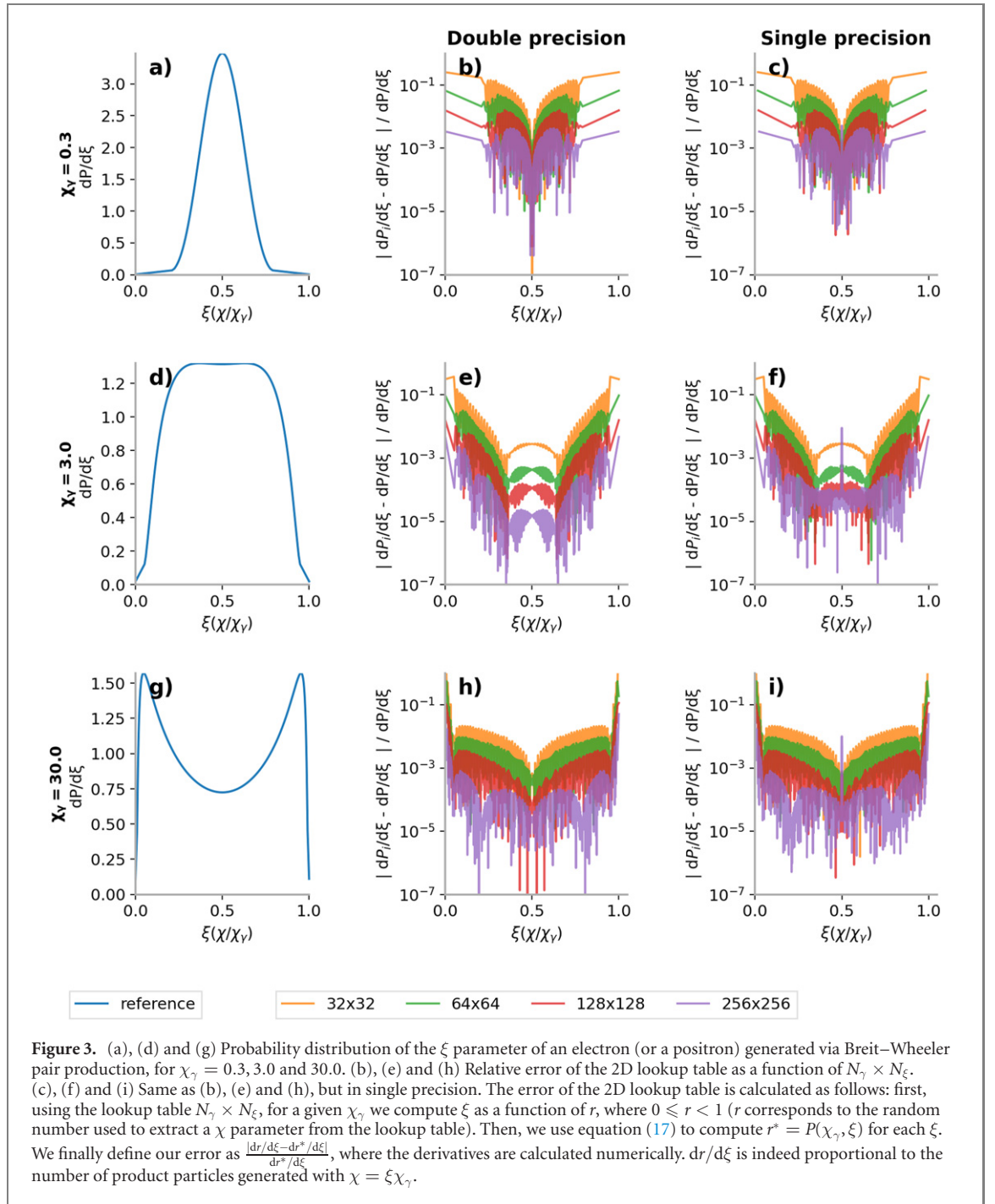
This section describes the specific implementation choices made for PICSAR-QED. In particular, we provide details on how lookup tables are calculated and how interpolation within these tables is performed. We also assess how the precision of the lookup tables (number of points and use of single or double precision) affects the accuracy of the results. This is particularly important from the perspective of a user. Indeed, although the general idea of the method to implement sf-QED processes in PIC codes has already been described in the literature, to the best of the authors' knowledge, detailed guidelines on how to choose the parameters of the lookup tables have never been published. At the end of the section, we finally discuss specific choices aimed at achieving portability across multiple architectures.

Before delving into the implementation choices made for PICSAR-QED, it is important to clarify that methods to compute the lookup tables are provided for CPU architectures only. This is due to the fact that they require special functions not yet implemented for GPUs (e.g. Bessel functions of fractional order), and rely on the CPU-only Boost library for sophisticated quadrature methods, such as *tanh-sinh* [62, 63]. Computing the lookup tables typically requires only few tens of seconds on a multi-core CPU, so, in principle, they could be generated at the beginning of each simulation. In practice, it is often more convenient to store them on disk and load them whenever needed (lookup tables typically require only few megabytes of storage).

3.1. Nonlinear Breit–Wheeler pair production: implementation choices and benchmarks

Two lookup tables are needed for Breit–Wheeler pair production: a one-dimensional table for $T(\chi_\gamma)$ and a two-dimensional table for the cumulative probability distribution $P(\chi_\gamma, \chi)$. For $T(\chi_\gamma)$, PICSAR-QED library adopts a solution very similar to that described for the Smilei PIC code [24, 39]. The $T(\chi_\gamma)$ table is generated between a minimum value $\chi_{\gamma,\min}$ and a maximum value $\chi_{\gamma,\max}$, with N_{χ_γ} points logarithmically distributed between the extrema (actually, $\ln T$ is stored in the table). The choice of a logarithmic scale for χ_γ allows spanning several orders of magnitude with a limited number of points, following the strategy proposed in [23]. Outside the extrema of the table, we use the approximations in equation (10). In practice, this is not a significant issue, provided that $\chi_{\min} \lesssim 0.1$ and $\chi_{\max} \gtrsim 1000$, since the asymptotic limit in equation (10) is a very good approximation (besides, at $\chi_\gamma \sim 0.1$ Breit–Wheeler pair production rapidly becomes negligible). On the other hand, in the range $\chi_{\min} < \chi_\gamma < \chi_{\max}$, we perform an interpolation. Specifically, in order to calculate $T(\chi_\gamma^*)$, we first individuate two contiguous tabulated values $\chi_{\gamma,n}$ and $\chi_{\gamma,n+1}$ such that $\chi_{\gamma,n} \leq \chi_\gamma^* < \chi_{\gamma,n+1}$. Then, we compute $T(\chi_\gamma^*)$ as:

$$T(\chi_\gamma^*) = \exp \left(\ln T_{\gamma,n} + (\ln \chi_\gamma^* - \ln \chi_{\gamma,n}) \frac{\ln T_{\gamma,n+1} - \ln T_{\gamma,n}}{\ln \chi_{\gamma,n+1} - \ln \chi_{\gamma,n}} \right) \quad (22)$$



where $T_{\gamma,n}$ and $T_{\gamma,n+1}$ are the tabulated values corresponding to $\chi_{\gamma,n}$ and $\chi_{\gamma,n+1}$. Figure 2 provides detailed results on how different choices of N_{χ_γ} and performing all the calculations in single or double precision affects the accuracy of the table. As expected, we find that increasing the number of table points reduces the error. In order to achieve an error below few percents for $\chi_\gamma > 0.1$, tables must be calculated with at least 128 points. Calculating the tables and performing the interpolation in single or in double precision does not seem to affect the final error significantly.

For $P(\chi_\gamma, \chi)$, PICSAR-QED adopts a significantly more complex strategy, which partially differs with respect to implementations described elsewhere. First of all, we consider $P(\chi_\gamma, \xi)$, where $\xi = \chi/\chi_\gamma$, so that $0 \leq \xi \leq 1$. Moreover, we can exploit the symmetry $P(\chi_\gamma, \xi) = 1 - P(\chi_\gamma, 1 - \xi)$ to store the table only in the range $0 \leq \xi \leq 0.5$. $P(\chi_\gamma, \xi)$ is then generated in the range $\chi_{\gamma,\min} \leq \chi_\gamma \leq \chi_{\gamma,\max}$, with N_{χ_γ} points logarithmically distributed between the extrema, and in the range $0 \leq \xi \leq 0.5$, with N_ξ linearly spaced points. If $\chi_\gamma < \chi_{\gamma,\min}$ or $\chi_\gamma > \chi_{\gamma,\max}$, we replace χ_γ with either $\chi_{\gamma,\min}$ or $\chi_{\gamma,\max}$. This means that a user must choose those extrema in such a way that the whole χ_γ range relevant for a given application is included in the table.

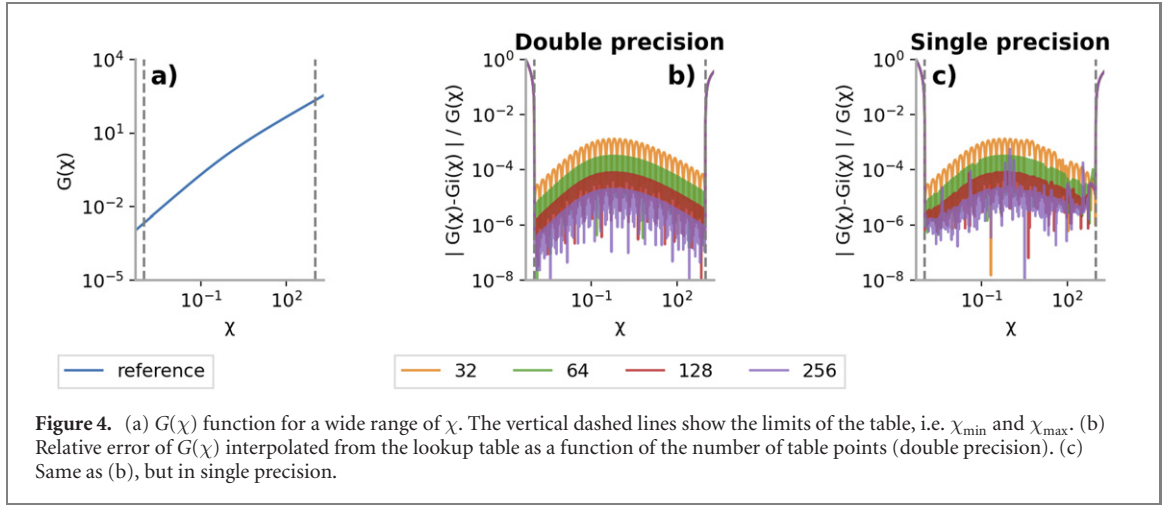


Figure 4. (a) $G(\chi)$ function for a wide range of χ . The vertical dashed lines show the limits of the table, i.e. χ_{\min} and χ_{\max} . (b) Relative error of $G(\chi)$ interpolated from the lookup table as a function of the number of table points (double precision). (c) Same as (b), but in single precision.

When a photon with χ_γ decays into an electron–positron pair via nonlinear Breit–Wheeler pair production, a random number r is extracted from a uniform distribution between 0 and 1, which is used to calculate the quantum parameters of the generated particles. If $r \leq 0.5$, χ is the quantum parameter of the generated electron. Otherwise χ is the quantum parameter of the generated positron. In this second case, we replace $r \rightarrow (1 - r)$, so as to enforce $0 \leq r \leq 0.5$. At this point, as in the previous case, we individuate the two contiguous tabulated values $\chi_{\gamma,n}$ and $\chi_{\gamma,n+1}$ such that $\chi_{\gamma,n} \leq \chi_\gamma^* < \chi_{\gamma,n+1}$. We can now define N_ξ new values \mathcal{P}_m :

$$\mathcal{P}_m = P_{n,m} + (\ln \chi_\gamma - \ln \chi_{\gamma,n}) \frac{P_{n+1,m} - P_{n,m}}{\ln \chi_{\gamma,n+1} - \ln \chi_{\gamma,n}} \quad (23)$$

where $P_{n,m}$ is the table value corresponding to $\chi_{\gamma,n}$ and ξ_m . By performing a binary search, we can find m^* such that:

$$\mathcal{P}_{m^*} \leq r < \mathcal{P}_{m^*+1}. \quad (24)$$

We can finally calculate χ with a second linear interpolation:

$$\chi / \chi_\gamma = \left(\xi_{m^*} + (r - \mathcal{P}_{m^*}) \frac{\xi_{m^*+1} - \xi_{m^*}}{\mathcal{P}_{m^*+1} - \mathcal{P}_{m^*}} \right). \quad (25)$$

Figure 3 provides detailed results on how different choices of $N_{\chi_\gamma} \times N_\xi$ and performing all the calculations in single or double precision affect the accuracy of the table. Also in this case, increasing table resolution results in a better precision of the table and a minimum resolution of 64–128 points in each dimension is required to keep the relative error below few percents. Again, performing the calculations in single precision does not affect these conclusions significantly.

3.2. Nonlinear Compton photon emission: implementation choices and benchmarks

As for nonlinear Breit–Wheeler, two lookup tables are needed for nonlinear Compton photon emission: a one-dimensional table for $G(\chi)$ and a two-dimensional table for the cumulative probability distribution $P(\chi, \chi_\gamma)$. For $G(\chi)$, the PICSAR-QED library adopts a solution very similar to that described for the Smilei PIC code [24, 39]. The $G(\chi)$ table is generated between a minimum value χ_{\min} and a maximum value χ_{\max} , with N_χ points logarithmically distributed between the extrema (actually, $\ln G$ is stored in the table). The choice of a logarithmic scale for χ allows spanning several orders of magnitude with a limited number of points, following the strategy proposed in [23]. Outside the extrema of the table we use either the first or the last value stored in the table, while within this range we perform an interpolation (which means that a user must select those extrema in order to cover all the χ range of interest). Specifically, in order to calculate $G(\chi^*)$, we first individuate the two contiguous tabulated values χ_n and χ_{n+1} such that $\chi_n \leq \chi^* < \chi_{n+1}$. Then, we compute $G(\chi^*)$ as:

$$G(\chi^*) = \exp \left(\ln G_n + (\ln \chi^* - \ln \chi_n) \frac{\ln G_{n+1} - \ln G_n}{\ln \chi_{n+1} - \ln \chi_n} \right) \quad (26)$$

where G_n and G_{n+1} are the tabulated values corresponding to χ_n and χ_{n+1} . Figure 4 provides detailed results on how different choices of N_χ and performing all the calculations in single or double precision affects the accuracy of the table. As for Breit–Wheeler pair production, increasing the number of table points reduces the error and calculating the tables and performing the interpolation in single or in double

precision does not seem to affect the final error significantly. However, in this case, a resolution as low as 32 points is already enough to reduce the error below the percent level across the whole χ range considered here. As far as $P(\chi, \chi_\gamma)$ is of concern, as for Breit–Wheeler pair production, PICSAR-QED adopts a significantly more complex strategy, which partially differs with respect to implementations described elsewhere. First of all, we consider $P(\chi, \xi)$, where $\xi = \chi_\gamma/\chi$, so that $0 \leq \xi \leq 1$. $P(\chi, \xi)$ is then generated in the range $\chi_{\min} \leq \chi \leq \chi_{\max}$, with N_χ points logarithmically distributed between the extrema, and in the range $\xi_{\min} \leq \xi \leq 1$, with N_ξ logarithmically distributed points. ξ_{\min} must be low enough that photons below the threshold contribute negligibly to the total energy loss via nonlinear Compton scattering. If $\chi < \chi_{\min}$ or $\chi > \chi_{\max}$, we replace χ with either χ_{\min} or χ_{\max} , so that the extrema must be selected in order to include all the χ range of interest. In the table we actually store $\ln P$ instead of P .

When an electron or a positron with quantum parameter χ emits a high-energy photon via nonlinear Compton process, a random number r is extracted from a uniform distribution between 0 and 1. At this point, as in the previous case, we individuate the two contiguous tabulated values χ_n and χ_{n+1} such that $\chi_n \leq \chi^* < \chi_{n+1}$. We can now define:

$$\ln \mathcal{P}_m = \ln P_{n,m} + (\ln \chi_\gamma - \ln \chi_{\gamma,n}) \frac{\ln P_{n+1,m} - \ln P_{n,m}}{\ln \chi_{\gamma,n+1} - \ln \chi_{\gamma,n}} \quad (27)$$

where $P_{n,m}$ is the table value corresponding to χ_n and ξ_m . By performing a binary search, we can find m^* such that:

$$\ln \mathcal{P}_{m^*} \leq \ln r < \ln \mathcal{P}_{m^*+1}. \quad (28)$$

We can finally calculate χ_γ with a second linear interpolation:

$$\chi_\gamma = \chi \exp \left(\ln \xi_{m^*} + (\ln r - \ln \mathcal{P}_{m^*}) \frac{\ln \xi_{m^*+1} - \ln \xi_{m^*}}{\ln \mathcal{P}_{m^*+1} - \ln \mathcal{P}_{m^*}} \right). \quad (29)$$

Figure 5 provides detailed results on how different choices of $N_\chi \times N_\xi$ and performing all the calculations in single or double precision affect the accuracy of the table. As for Breit–Wheeler pair production, increasing the number of table points reduces the error and calculating the tables and performing the interpolation in single or in double precision does not seem to affect the final error significantly. A resolution of 128 points in each dimension is required to reduce the error to the few percent level across the whole χ range considered here.

Using logarithmically spaced ξ points for the photon emission lookup table inevitably results in a reduced resolution for larger ξ (since the spacing between two consecutive points becomes larger). This corresponds to the high-energy tail of the energy spectra of the emitted photons. Therefore, using this strategy for the lookup table may result in photon energy distributions with relatively large errors at high energies, unless a very high number of points is used (see section 5). This may be an issue especially at very high χ parameters since a narrow peak at $\xi \sim 1$ may appear in the energy distribution of the emitted photons [5, 64, 65]. To address this issue, PICSAR-QED provides an alternative strategy for the spacing of the ξ axis: using logarithmically spaced points up to a given ξ^* (e.g. $\xi^* \sim 0.1$), and linearly spaced points up to $\xi = 1$. This feature is currently experimental (neither its performances nor the effect of varying ξ^* have been thoroughly assessed), but preliminary tests reported in section 5 show that it allows resolving the tail of the photon energy distribution with significantly fewer total table points.

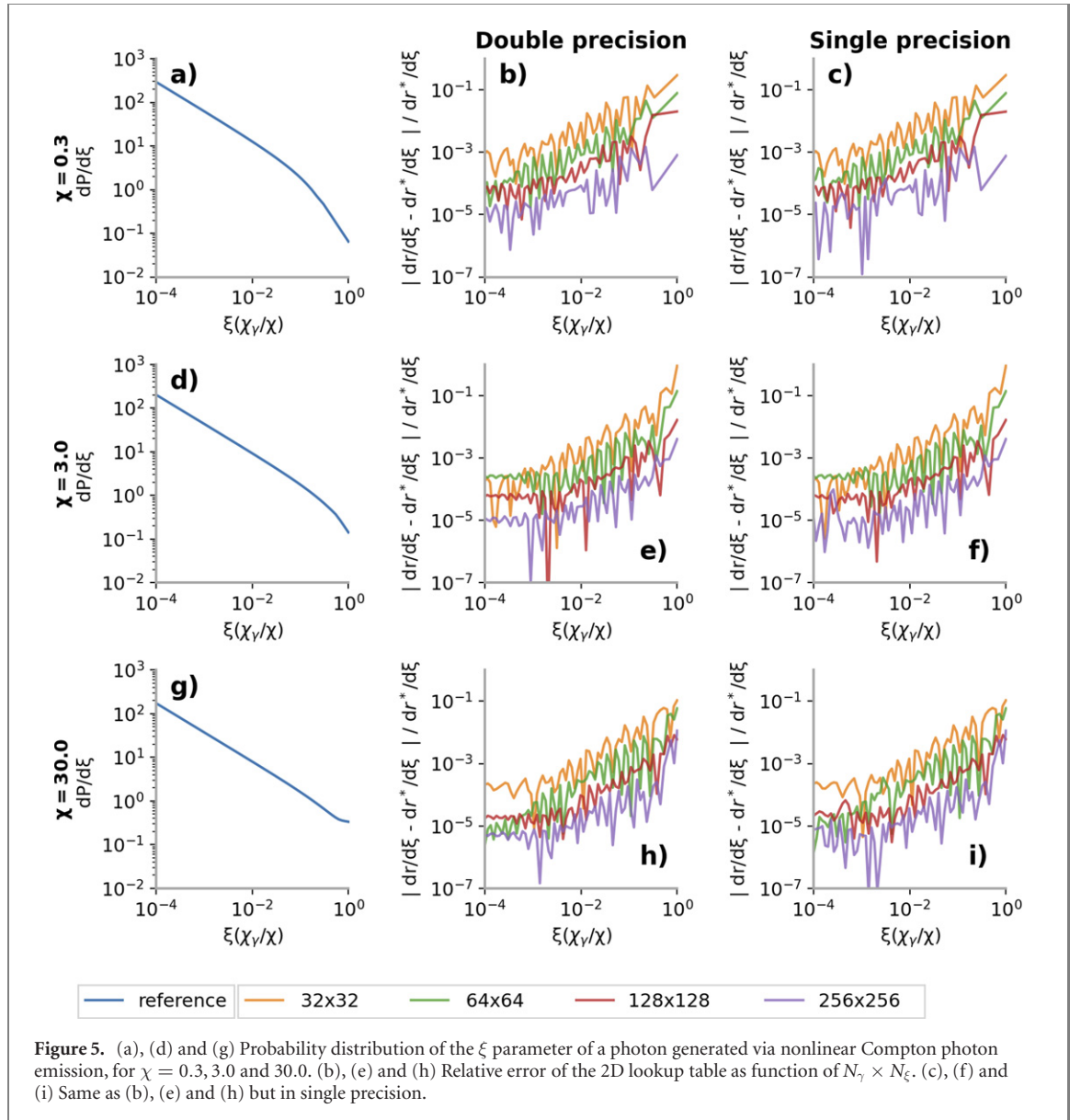
In all the benchmarks reported in this work, we have used the lookup table with the logarithmically spaced ξ points, except for the results reported in figure 7.

3.3. Portability across different architectures

From a technical point of view, PICSAR-QED is a C++14, *header-only* library, designed to integrate easily into other projects, and to provide methods able to run efficiently on different computing architectures. This section describes how these goals are achieved.

Support of different unit systems—Internal calculations in PICSAR-QED are performed adopting *Heaviside–Lorentz units* with 1 MeV chosen as the reference energy. However, the interface of PICSAR-QED also supports *SI units* and normalized units where the speed of light, the elementary charge, and the electron mass are equal to one and either a reference length or a reference frequency is used (in these cases the value of the reference quantity must be provided). In practice, since the library is written in C++, this is achieved via templates, in order to avoid code duplication. The choice of the units is performed at compilation time, in order to avoid overheads at runtime. Since lookup tables are adimensional, once generated they can be used with any choice of unit system.

Support for single and double precision—we provide simple and double precision versions of each method (in practice, since the library is written in C++ this is also achieved via templates). This flexibility



is crucial, especially for methods that should be used at runtime, since on several architectures running in single precision leads to large speedups. Moreover, it allows integrating PICSAR-QED with codes using single precision, avoiding expensive runtime floating point casts. Lookup tables can be computed either in single or double precision, with the former case being significantly faster due to relaxed tolerance required for numerical integration. PICSAR-QED also provides an option to compute the lookup tables in double precision and store them in single precision.

Avoid dependencies on specific pseudo-random number generators—PICSAR-QED is a Monte Carlo module. Therefore, pseudo-random numbers are needed at runtime. Since pseudo-random number generators have different interfaces in different libraries and performance portability frameworks, we decided not to force the use of a specific pseudo-random number generator, nor to include a pseudo-random number generator in PICSAR-QED. Our design specifications requires that a random number (uniformly distributed between zero and one) is passed to each runtime function requiring randomness. This gives complete freedom to the users on how to generate such random numbers.

Compatibility with different architectures—PICSAR-QED provides a collection of methods that can be divided into two categories: *runtime* methods (which are actually needed during a PIC simulation) and *lookup table generation methods*, which are needed only to generate lookup tables for later use. Only *runtime* functions need to be portable on different architectures, while methods for lookup-table generation need only to run on CPUs (moreover, their compilation for some architectures, namely GPUs, is not currently possible). In order to achieve portability across different architectures, all the *runtime* methods are pre-pended with some macros, whose values must be set appropriately to compile the code for CPUs or

Table 1. Performance benchmarks of the main functions provided by PICSAR-QED on different architectures with different paradigms. Each cell reports two numbers, which represent the time required to complete the test in milliseconds (each number is actually the average of three runs of the test). The top number refers to the case in double precision, while the bottom number refers to the case in single precision. For the Intel Xeon Gold 6152 and AMD EPYC 7302 cases, the number in parenthesis is the number of OpenMP threads used for the test. (+*fast math*) and (+*precise*), as documented in appendix C, refer to the use of floating point models not strictly IEEE-compliant in the compilation of the code, which normally allows for additional optimizations.

Hardware	Test case	Kernel timing (ms) $\begin{pmatrix} \text{double} \\ \text{float} \end{pmatrix}$			
		BW evol	BW prod	QS evol	QS em
NVIDIA Quadro GV100	CUDA	14.52	29.94	14.62	26.88
		7.43	15.70	7.39	14.15
NVIDIA Quadro GV100	Kokkos + CUDA	15.11	109.87	15.11	115.22
		7.91	81.44	8.96	79.87
2x Intel Xeon Gold 6152 (11)	OpenMP (+precise)	305.70	1354.27	323.09	1454.12
		122.51	948.82	135.89	973.01
2x Intel Xeon Gold 6152 (11)	Kokkos + OpenMP	1027.38	6622.17	1194.59	7078.74
		1176.31	6609.26	826.53	6925.94
2x Intel Xeon Gold 6152 (22)	OpenMP (+precise)	144.28	738.31	173.04	789.34
		69.70	533.57	68.94	544.17
2x Intel Xeon Gold 6152 (22)	Kokkos + OpenMP	531.24	4349.26	612.26	4582.44
		622.76	4389.06	436.75	4492.05
2x Intel Xeon Gold 6152 (44)	OpenMP (+precise)	125.17	452.58	130.38	478.98
		57.82	328.04	66.69	342.09
2x Intel Xeon Gold 6152 (44)	Kokkos + OpenMP	324.07	3500.50	375.32	3655.18
		369.82	3543.78	263.37	3606.69
2x Intel Xeon Gold 6152 (88)	OpenMP (+precise)	157.50	326.59	135.68	343.62
		62.36	243.98	70.14	254.66
2x Intel Xeon Gold 6152 (88)	Kokkos + OpenMP	205.15	1135.29	225.32	1201.02
		215.60	1124.45	166.59	1093.64
AMD EPYC 7302 (32)	OpenMP	338.77	945.99	401.09	931.46
		189.56	926.39	331.63	924.30
AMD EPYC 7302 (32)	OpenMP (+fast math)	349.69	663.94	357.94	634.82
		169.36	558.94	253.24	504.24
AMD EPYC 7302 (32)	Kokkos + OpenMP	349.17	5242.70	392.12	4967.67
		330.48	4699.89	248.85	4632.18

GPUs, or to use performance portability frameworks like Kokkos [66] and AMReX [67], as explained in detail in appendix A. While Kokkos is primarily designed as a performance portability framework, AMReX is actually a library designed to support massively parallel block-structured adaptive mesh refinement (AMR) applications, but it also offers features enabling performance portability of the applications built on top of it, like the PIC code WarpX.

Another key concept enabling portability concerns the data structures. In this regard, PICSAR-QED provides containers, such as those used internally for the lookup-tables, which must be initialized with methods running on CPUs, while at the same time being available in GPU kernels, if the library is compiled for those architectures. Achieving this may require some effort from the user, but the amount of effort is minimal for Kokkos and AMReX, as well as for programming models like CUDA [68], as shown in appendix B.

4. Performance benchmarks on different architectures

Since PICSAR-QED can be compiled for different architectures and integrated with different performance portability frameworks, we have carried out extensive performance benchmarks of the four most important kernels of the library:

- Breit–Wheeler optical depth evolution
- Breit–Wheeler pair production
- Nonlinear Compton optical depth evolution
- Nonlinear Compton photon emission.

Those benchmarks were carried out with a test program using CUDA on an NVIDIA Quadro GV100 GPU, with a test program using OpenMP on a dual-socket machine with Intel Xeon Gold 6152 CPUs and on an AMD EPYC 7302 CPU, and with a test program using the Kokkos library on all the aforementioned

architectures. In some selected cases, the effect of changing the number of threads and enabling non IEEE-compliant aggressive floating-point optimizations ('fast math') was tested as well. We also performed some initial benchmarks on the Fujitsu A64FX CPU, which demonstrate that PICSAR-QED can be used on this architecture, but the results are too preliminary to be included in a fair benchmark. In all cases the benchmark was carried out with 10^8 particles, each one having ten real components: the three components of the momentum, the six components of the electromagnetic field, and the optical depth. Those quantities are initialized randomly, drawing each component of \mathbf{E} in $[-E_s/100, E_s/100]$, each component of \mathbf{B} between $[-E_s/(100c), E_s/(100c)]$, each component of the momentum in $[-1 \text{ GeV}/c, 1 \text{ GeV}/c]$ and the optical depth from an exponential distribution. 1D lookup tables were generated with 256 points, while 2D lookup tables were generated with 256×256 points. χ_{\min} and $\chi_{\gamma, \min}$ were chosen to be, respectively, 0.001 and 0.02. χ_{\max} and $\chi_{\gamma, \max}$ were both 10^3 , while ξ_{\min} was 10^{-12} . Each kernel was tested in double precision and in single precision for all the 10^8 particles. Appendix C provides details on how the code was compiled in each case. Table 1 reports the results of these benchmarks. The results show remarkable performances with the NVIDIA GV100 GPU, and a near perfect halving of the time spent in each kernel passing from double precision to single precision (as for many architectures, with NVIDIA GV100 the theoretical maximum FLOPS in single precision is twice the maximum FLOPS attainable in double precision). On the other hand, we rarely observe such halving in the case of CPUs. However, this can be explained as follows. With CPUs, the performance doubling passing from double precision to single precision occurs normally only if a kernel can take full advantage of SIMD vectorization. For SIMD vectorization to occur, a kernel must respect relatively stringent requirements. Pair production and photon emission kernels are too complex to be fully vectorized, especially due to the branching conditions in the binary search step included in those kernels. Optical depth evolution kernels are significantly easier to vectorize, although, as shown in the table, this requires to relax floating point arithmetic (relaxing strict IEEE compliance for floating point operations allows most compilers to perform more extensive optimizations, such as reordering of commutative math operations and using approximate versions of some functions). We also observed that using Kokkos introduces an overhead, which is small (even negligible in some cases) for the optical depth evolution kernels. However, we observed a significant overhead in the case of the pair production kernel and of the photon emission kernel. Investigation of this issue is currently in progress. In any case, although the overhead for some kernels is substantial, in a complete PIC simulation this is not likely to be a significant issue. Indeed, while the optical evolution kernels are executed at each timestep for each particle, pair production and photon emission kernels are relevant only for the few of them undergoing a QED process in a given timestep (for the simulation to be reliable, the timestep must be chosen small enough for pair production and photon emission to occur significantly less than every timestep for each particle [23]).

A fairer comparison between different computing architectures requires to take into account their different power consumption (i.e. 250 W for NVIDIA Quadro GV100, 280 W for a pair of Intel Xeon Gold 6152, and 155 W for AMD EPYC 7302). Therefore, table 2 reports, for a selection of cases, the total energy required to apply each kernel to all the particles. We note that, without using Kokkos, performing a test on a NVIDIA Quadro GV100 GPU requires one order of magnitude less energy than performing the same test on CPU.

5. Integration of PICSAR-QED with WarpX PIC code

In order to integrate PICSAR-QED with WarpX, we followed standard procedures described in the literature [25]. We first added photon macro-particles. Then, for particles involved in QED processes, we added a new real component: the *optical depth*. We then modified the particle evolution routines in order to update this component for particles participating in QED processes. Only for particles whose optical depth reaches zero, we call either a routine to generate a pair, if the particle is a photon, or a routine to emit a photon, if the particle is an electron or a positron. In WarpX, all particle creation from the nonlinear Compton scattering, nonlinear Breit–Wheeler and Schwinger processes occurs between the Maxwell solver and the particle pusher. In order to avoid the generation of too many low-energy photon particles, WarpX allows setting a threshold for keeping such particles after creation. If the photon energy is below such threshold, the emitting particle is slowed down as if the photon were emitted, but the photon particle is discarded immediately after creation. By default, the threshold is set to $2m_e c^2$, because particles with lower energy will almost certainly not decay into a pair. Since simulating QED effects makes sense only when QED production rates are sufficiently high, WarpX offers the possibility to select a minimum quantum parameter threshold below which Breit–Wheeler pair production is ignored and photon emission is simulated with a classical model [69, 70]. The suggested values of such thresholds in WarpX are $\chi = 10^{-2}$ for Breit–Wheeler pair production (since below this threshold the pair production rate is extremely small), and $\chi_\gamma = 10^{-3}$ for Compton emission (following [24]).

Table 2. Performance benchmarks of the main functions provided by PICSAR-QED on different architectures with different paradigms. The table is based on a selection of the data reported in table 1. The time measured for each test case was multiplied by the *thermal design point* power of each processor, giving the energy required to complete each test. The top number refers to the case in double precision, while the bottom number refers to the case in single precision. For the Intel Xeon Gold 6152 and AMD EPYC 7302 cases, the number in parenthesis is the number of OpenMP threads used for the test. (+*fast math*) and (+*precise*), as documented in appendix C, refer to the use of floating point models not strictly IEEE-compliant in the compilation of the code, which normally allows for additional optimizations.

Hardware	Test case	Used energy per kernel (J) $\left(\begin{smallmatrix} \text{double} \\ \text{float} \end{smallmatrix}\right)$			
		BW evol	BW prod	QS evol	QS em
NVIDIA Quadro GV100	CUDA	3.63	7.49	3.66	6.72
		1.86	3.93	1.85	3.54
NVIDIA Quadro GV100	Kokkos + CUDA	3.78	27.47	3.78	28.81
		1.98	20.36	2.24	19.97
2x Intel Xeon Gold 6152 (88)	OpenMP (+precise)	44.1	91.45	37.99	96.21
		17.46	68.31	19.64	71.30
2x Intel Xeon Gold 6152 (88)	Kokkos + OpenMP	57.44	317.88	63.09	336.29
		60.37	314.85	46.65	306.22
AMD EPYC 7302 (32)	OpenMP (+fast math)	54.20	102.91	55.48	98.40
		26.25	86.64	39.25	78.16
AMD EPYC 7302 (32)	Kokkos + OpenMP	54.12	812.62	60.78	769.99
		51.22	728.48	38.57	717.99

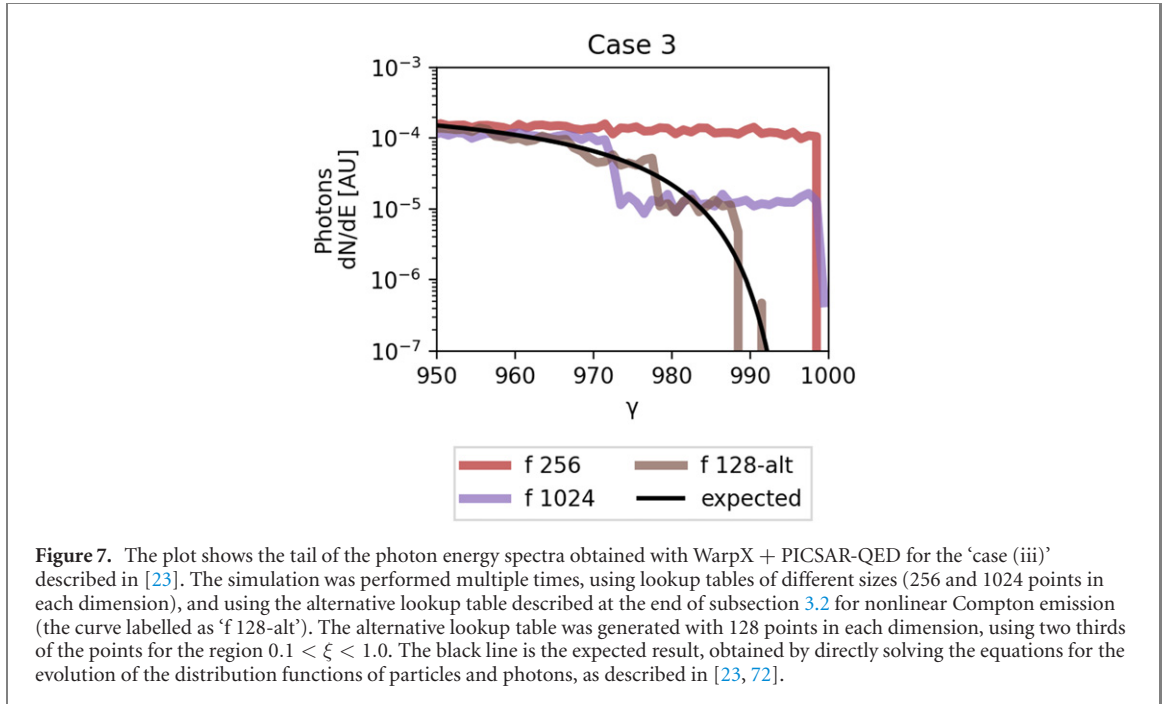
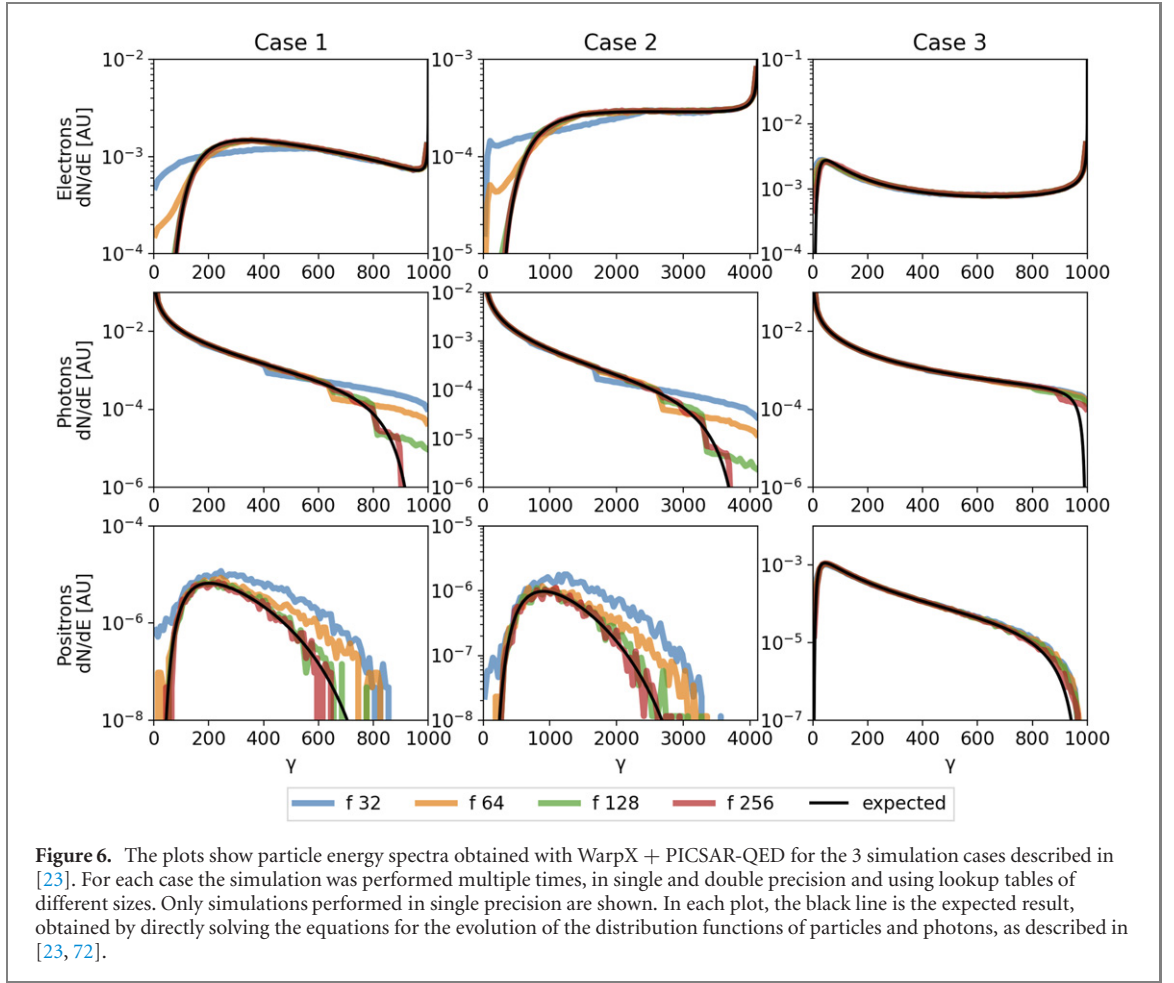
Schwinger pair creation is included in WarpX by repeating the following procedure at every timestep: first, all six components of the electromagnetic field are interpolated to same location in the grid in order to avoid numerical errors when computing the field invariants [71]. Then, the Schwinger pair production rate is computed in every cell using the PICSAR-QED routines. By multiplying this rate by the timestep and the volume of a cell (in 2D simulations, the size in the third dimension necessary to compute the volume of a cell is a user-defined parameter), we obtain the expected number of Schwinger pairs generated in a cell within a timestep. The actual number of pairs that is created is then drawn from a Poisson distribution (in order to avoid unnecessarily long computations, the Poisson distribution is replaced by a Gaussian distribution if the expected number of pairs is above a given threshold, which is 25 by default). If the obtained number of pairs is greater than zero, one macro-electron and one macro-positron are added to the simulation, with a weight corresponding to the number of physical particles created. The particles are initialized with zero momentum at the position in the cell where the electromagnetic field has been interpolated.

When WarpX is compiled for GPUs, all the routines dealing with QED processes (optical depth initialization and evolution, pair production and photon emission) are carried out in GPU kernels, with the exception of the initial lookup tables generation. Lookup tables generation typically requires a few minutes, and can benefit from multi-core parallelization (which can bring the time required to compute the tables down to few tens of seconds). The user can either generate tables by setting the parameters in WarpX input file or load a lookup table generated beforehand.

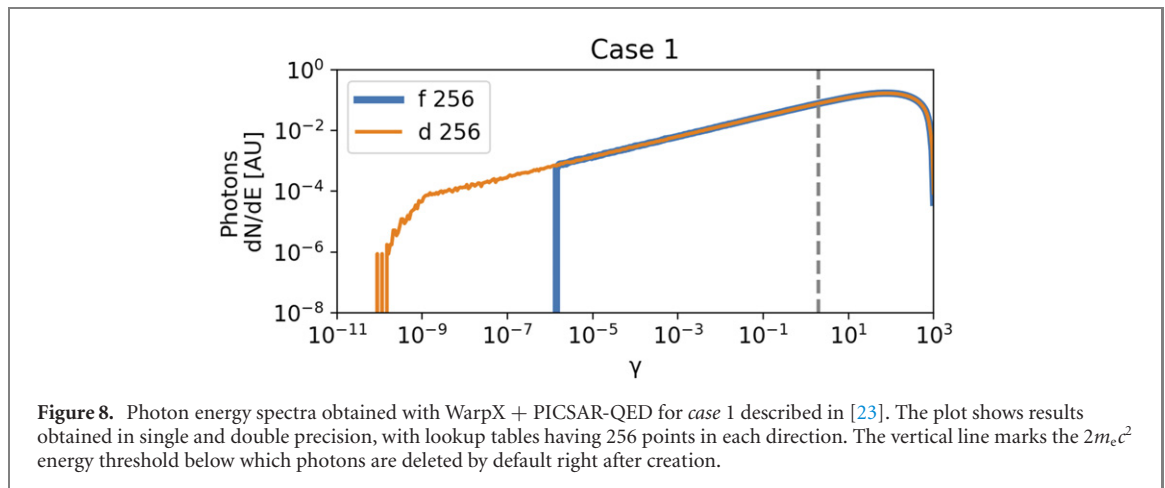
5.1. Benchmarks with existing literature

In order to test WarpX + PICSAR-QED, we reproduced the three test cases mentioned in [23]. In this paper by Ridgers *et al*, the authors describe their implementation of a QED module to simulate QED processes in laser–plasma interaction. They test this module (which is built-in in the EPOCH PIC code) in the following three physical scenarios:

- An electron population with initial Lorentz factor $\gamma_0 = 1000$ propagating in a perpendicular, static, magnetic field $B_0 = 10^{-3}E_s/c$. This case corresponds to an initial quantum parameter $\chi = 1$. The duration of the simulation is 1 fs.
- An electron population with initial Lorentz factor $\gamma_0 = 4120$ counter-propagating with a circularly polarized plane wave with $E_0 = 1.22 \times 10^{-4}E_s$. This case corresponds to an initial quantum parameter $\chi = 1$. The duration of the simulation is 3 fs.
- An electron population with initial Lorentz factor $\gamma_0 = 1000$ propagating in a perpendicular, static, magnetic field $B_0 = 9 \times 10^{-3}E_s/c$. This case corresponds to an initial quantum parameter $\chi = 9$. The duration of the simulation is 0.1 fs.



We simulated these three cases with WarpX + PICSAR-QED, using the code in single precision and with lookup tables resolutions ranging from 32 points to 256 points in each dimension. Figure 6 reports the energy spectra of the particles at the end of the simulation, for the three cases and for the 4 lookup table resolutions tested in the benchmark. Provided that the resolution of the lookup tables is sufficiently high, our results closely reproduce those published in the paper by Ridgers *et al*, which confirms the correctness



of the QED modules in WarpX. However, it is worth remarking that the tail of the photon energy distribution is not perfectly reproduced, even with lookup tables having 256 points in each dimension. This is a direct consequence of the choice of a logarithmically spaced lookup table, as explained in subsection 3.2. To better simulate the tail of the photon energy spectra, a strategy could be to increase the number of table points. This is exemplified in figure 7, which highlights the tail of the photon energy distribution for the third simulation case. A new curve obtained with lookup tables having 1024 points is closer to the expected one than the curve obtained with lookup tables having 256 points, although this convergence towards the right solution seems to be quite slow. A more efficient strategy to improve the modeling of the tail of the distribution is to use the alternative table point spacing proposed at the end of subsection 3.2, which is still experimental in PICSAR-QED (and, as a consequence, in WarpX). As shown in figure 7, this lookup table strategy allows for a very good agreement with the expected curve using only 128 table points.

Concerning the results reported in figure 6, we repeated the tests also in double precision, observing only minor differences. Running the code in single precision (and therefore using lookup tables calculated in single precision), however, has an effect on the photon energy spectrum at very low energy, as shown in figure 8. The spectra are very similar, but in single precision the low energy part of the spectrum is truncated. This is simply due to the smaller range of real numbers that single precision can represent, and is a minor concern, since photons with such a low energy have a negligible impact on the total radiative losses. Moreover, those photons are orders of magnitude below $2m_e c^2$, which means that they cannot decay into electron–positron pairs via Breit–Wheeler pair production.

6. Conclusions

This paper presents PICSAR-QED, a Monte Carlo module to simulate strong-field QED processes in PIC codes designed for next-generation supercomputers. PICSAR-QED is conceived to be easily included in other processes and to be portable to different architectures. The outcome of several benchmarks, which demonstrate that the library produces reliable results and can run efficiently on CPUs and GPUs, is reported. Moreover, a detailed investigation on how the parameters used to generate the lookup tables affect the simulation results is provided. The main outcomes of this investigation are that (i) performing the calculations of the QED routines in single precision does not affect the accuracy of the results, and (ii) lookup tables with 128 or 256 points in each direction are sufficient to ensure small interpolation errors. Finally, the integration of PICSAR-QED with a state-of-the-art PIC code, WarpX, and the validation of such integration are discussed. WarpX + PICSAR-QED has been already used for production simulations of laser–plasma interaction at QED relevant intensities [73].

Acknowledgments

This research used the open-source particle-in-cell code WarpX <https://github.com/ECP-WarpX/WarpX>, primarily funded by the US DOE Exascale Computing Project. Primary WarpX contributors are with LBNL, LLNL, CEA-LIDYL, SLAC, DESY, CERN, and Modern Electron. We acknowledge all WarpX contributors. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National

Laboratory, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC05-00OR22725. We acknowledge the support of the US DOE Exascale Computing Project and of the Director, Office of Science, Office of High Energy Physics, of the US Department of Energy under Contract No. DEAC02-05CH11231. This work was supported by the French National Research Agency (ANR) T-ERC program (Grant No. ANR-18-ERC2-0002). We also acknowledge the financial support of the Cross-Disciplinary Program on Numerical Simulation of CEA, the French Alternative Energies and Atomic Energy Commission. This project has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 871072. The authors would like to thank Dr Mathieu Lobet (Maison de la Simulation, CEA-Saclay, France) for useful discussions concerning the implementation of QED effects in PIC codes.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Appendix A. Portability across different architectures

In PICSAR-QED, all the runtime methods are pre-pended with two macros, `PXRMP_GPU_QUALIFIER` and `PXRMP_FORCE_INLINE`, as exemplified here for the function calculating the Schwinger pair production rate:

```
template<
    typename RealType,
    unit_system UnitSystem = unit_system::SI
>
PXRMP_GPU_QUALIFIER
PXRMP_FORCE_INLINE
RealType pair_production_rate(
    const RealType ex, const RealType ey, const RealType ez,
    const RealType bx, const RealType by, const RealType bz,
    const RealType ref_quantity = math::one<RealType>);
```

By setting those two macros according to table 3, runtime functions can be either:

- Compiled directly for CPUs.
- Compiled directly for NVIDIA GPUs using CUDA.
- Made compatible with the Kokkos performance portability library (and then compiled for CPUs or GPUs).
- Made compatible with the AMReX library (and then compiled for CPUs or GPUs).

Appendix B. Details on the data structures used in PICSAR-QED

This appendix provides some details on the underlying data structures used for the lookup tables by PICSAR-QED. Lookup tables use several 1D vectors to store table data (even 2D lookup tables use 1D vectors as their underlying data structure). Those vectors must be manipulated on CPU when the lookup table is generated or loaded from disk, but they must be accessible within GPU kernels if PICSAR-QED is compiled for GPU architectures. Moreover, a machinery to seamlessly pass data between the CPU and the GPU is needed. AMReX already provides such a vector type: `amrex::Gpu::DeviceVector` (with the additional prescription of calling a GPU synchronization method after table initialization). In pure CUDA, a very thin wrapper around the `device_vector` provided by CUDA thrust library is enough to fulfill

Table 3. Reference table on how to set `PXRMP_GPU_QUALIFIER` and `PXRMP_FORCE_INLINE` to compile PICSAR-QED methods for CPUs and GPUs and/or to make them compatible with AMReX or Kokkos.

Case	PXRMP_GPU_QUALIFIER	PXRMP_FORCE_INLINE
Serial or openMP (CPU)	—	Compiler-dependent ^a
CUDA	<code>__host__ __device__</code>	<code>__forceinline__</code>
Kokkos	—	<code>KOKKOS_FORCEINLINE_FUNCTION</code>
AMReX	<code>AMREX_GPU_HOST_DEVICE</code>	<code>AMREX_FORCE_INLINE</code>

^aFor most compilers this must be `inline __attribute__((always_inline))`, which is the default value in PICSAR-QED.

these requirements:

```
template<typename RealType>
class ThrustDeviceWrapper : public thrust::device_vector<RealType>
{
public:
    template<typename... Args>
    ThrustDeviceWrapper(Args&&... args) :
        thrust::device_vector<RealType>(std::forward<Args>(args)...) {}

    const RealType* data() const
    {
        return thrust::raw_pointer_cast(
            thrust::device_vector<RealType>::data());
    }
};
```

Similarly, for Kokkos, a thin wrapper around `Kokkos::vector<Real>` can be used:

```
template <typename Real>
class KokkosVectorWrapper : public Kokkos::vector<Real>
{
    using KV = Kokkos::vector<Real>;

public:

    template<typename... Args>
    KokkosVectorWrapper(Args&&... args) : KV(std::forward<Args>(args)...) {}

    void pxr_sync()
    {
        Kokkos::deep_copy(KV::d_view, KV::h_view);
    }

    const Real* data() const
    {
        return KV::d_view.data();
    }
};
```

PICSAR-QED calls the method `pxr_sync()` whenever needed, if it detects that a type having such a method is used as the underlying vector type.

On CPUs, lookup tables can be passed directly as a constant reference to *runtime* functions. On GPUs, an additional step is needed: each table can export a ‘table view’, which is a lookup table of the same kind

Table 4. Details on the configurations used for the performance benchmarks.

NVIDIA Quadro GV100: CUDA	
GPU details:	NVIDIA Quadro GV100, 32GB RAM,
Compilers:	Driver Version: 450.51.05, CUDA version: 11.0
Optimizations:	g++ (GCC) 7.3.1 (host), nvcc v11.0.194 -gpu-architecture = sm_70 -O3 -use_fast_math
NVIDIA Quadro GV100: Kokkos + CUDA	
GPU details:	(See above)
Compilers:	g++ (GCC) 7.3.1 (host), nvcc v11.0.194
Kokkos version:	Kokkos tag: 3.3.01
Optimizations:	-O3, Kokkos_ARCH_VOLTA70 = ON, Kokkos_ENABLE_CUDA = ON, Kokkos_ENABLE_CUDA_CONSTEXPR = ON, Kokkos_ENABLE_CUDA_LAMBDA = ON, Kokkos_ENABLE_AGGRESSIVE_VECTORIZATION
2x Intel Xeon Gold 6152: OpenMP (+precise)	
Compilers:	Intel(R) oneAPI DPC++/C++ Compiler 2021.3.0
Optimizations:	-O3 -fopenmp -march = native -mtune = native
OpenMP options:	-fp-model = precise OMP_PROC_BIND = spread
2x Intel Xeon Gold 6152: Kokkos + OpenMP	
Compilers:	g++ (Spack GCC) 11.1.0 (with Graphite)
Kokkos version:	Kokkos tag: 3.3.01
Optimizations:	-O3, Kokkos_ARCH_SKX = ON, Kokkos_ENABLE_OPENMP = ON, Kokkos_ENABLE_AGGRESSIVE_VECTORIZATION
OpenMP options:	OMP_PROC_BIND = spread
AMD EPYC 7302: OpenMP (with or without -fast-math)	
Compilers:	AMD clang version 12.0.0
Optimizations:	-O3 (-Ofast -ffast-math) -fopenmp -march = znver3
OpenMP options:	OMP_PROC_BIND = spread
Notes:	Lookup tables generation not compiled with fast-math
AMD EPYC 7302: Kokkos + OpenMP	
Compiler versions:	AMD clang version 12.0.0
Kokkos version:	Kokkos tag: 3.3.01
Optimizations:	-O3 -march = znver2 -mtune = znver2, Kokkos_ENABLE_OPENMP = ON, Kokkos_ENABLE_AGGRESSIVE_VECTORIZATION
OpenMP options:	OMP_PROC_BIND = spread

internally using non-owning pointers to the raw table data. This ‘table view’ can be safely passed by copy to GPU kernels. This more complex procedure also works on CPUs. Therefore, if libraries such as AMReX or Kokkos are used, the same code can be compiled for both GPUs and CPUs.

Appendix C. Technical details on PICSAR-QED benchmarks









Table 4 reports several details (compiler versions, compilation options ...) on the configurations used for the benchmarks discussed in section 4. To ensure reproducibility of our results, we also provide the *git commit number* corresponding to the version of the software that we have used in this work:

- PICSAR-QED [43]: c16b642e3dcf860480dd1dd21cefa3874f395773
- WarpX [74]: b83f2949a1ac2eed003e991e9653b8427716bf14
- AMReX [75]: b15b1cf8d282cbb2c0d0bc0c7b049a79375ea63c

The code used for the performance benchmarks will be made available on the repository of PICSAR-QED.

We note that the very few tests performed with the alternative lookup table strategy described at the end of subsection 3.2 were carried out with a version of the code which, at time of writing, is not yet available in the main PICSAR-QED and WarpX repositories.

ORCID iDs

Luca Fedeli  <https://orcid.org/0000-0002-7215-4178>
 Neil Zaïm  <https://orcid.org/0000-0002-9582-5894>
 Antonin Sainte-Marie  <https://orcid.org/0000-0003-4003-6246>
 Maxence Thévenet  <https://orcid.org/0000-0001-7216-2277>
 Axel Huebl  <https://orcid.org/0000-0003-1943-7141>
 Andrew Myers  <https://orcid.org/0000-0001-8427-8330>
 Jean-Luc Vay  <https://orcid.org/0000-0002-0040-799X>
 Henri Vincenti  <https://orcid.org/0000-0002-9839-2692>

References

- [1] Marklund M and Shukla P K 2006 Nonlinear collective effects in photon–photon and photon–plasma interactions *Rev. Mod. Phys.* **78** 591–640
- [2] Bell A R and Kirk J G 2008 Possibility of prolific pair production with high-power lasers *Phys. Rev. Lett.* **101** 200403
- [3] Ridgers C P, Brady C S, Ducloux R, Kirk J G, Bennett K, Arber T D, Robinson A P L and Bell A R 2012 Dense electron–positron plasmas and ultraintense γ rays from laser-irradiated solids *Phys. Rev. Lett.* **108** 165006
- [4] Di Piazza A, Müller C, Hatsagortsyan K Z and Keitel C H 2012 Extremely high-intensity laser interactions with fundamental quantum systems *Rev. Mod. Phys.* **84** 1177–228
- [5] Bulanov S S, Schroeder C B, Esarey E and Leemans W P 2013 Electromagnetic cascade in high-energy electron, positron, and photon interactions with intense laser pulses *Phys. Rev. A* **87** 062110
- [6] Poder K et al 2018 Experimental signatures of the quantum nature of radiation reaction in the field of an ultraintense laser *Phys. Rev. X* **8** 031004
- [7] Cole J M et al 2018 Experimental evidence of radiation reaction in the collision of a high-intensity laser pulse with a laser-wakefield accelerated electron beam *Phys. Rev. X* **8** 011020
- [8] Bucksbaum P H et al 2020 Probing QED cascades and pair plasmas in Laboratory Experiments LoI to cosmic Frontier, Snowmass 2021, Letter of Intent <https://snowmass21.org/docs/files/summaries/CF/SNOWMASS21-CF1-001.pdf>
- [9] Zhang P, Bulanov S S, Seipt D, Arefiev A V and Thomas A G R 2020 Relativistic plasma physics in supercritical fields *Phys. Plasmas* **27** 050601
- [10] Gonoskov A, Blackburn T G, Marklund M and Bulanov S S 2021 Charged particle motion and radiation in strong electromagnetic fields (arXiv:2107.02161 [physics.plasm-ph])
- [11] Vincenti H 2019 Achieving extreme light intensities using optically curved relativistic plasma mirrors *Phys. Rev. Lett.* **123** 105001
- [12] Michel F C 1982 Theory of pulsar magnetospheres *Rev. Mod. Phys.* **54** 1–66
- [13] Blandford R D and Znajek R L 1977 Electromagnetic extraction of energy from Kerr black holes *Mon. Not. R. Astron. Soc.* **179** 433–56
- [14] Ruffini R, Vereshchagin G and Xue S-S 2010 Electron–positron pairs in physics and astrophysics: from heavy nuclei to black holes *Phys. Rep.* **487** 1–140
- [15] Harding A K and Lai D 2006 Physics of strongly magnetized neutron stars *Rep. Prog. Phys.* **69** 2631–708
- [16] Uzdensky D A and Rightley S 2014 Plasma physics of extreme astrophysical environments *Rep. Prog. Phys.* **77** 036902
- [17] Philippov A, Timokhin A and Spitkovsky A 2020 Origin of pulsar radio emission *Phys. Rev. Lett.* **124** 245101
- [18] Mészáros P, Ramirez-Ruiz E and Rees M J 2001 e^+ pair cascades and precursors in gamma-ray bursts *Astrophys. J.* **554** 660
- [19] Birdsall C K and Langdon A B 1991 *Plasma Physics Via Computer Simulation* 1st Edition (Boca Raton: CRC Press) <https://doi.org/10.1201/9781315275048>
- [20] Hockney R W and Eastwood J W 1988 *Computer Simulation Using Particles* (Bristol: Hilger)
- [21] Arber T D et al 2015 Contemporary particle-in-cell approach to laser–plasma modelling *Plasma Phys. Control. Fusion* **57** 113001
- [22] Ducloux R, Kirk J G and Bell A R 2010 Monte Carlo calculations of pair production in high-intensity laser–plasma interactions *Plasma Phys. Control. Fusion* **53** 015009
- [23] Ridgers C P, Kirk J G, Ducloux R, Blackburn T G, Brady C S, Bennett K, Arber T D and Bell A R 2014 Modelling gamma-ray photon emission and pair production in high-intensity laser–matter interactions *J. Comput. Phys.* **260** 273–85
- [24] Lobet M 2015 Effets radiatifs et d'électrodynamique quantique dans l'interaction laser-matière ultra-relativiste *PhD Dissertation* Université de Bordeaux
- [25] Gonoskov A et al 2015 Extended particle-in-cell schemes for physics in ultrastrong laser fields: review and developments *Phys. Rev. E* **92** 023305
- [26] Lobet M, d'Humières E, Grech M, Ruyer C, Davoine X and Gremillet L 2016 Modeling of radiative and quantum electrodynamics effects in PIC simulations of ultra-relativistic laser–plasma interaction *J. Phys.: Conf. Ser.* **688** 012058
- [27] Nikishov A and Ritus V 1964 Quantum processes in the field of a plane electromagnetic wave and in a constant field: I. *Sov. Phys. - JETP* **19** 529–41
- [28] Erber T 1966 High-energy electromagnetic conversion processes in intense magnetic fields *Rev. Mod. Phys.* **38** 626–59
- [29] Baier V and Katkov V 1968 Processes involved in the motion of high energy particles in a magnetic field *Sov. Phys. - JETP* **26** 854
- [30] Ritus V 1985 Quantum effects of the interaction of elementary particles with an intense electromagnetic field *J. Sov. Laser Res.* **6** 497–617
- [31] Kirk J G, Bell A R and Arka I 2009 Pair production in counter-propagating laser beams *Plasma Phys. Control. Fusion* **51** 085008
- [32] Heldens S, Hijma P, Van Werkhoven B, Maassen J, Belloum A S Z and Van Nieuwpoort R V 2020 The landscape of exascale research: a data-driven literature analysis *ACM Comput. Surv.* **53** 1–43
- [33] Vazhkudai S S et al 2018 The design, deployment, and evaluation of the CORAL pre-exascale systems SC18: *Int. Conf. for High Performance Computing, Networking, Storage and Analysis* pp 661–72
- [34] Sato M et al 2020 *Co-Design for A64FX Manycore Processor and 'Fugaku' Ser. SC '20* (Atlanta: IEEE Press)
- [35] Fonseca R A et al 2002 Osiris: a three-dimensional, fully relativistic particle in cell code for modeling plasma based accelerators *Int. Conf. on Computational Science* (Berlin: Springer) pp 342–51

- [36] Bastrakov S, Donchenko R, Gonoskov A, Efimenko E, Malyshev A, Meyerov I and Surmin I 2012 Particle-in-cell plasma simulation on heterogeneous cluster systems *J. Comput. Sci.* **3** 474–9
- [37] Bird R, Tan N, Luedtke S V, Harrell S, Taufer M and Albright B 2021 VPIC 2.0: next generation particle-in-cell simulations *IEEE Trans. Parallel Distrib. Syst.* **33** 952–63
- [38] Bussmann M et al 2013 Radiative signatures of the relativistic Kelvin–Helmholtz instability *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Ser. SC '13* (Denver, Colorado) (New York: ACM) 5:pp 1:12–5
- [39] Derouillat J et al 2018 Smilei: a collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation *Comput. Phys. Commun.* **222** 351–73
- [40] Spitkovsky A 2005 Simulations of relativistic collisionless shocks: shock structure and particle acceleration *AIP Conf. Proc.* **801** 345–50
- [41] Lifschitz A F, Davoine X, Lefebvre E, Faure J, Rechatin C and Malka V 2009 Particle-in-cell modelling of laser–plasma interaction using Fourier decomposition *J. Comput. Phys.* **228** 803–1814
- [42] Martinez B, Lobet M, Duclous R, d’Humières E and Gremillet L 2019 High-energy radiation and pair production by Coulomb processes in particle-in-cell simulations *Phys. Plasmas* **26** 103109
- [43] PICSAR collaboration 2021 PICSAR-QED Github Repository <https://github.com/ECP-WarpX/picsar>
- [44] Vincenti H, Lobet M, Lehe R, Vay J-L and Deslippe J 2017 Pic codes on the road to exascale architectures *Exascale Scientific Applications* ed T Straatsma, K B Antypas and T Williams (Boca Raton: CRC Press) Ch 7
- [45] Vincenti H and Vay J-L 2016 Detailed analysis of the effects of stencil spatial variations with arbitrary high-order finite-difference Maxwell solver *Comput. Phys. Commun.* **200** 147–67
- [46] Vincenti H and Vay J-L 2018 Ultrahigh-order Maxwell solver with extreme scalability for electromagnetic PIC simulations of plasmas *Comput. Phys. Commun.* **228** 22–9
- [47] Blaclard G, Vincenti H, Lehe R and Vay J L 2017 Pseudospectral Maxwell solvers for an accurate modeling of Doppler harmonic generation on plasma mirrors with particle-in-cell codes *Phys. Rev. E* **96** 033305
- [48] Vay J-L et al 2018 Warp-X: a new exascale computing platform for beam-plasma simulations *Nucl. Instrum. Methods Phys. Res. A* **909** 476–9
- [49] Myers A et al 2021 Porting WarpX to GPU-accelerated platforms *Parallel Comput.* **108** 102 833
- [50] Messina P 2017 The exascale computing project *Comput. Sci. Eng.* **19** 63–7
- [51] Sauter F 1931 Über das verhalten eines elektrons im homogenen elektrischen feld nach der relativistischen theorie diracs *Z. Phys.* **69** 742–64
- [52] Heisenberg W and Euler H 1936 Folgerungen aus der diracschen theorie des positrons *Z. Phys.* **98** 714–32
- [53] Schwinger J 1951 On gauge invariance and vacuum polarization *Phys. Rev.* **82** 664–79
- [54] Yoon J W, Kim Y G, Choi I W, Sung J H, Lee H W, Lee S K and Nam C H 2021 Realization of laser intensity over 1023 W cm^{-2} *Optica* **8** 630–5
- [55] Niel F, Riconda C, Amiranoff F, Duclous R and Grech M 2018 From quantum to classical modeling of radiation reaction: a focus on stochasticity effects *Phys. Rev. E* **97** 043209
- [56] Di Piazza A, Tamburini M, Meuren S and Keitel C H 2018 Implementing nonlinear Compton scattering beyond the local-constant-field approximation *Phys. Rev. A* **98** 012134
- [57] Di A 2014 Piazza ultrarelativistic electron states in a general background electromagnetic field *Phys. Rev. Lett.* **113** 040402
- [58] Elkina N V et al 2011 QED cascades induced by circularly polarized laser fields *Phys. Rev. Spec. Top.–Accel. Beams* **14** 054401
- [59] Baumann C, Nerush E, Pukhov A and Kostyukov I Y 2019 Probing nonperturbative QED with electron–laser collisions *Sci. Rep.* **9** 1–8
- [60] Di Piazza A, Wistisen T N, Tamburini M and Uggerhoj U I 2020 Testing strong field QED close to the fully nonperturbative regime using aligned crystals *Phys. Rev. Lett.* **124** 044801
- [61] Narozhny N B, Bulanov S S, Mur V D and Popov V S 2004 $e^+ e^-$ -pair production by a focused laser pulse in vacuum *Phys. Lett. A* **330** 1–6
- [62] Mori M 1985 Quadrature formulas obtained by variable transformation and the DE-rule *J. Comput. Appl. Math.* **12–13** 119–30
- [63] Bailey D H, Jayabalan K and Li X S 2005 A comparison of three high-precision quadrature schemes *Exp. Math.* **14** 17–329
- [64] Seipt D and King B 2020 Spin- and polarization-dependent locally-constant-field approximation rates for nonlinear Compton and Breit–Wheeler processes *Phys. Rev. A* **102** 052805
- [65] Tamburini M and Meuren S 2021 Efficient high-energy photon production in the supercritical QED regime *Phys. Rev. D* **104** L091903
- [66] Edwards H C, Trott C R and Sunderland D 2014 Kokkos: enabling manycore performance portability through polymorphic memory access patterns *J. Parallel Distrib. Comput.* **74** 3202–16
- [67] Zhang W et al 2019 AMReX: a framework for block-structured adaptive mesh refinement *J. Open Source Softw.* **4** 1370
- [68] Nickolls J, Buck I, Garland M and Skadron K 2008 Scalable parallel programming with CUDA *Queue* **6** 40–53
- [69] Tamburini M, Pegoraro F, Piazza A D, Keitel C H and Macchi A 2010 Radiation reaction effects on radiation pressure acceleration *New J. Phys.* **12** 123005
- [70] Vranic M, Martins J L, Fonseca R A and Silva L O 2016 Classical radiation reaction in particle-in-cell simulations *Comput. Phys. Commun.* **204** 141–51
- [71] Zoni E et al 2021 A hybrid nodal-staggered pseudo-spectral electromagnetic particle in-cell method with finite-order centering (arXiv:2106.12919)
- [72] Sokolov I V, Naumova N M, Nees J A and Mourou G A 2010 Pair creation in QED-strong pulsed laser fields interacting with electron beams *Phys. Rev. Lett.* **105** 195005
- [73] Fedeli L, Sainte-Marie A, Zaim N, Thévenet M, Vay J L, Myers A, Quéré F and Vincenti H 2021 Probing strong-field QED with Doppler-boosted petawatt-class lasers *Phys. Rev. Lett.* **127** 114801
- [74] WarpX collaboration 2021 WarpX Github Repository <https://github.com/ECP-WarpX/WarpX>
- [75] AMReX collaboration 2021 AMReX Github Repository <https://github.com/AMReX-Codes/amrex>