Simulating the Nonlinear QED Vacuum

Andreas Maximilian Lindner

München 2023

Simulating the Nonlinear QED Vacuum

Andreas Maximilian Lindner

Dissertation
an der Fakultät für Physik
der Ludwig-Maximilians-Universität
München

vorgelegt von
Andreas Maximilian Lindner
aus Gräfelfing

München, den 05.05.2023

Erstgutachter: Prof. Dr. Hartmut Ruhl
Zweitgutachter: Prof. Dr. Holger Gies
Tag der mündlichen Prüfung: 07.07.2023

Zusammenfassung

Aufgrund von Vakuumfluktuationen, einer Kernvorhersage der Quantentheorie, sind virtuelle Elektron–Positron Paare omnipräsent im Vakuum der Quantenelektrodynamik (QED) und können Photon–Photon Wechselwirkungen vermitteln. Daher verhält sich das Vakuum der QED als ein nicht-lineares, polarisierbares Medium mit entsprechend quantifizierbaren Effekten.
Die zugehörige, niederenergetische Theorie für die Quantenelektrodynamik ist durch die bekannte effektive Wirkung nach Heisenberg und Euler gegeben. Diese erweitert in einer Schwachfeldentwicklung die Theorie des Elektromagnetismus von Maxwell mit effektiven Photon–Photon Interaktionen.
Rein optische Effekte infolge von Photon–Photon Streuung können durch hochintensive Laser im Vakuum ausgelöst werden und dabei faszinierende Phänomene auftreten lassen. Da Experimente bei ausreichend hohen Intensitäten bislang noch nicht existieren, sind diese Phänomene noch nicht nachgewiesen worden. Die theoretischen Vorhersagen auf diesem Gebiet sollen durch Computersimulationen unterstützt werden.
Ein äußerst akkurater, numerischer Solver für die führenden Ordnungen der Schwachfeldentwicklung nach Heisenberg und Euler wird in dieser Arbeit präsentiert. Eine Implementierung des Solvers in *C++* ist als Teil des Projekts frei verfügbar veröffentlicht. Der Name der Software lautet *HEWES*, welcher für „Heisenberg–Euler Weak-Field Expansion Simulator“ steht.
Der Solver wird in einer Raumdimension anhand von verfügbaren analytischen Ergebnissen zur Vakuumdoppelbrechung und der Erzeugung höherer Harmonischer validiert. Die Möglichkeiten, über analytisch lösbare Szenarien hinauszugehen, wird anhand von komplizierteren Rahmenbedingungen in höheren Dimensionen demonstriert.
Der zugrunde liegende Algorithmus besitzt eine nahezu lineare Vakuumdispersionsrelation, selbst für vergleichsweise kleine Wellenlängen. Des Weiteren gestaltet die Dispersionsrelation das numerische Schema stabil und Aliasing-frei. Die Genauigkeit der vorgestellten Implementierung ist dabei außergewöhnlich hoch, sowohl bezüglich der Ordnung des numerischen Schemas (dreizehnte Ordnung), als auch bezüglich der Ordnung der effektiven Entwicklung (bis zu sechs Photonen). Die Leistung des Solvers wird unter verschiedenen Konfigurationen evaluiert.

Da im Endeffekt realistische Simulationen auf hochauflösenden Gittern verlangt werden, wird besonderes Augenmerk auf die Skalierbarkeit und eine effiziente Implementierung der Software gelegt. Der Code ist derart gestaltet, dass er auf stark verteilten Systemen laufen kann und somit in der Lage ist, großangelegte Simulationen in allen drei Raumdimensionen und der Zeit durchzuführen.

Abstract

Due to vacuum fluctuations, a key prediction of quantum theory, virtual electron–positron pairs are omnipresent in the vacuum of quantum electrodynamics (QED) and can mediate photon–photon interactions as a consequence. Therefore, the vacuum of QED acts as a nonlinear, polarizable medium with quantifiable effects.

The corresponding low-energy theory for quantum electrodynamics is provided by the famous Heisenberg–Euler effective action. The latter, in a weak-field expansion, supplements Maxwell's theory of electromagnetism with effective light–light interactions.

All-optical effects due to photon–photon scattering can be triggered by high-intensity laser fields in the vacuum, causing a variety of intriguing phenomena. Due to the lack of sensitive experiments at sufficiently high intensity, these phenomena have not been accessible thus far. Theoretical predictions should be supported by computer simulations.

A highly accurate numerical solver for the leading orders of the weak-field expansion due to Heisenberg and Euler is presented in this work. A *C++* implementation of the solver is published open source as part of the project under the name *HEWES* which stands for "Heisenberg–Euler Weak-Field Expansion Simulator".

The solver is cross-checked in one spatial dimension against a set of already known analytical results of quantum vacuum effects such as birefringence and harmonic generation. Its power to go beyond analytically solvable scenarios is demonstrated in more complicated configurations in higher dimensions.

The underlying algorithm possesses an almost linear vacuum dispersion relation even for comparably small wavelengths. The latter further renders the scheme numerically stable and aliasing-free. The accuracy of the presented implementation is outstandingly high both in the numerical scheme (order thirteen) and the effective expansion (up to six-photon interactions). A performance evaluation for different settings is conducted.

Due to the need for realistic simulations on high-resolution lattices, special emphasis is put on the scalability and efficient implementation of the code. The code is designed to run on massively distributed computing systems and therewith is capable of conducting costly simulations in full three spatial dimensions plus time.

Outline

The thesis commences with an introduction to the fundamentals of the quantum vacuum. Subsequently, the simulation project is outlined, detailing the algorithm, the software, and a number of conducted simulations. Afterwards, aspects of the performance and accuracy are discussed, before lastly future prospects are set forth.

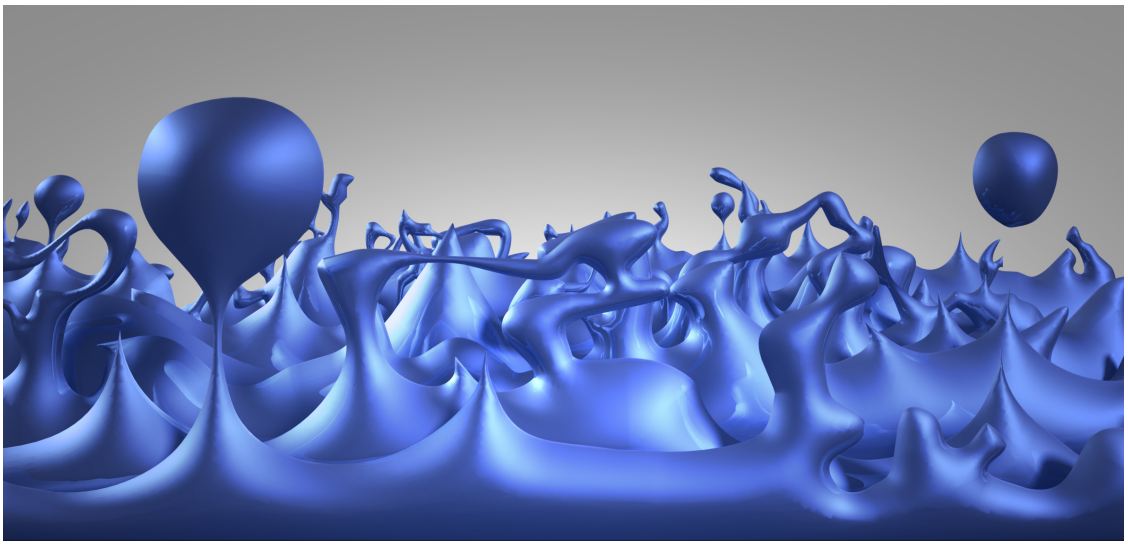
Specifically,

- Chapter 1 introduces the quantum vacuum;
- Chapter 2 introduces the simulation project and outlines the numerical algorithm;
- Chapter 3 introduces *HEWES*: The Heisenberg–Euler Weak-Field Expansion Simulator;
- Chapter 4 demonstrates simulations performed with the help of *HEWES*;
- Chapter 5 discusses performance analysis and tuning;
- Chapter 6 goes into some detail about the accuracy of the solver;
- Chapter 7 provides an outlook on ongoing and future projects; and
- Chapter 8 closes the thesis with the conclusion.

Contents

Chapter 1

The Quantum Vacuum



Vacuum fluctuations (image credit: NASA/CXC/M.WEISS)

In eighteenth-century Newtonian mechanics,
the three-body problem was insoluble.
With the birth of general relativity around 1910
and quantum electrodynamics in 1930,
the two- and one-body problems became insoluble.
And within modern quantum field theory,
the problem of zero bodies (vacuum) is insoluble. [1]

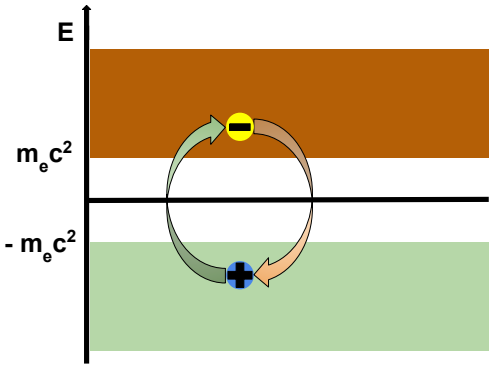


Figure 1.1: Fluctuations in the Dirac sea picture. An electron occupying a position in the sea jumps into the positive energy continuum, leaving a hole (positron) and then returns to its position.

1.1 Introduction

Due to the laws of quantum mechanics the vacuum is not the void of empty space. Uncertainty, which is inherent in the quantum nature, prohibits the *nothing*. Particles can and have to appear out of nothing for very short periods of time, according to Heisenberg's uncertainty relation with respect to energy and time $\triangle E \cdot \triangle t \geq \hbar/2$ in conjunction with Einstein's energy-mass relation $E = mc^2$. Constrained by symmetry, particles thereby emerge in pairs of particle and antiparticle with opposite spins, glimpsing up and vanishing again, forming the quantum vacuum *fluctuations*.

In quantum field theory (QFT) the vacuum therefore manifests itself as a vibrant place of particle–antiparticle fluctuations. In the language of Feynman diagrams it is formed of closed-loops consisting of purely virtual processes without external lines. These need nevertheless to be taken into account and yield profound results.

The pervasive charged-particle fluctuations act as virtual dipoles and thereby render the quantum vacuum a dynamical, polarizable medium. Under the influence of an external electric field, the vacuum can be polarized. As a consequence of the coupling between electromagnetic radiation and the charged-particle fluctuations, effective light–light interaction, or photon–photon scattering, can be mediated.

In quantum electrodynamics (QED) the fluctuations consist of electron–positron pairs. Employing the somewhat old-fashioned Dirac sea picture, the vacuum would be conceived of as a boiling Dirac sea. Owing to quantum uncertainty the sea cannot be calm and the vacuum is conferred a temperature (energy). In other words, it is the vacuum energy that causes a permanent bubbling in the sea, as depicted in the title image of this chapter and in the animations referenced in [2]. A suspected relation to dark energy is discussed below.

Bubbles correspond to electrons on the surface of the Dirac sea that jump over the $2m_e$ mass-gap into the positive continuum, leaving holes, and falling back into the sea [3]. A depiction is given in Figure 1.1.

According to the uncertainty relation with respect to energy and time, these bubbles are tiny and elapse (annihilate) very quickly. The scales are defined by the Compton length and time of the particles, here electrons/positrons. This is so quick that these particles are only virtual and can merely have indirect effects in the form of quantum loop corrections. In this sense, the quantum vacuum can be understood as being empty of real matter, yet full of virtual matter, which does not

give note to itself unless perturbed.

Outline

The probably best known quantum vacuum phenomena are recapitulated in Section 1.2 before in Section 1.3 the most intriguing effect of the quantum vacuum, the creation of matter out of nothing, is introduced. Further fundamental consequences of the quantum vacuum theory and recent advances in the field are outlined in Section 1.4.

The focus of the present work lies on optical effects. These are introduced in Section 1.5 with an outline of the paramount Heisenberg–Euler Lagrangian and its weak-field expansion following in Section 1.6. The latter forms the theoretical basis of the ongoing investigation put forward in the subsequent chapters.

1.2 Commonly known quantum vacuum effects

Lamb shift

The Lamb shift is among the best known effects of early quantum field theory. It constitutes a fundamental discovery during the study of atoms that led to a deeper understanding of particles and in particular quantum fields [4–6]. The effect, absent in Dirac's relativistic quantum mechanics [7], can be explained by QED.

Energy levels in atoms, which are supposed to be exactly equal according to Dirac's theory, in fact exhibit tiny differences both due to the self-energy of electrons, i.e., the *Zitterbewegung* that alters the effective nuclear potential sensed by the electron, and vacuum polarization that also shields parts of the nuclear potential. This implies that the effective potential acting on the electron is modified by vacuum polarization effects, leading to a perturbation of the trajectory at the magnitude of the Compton scale of the electron. Vertex corrections leading to the anomalous magnetic moment of the electron form another contribution to the Lamb shift.

Casimir effect

The Casimir effect is induced by placing two conducting plates very near to each other in the vacuum. Thereby, boundary conditions are imposed onto the virtual photons between the plates, but not onto those outside. The restriction of allowed wavelengths on the inside limits the number of possible states there and hence causes a pressure from the unrestricted outside. As a consequence, the plates are pushed together [8, 9]. The effect has oftentimes been verified experimentally, see, e.g., [10, 11], and is today under great control [12, 13].

The effect of vacuum polarization is well known, yet there are numerous non-intuitive effects beside the commonly known Casimir effect and Lamb shift that are due to the quantum nature of the vacuum.

1.3 Matter out of nothing

Energy is matter and the other way round, according to Einstein's relation $E = mc^2$. It is therefore not only possible to produce energy from matter as in fission reactors, but also to produce matter from energy. The latter happens all the time in particle collider experiments, where particles are

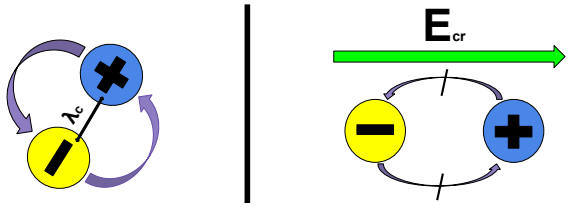


Figure 1.2: Vacuum polarization. *Left:* Unordered virtual electron–positron pairs are created and annihilated at the Compton scale in the vacuum. *Right:* An electric field is applied and polarizes the vacuum. When the critical field strength is reached, a large amount of pairs is torn apart.

accelerated to high momenta and brought to collision. Thereby, large amounts of new particles are created that can subsequently be detected and analyzed.

Traditionally, QFT is tested with such collider experiments operating in the high-energy but low-intensity regime, thus called high-energy physics. Another approach is to explore the low-energy, high-intensity regime of strong-field physics, which owing to a lack of sensitive apparatuses at sufficiently high-intensities has not been accessible thus. Dedicated to the vacuum, this approach might all the same be a path towards the discovery of new physics, since the fluctuations in quantum field theory generally encompass all existing particles. Hence, the influence of yet unknown particle fields might be inscribed in quantum vacuum signatures [14–16].

This field of research was opened by a thought related to the idea of particle collisions that emerged in the early days of quantum field theory: Can matter be created from nothing but pure energy? The detailed answer was given by Julian Schwinger in 1951 [17], whilst the idea dates back to the 1930s with Sauter, Euler, Kockel, Heisenberg, and Weisskopf [18–21]. The creation of *real* charged-particle pairs can be induced by an external electric field that tears the virtual particle–antiparticle pairs apart, so far that they do not annihilate each other again, see Figure 1.2.

This sounds mystical: Matter produced by light out of the nothing. For this effect to take place, however, extremely high intensities of the applied electric field are required. The critical field strength, where the effect of pair creation becomes dominant, is $E_{\mathrm{cr}} \sim 10^{18}\,\mathrm{V\,m^{-1}}$, c.f. Section 1.6.1. The ex-virtual particles soak up the energy for their masses from the applied external field.

The creation of real electron–positron pairs by a strong and (quasi-) static electric field, called Schwinger effect after its theoretical discoverer, has been a long-standing but long unobserved prediction of early quantum theory. Until only recently. The experimental breakthrough is discussed in Section 1.4.

Pair production based on vacuum excitation in electrodynamics was described by Schwinger via the effective action approach. It is a profound effect that inherently unites quantum theory with ultra-relativistic dynamics. The effective action approach makes use of the Heisenberg–Euler Lagrangian, discussed in Section 1.6. The imaginary part of the latter causes a non-persistence of the vacuum state as a consequence of excessive pair creation [22, 23]. By virtue of an exponential suppression of pair creation in this formalism, it is inferred that only the unprecedented field strength E_{cr} is capable of causing a breakdown of the vacuum [24]. That subject not part of the investigations of the present work, but debated in the outlook in Chapter 7.

To reestablish a connection to high-energy physics, a similar phenomenon as depicted in Figure 1.2 is observable with bound states of two quarks with opposite charge, mesons, when torn apart.

With an electric field of sufficient strength the quarks can only be separated at the cost of the simultaneous creation of two further quarks, such that the final product are two separate mesons. Hence, the energy supplied by the electric field has to be at least as much as to create the extra masses.

1.4 Fundamental ramifications and recent breakthroughs

QED in vacuum is generally considered to be the most successful physical theory in terms of agreement with experiments [25]. For an overview of the first steps towards the nonlinear QED vacuum and the ensuing historical development, see [26]. In [27] an introduction to QED and QFT is given, beginning with the concept of vacuum fluctuations. This vividly makes clear once more the deep, fundamental level the fluctuations constitute to quantum theory.

Relation to fundamental physics

The quantum vacuum is the fundamental ground state of nature as described by quantum field theory. With quantum fields dominating the universe, the latter can be considered having its own zero-point energy attributable to the uncertainty principle. The accelerating expansion of the universe caused by dark energy may be a consequence of vacuum fluctuations, with the microscopic vacuum activity conferring space the dark energy [28, 29]. The dark energy factor, the cosmological constant, is modeled as a universal field of energy in empty vacuum [30]. However, there is the unnatural disparity between the cosmological constant of general relativity, measured via the expansion rate of the universe, and the QFT prediction for vacuum energy, calculated via the vacuum loop processes mentioned above [31, 32]. The calculation of Feynman diagrams generates an extraordinary large value for the zero-point energy, but fortunately the expansion of the universe is much slower than that prediction.

Vacuum fluctuations are at the origin of Hawking radiation [33]. Fluctuations at the event horizons of black holes are torn apart, real particles come into existence, taking the required energy from the black hole that slowly evaporates. Vacuum fluctuations are similarly connected to the Unruh effect. A constantly accelerated observer might witness real particles where quantum fluctuations are present to still observers [34, 35].

Random quantum fluctuations in the early universe could have been stretched by inflation, giving rise to the observable inhomogeneities in density and the cosmic background radiation. Tiny temperature differences in the cosmic microwave background are thus the scars of vacuum fluctuations in the early universe. As a consequence, the vacuum fluctuations are suspected to be responsible for the formation of vast galaxy clusters there are today [36].

A currently hot topic is the anomalous magnetic moment of the muon that quantum vacuum polarization is suspected to be a key contribution to [37–42]. Many, and possibly yet unknown, particles in virtual form affect the magnetic moment. This underlines the fact that via the quantum vacuum new physics can be investigated.

Recent experimental advances

Only recently, the Schwinger effect has finally been observed in graphene superlattices [43] and light–light interactions have been detected in crystals [44]. This form of research constitutes the basis of nonlinear optics. The effects have not been observed in the quantum vacuum, though.

The Schwinger effect is to be distinguished from the Breit-Wheeler process [45–47] and general multi-photon pair production [48, 49], where the pairs are created dynamically through photon collisions or decays, with photons of frequencies higher than the Compton frequency of the electron. While in the Schwinger case high field strengths are required, the latter case requires high energies. The former is "creation of matter from nothing", the latter "from light". The distinction in an experiment can be more easily achieved, e.g., via the additional use of time-dependent magnetic fields instead of employing merely an electric field triggering the effects [50].

An investigation of the comparatively small energy density of the vacuum discussed above is in progress with the goal of measuring the "weight" of the vacuum [51]. The ansatz is to measure the weight of virtual particles in the vacuum with the help of the Casimir force in relation to the gravitational force. That would be another important step towards a possible reconciliation of the big theories of the large and the small scales.

Effects of the quantum vacuum have even become visible to humans. Lasers trying to focus discrete photons to accurate positions increase the uncertainty in photon number. In a remarkable experiment at room temperature, the resulting fluctuation force pushed mirrors such that it was noticeable with a human eye [52]. Remarkably, the quantum vacuum even pushes the 40 kg mirrors of the *LIGO* experiment [53].

Experiments have verified heat transfer through the vacuum via fluctuations [54] and the fluctuations are suspected to cause friction [55]. Even though it has not made its way into experiments, the *NASA* investigated the idea of using Casimir forces enabling propulsion techniques for vacuum-powered spacecrafts [56].

Finally, the arrow of time might not be a result of initial conditions and sheer statistics, but in fact vacuum effects down to the Planck scale might influence trajectories even of macroscopic objects and make reversions impossible, as simulations of a system of three black holes suggest [57]. The universe is a chaotic system, such that even the tiniest fluctuations of particles may unforeseeably influence the trajectories of the largest existing objects. Yet, the quantum fluctuations may not occur purely at random, there is evidence for correlations [58].

1.5 Photon–photon interactions

The vacuum can be excited, e.g., by external fields. Within the Standard Model, the leading effect arises from the effective coupling of electromagnetic fields via virtual electron–positron pairs, and thus is governed by quantum electrodynamics [23].

One kind of quantum vacuum effects of particular interest for laboratory experiments is the class of optical effects. While direct tree-level photon–photon interactions are strictly forbidden in quantum electrodynamics, virtual particle–antiparticle pairs can mediate interactions of photons. Radiation on the other hand can influence the behavior of the particle–antiparticle dipoles in the vacuum. This has already been set forth in the introduction, depictions are given in Figures 1.2 and 1.3.

Nonlinear interactions can be triggered by high intensities of the external radiation field and cause a variety of phenomena manifesting the dynamic structure of the quantum vacuum because of the profound consequence for the equations of motion of electromagnetism. Maxwells linear theory of vacuum electrodynamics [59] is extended by effective nonlinear self-interactions, violating the classical superposition principle, at least for electromagnetic fields that are strong enough to

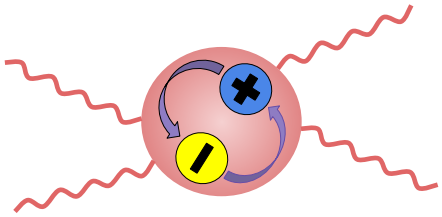


Figure 1.3: Photon–photon interaction via virtual electrically charged pairs in the vacuum.

excite the vacuum. Consequently, there are also quantifiable effects in the dynamics of electromagnetic waves due to the presence of virtual particle pairs in the vacuum.

A low-energy effective theory governing the nonlinear dynamics of strong classical electromagnetic fields in the quantum vacuum as a medium was famously devised by Euler, Kockel, Heisenberg, and Weisskopf [19–21]. Schwinger incorporated the theory into the larger QED picture [17]. The so-called Heisenberg–Euler Lagrangian, outlined in Section 1.6, constitutes the one-loop effective theory of QED, taking the radiation as classical background field which ought to vary over much larger scales than the Compton scale of the electron in space and time.

There are numerous optical effects attributable to the quantum nature of the vacuum. In general, these are based on light-by-light scattering phenomena [19, 60–73]. There is vacuum birefringence, caused by a preferred direction for charged particles in a strong electromagnetic field [74–92], which in inhomogeneous fields is a consequence of broken translational invariance accompanied by vacuum diffraction [80, 93–97]. A peculiar effect is quantum reflection with photons [98, 99]. Further, there are photon merging [100–103] with the generation of higher harmonics [104–115], and photon splitting [103, 116–123].

A promising approach to their detection is the investigation of the asymptotic dynamics of *probe* photons after passing through a strong-field region in the form of a high-intensity laser *pump* pulse. Probe photons traversing a strong field region can indirectly sense the applied strong pump field via the quantum fluctuations which couple to both probe and pump fields [85]. Due to the nonlinear interaction with the power pulses, signal photons of the probe field scatter in such a way that their dynamics or polarization distinguishes them from the main pulses. Properties and dynamics of the quantum vacuum, in turn, are hereby encoded into the signal photons.

This class of quantum vacuum experiment, where one electromagnetic field drives the nonlinear effect while the other carries its signature, is called *all-optical*. These experiments at the high-intensity frontier promise unprecedented detail in the study of the quantum vacuum in the near future. The aspiration is that the rapid and steady advancements in ultra-intense laser physics [124, 125] combined with theoretically optimized specifications will soon facilitate the experimental discovery [126–128]. Predictions from analytical calculations are idealized and make use of sufficiently simple interaction scenarios, which exposes the need for the versatility that is provided by numerical solvers.

For general reviews of the quantum vacuum, see [22, 26, 126–134]. A review of strong-field vacuum effects can be found in [135]. A concise overview of probing the nonlinear vacuum with laser experiments and a short survey of all kinds of quantum vacuum effects is given in [126]. For recent and future experimental developments of high-field fundamental physics, see [136].

1.6 The Heisenberg–Euler effective action

Unlike the strongly coupled QCD vacuum, where fermions condense, the vacuum of QED can efficiently be accessed perturbatively. Employing the perturbative effective action approach, the complete quantum theory is considered as a classical description of the low-energy degrees of freedom relevant to the vacuum [22]. The low-energy degrees of freedom are given by photons with wavelengths below the Compton length of the electron $\lambda_C \sim 10^{-12}\,\mathrm{m}$. The effective field theory is described by the Heisenberg–Euler Lagrangian and corresponds to QED in a constant electromagnetic background field after integrating out the high-energy degrees of freedom to the one-loop order [20, 130]. In consideration of the fact that considered radiation fields vary slowly compared to the Compton wavelength of the electron, it is assumed that the locally constant field approximation is valid [133, 137, 138]. Comprehensive reviews including the historical development and discussing the historic works are given in [26, 130].

In a convenient form the Heisenberg–Euler Lagrangian reads [130]

$$\mathscr{L}_{\mathrm{HE}} = -\frac{c^5 m_e^4}{8\hbar^3\pi^2}\int_0^\infty \frac{e^{-s}}{s^3}\left(\frac{s^2}{3}\left(a^2-b^2\right)-1+abs^2\cot(as)\coth(bs)\right)ds\,, \tag{1.1}$$

where

$$a = \sqrt{\sqrt{\mathscr{F}^2+\mathscr{G}^2}+\mathscr{F}}\,,\quad b = \sqrt{\sqrt{\mathscr{F}^2+\mathscr{G}^2}-\mathscr{F}}\,, \tag{1.2}$$

$$\mathscr{F} = -\frac{c^2 F^{\mu\nu}F_{\mu\nu}}{4E_{\mathrm{cr}}^2}\,,\quad \mathscr{G} = -\frac{c^2 F^{\mu\nu}F^*_{\mu\nu}}{4E_{\mathrm{cr}}^2}\,, \tag{1.3}$$

$$E_{\mathrm{cr}} = \frac{m_e^2 c^3}{e\hbar} = 1.323\times 10^{18}\,\mathrm{V\,m^{-1}}\,, \tag{1.4}$$

c is the vacuum speed of light, $\hbar$ the reduced Planck constant, and e, m_e are the charge and mass of the electron. The quantities F and F^* denote the electromagnetic field strength tensor and its dual and E_{cr} is the above mentioned critical field strength, which marks the scale where the magnitude of the vacuum nonlinearities becomes dominant.

Combining the classical Maxwell Lagrangian, with the nonlinear quantum vacuum corrections captured in the Heisenberg–Euler theory, yields a Lagrangian for the QED vacuum

$$\mathscr{L} = \mathscr{L}_{\mathrm{MW}} + \mathscr{L}_{\mathrm{HE}}\,. \tag{1.5}$$

The Maxwell Lagrangian is given by

$$\mathscr{L}_{\mathrm{MW}} = -\frac{1}{4\mu_0}F^{\mu\nu}F_{\mu\nu} = \frac{E_{\mathrm{cr}}^2}{c^2\mu_0}\mathscr{F}\,, \tag{1.6}$$

with the vacuum permeability

$$\mu_0 = 1.257\times 10^{-6}\,\mathrm{N\,A^{-2}}\,. \tag{1.7}$$

With the Maxwell vacuum referred to as the linear vacuum, where the superposition principle holds for the linear differential equations, the Heisenberg–Euler Lagrangian extends the theory by nonlinear terms, light–light interactions, and the vacuum is referred to as the nonlinear vacuum. The latter covers all quantum effects of vacuum polarization arising in a background electromagnetic field to the one-loop order.

1.6.1 QED strong-field regime

A short summary of the parameters that define the strong-field QED regime is in order. SI units are used throughout this work. Vacuum nonlinearities of QED are suppressed by inverse powers of the electron mass in the effective action approach. To produce electron–positron pairs in vast amounts, the Schwinger limit has to be accessed. The electron mass and charge are given by [139]

$$m_e = 9.109 \times 10^{-31}\,\mathrm{kg}\,, \tag{1.8}$$

$$e = 1.602 \times 10^{-19}\,\mathrm{C}\,. \tag{1.9}$$

Further, the vacuum speed of light and the Planck constant are given by

$$c = 2.998 \times 10^{8}\,\mathrm{m\,s^{-1}}\,, \tag{1.10}$$

$$h = 6.626 \times 10^{-34}\,\mathrm{J\,s}\,. \tag{1.11}$$

The Compton length and time of the electron are given by

$$\lambda_C = \frac{h}{m_e c} = 2.426 \times 10^{-12}\,\mathrm{m}\,, \tag{1.12}$$

$$t_C = \frac{\lambda_C}{c} = \frac{h}{m_e c^2} = 8.093 \times 10^{-21}\,\mathrm{s} \tag{1.13}$$

and the reduced Compton length and time of the electron by

$$\lambda_{C,\mathrm{red}} = \frac{\hbar}{m_e c} = 3.862 \times 10^{-13}\,\mathrm{m}\,, \tag{1.14}$$

$$t_{C,\mathrm{red}} = \frac{\hbar}{m_e c^2} = 1.288 \times 10^{-21}\,\mathrm{s}\,. \tag{1.15}$$

The critical field strengths that form the Schwinger- or Sauter-limit are then given by [18]

$$E_{\mathrm{cr}} = \frac{c^3 m_e^2}{e\hbar} = 1.323 \times 10^{18}\,\mathrm{V\,m^{-1}}\,, \tag{1.16}$$

$$B_{\mathrm{cr}} = \frac{E_{cr}}{c} = \frac{c^2 m_e^2}{e\hbar} = 4.414 \times 10^{9}\,\mathrm{T}\,. \tag{1.17}$$

For the Schwinger effect only an electric field is required. The critical field strength corresponds to the electric field that over the length of the reduced Compton wavelength exposes the electron rest energy

$$\lambda_{C,\mathrm{red}}\, e E_{\mathrm{cr}} = m_e c^2\,, \tag{1.18}$$

or, equivalently, over twice that length the rest energy of an electron–positron pair, see also Figure 1.2.

Electric fields with the required critical field strength E_{cr} open up a new regime of strong-field QED, where the properties of the vacuum are substantially different [17, 18, 140]. At the corresponding intensity of

$$I_{\mathrm{cr}} = \epsilon_0\, c\, E_{\mathrm{cr}}^2 = 4.648 \times 10^{33}\,\mathrm{W\,m^{-2}} \tag{1.19}$$

the vacuum is no longer stable. It is said to break down because fluctuating pairs are torn apart

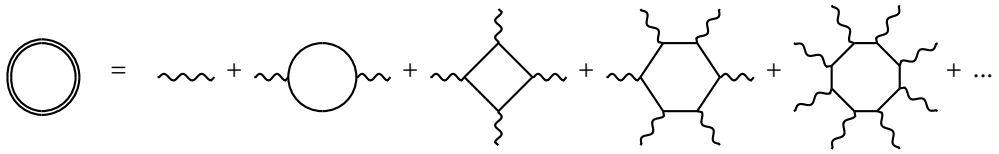


Figure 1.4: Depiction of the Heisenberg–Euler effective action of QED. The double-lined loop to the left represents the irradiated electron–positron loop, denoting the coupling of the external field to the fermion loop to all orders in the field strength. The diagrams to the right represent the linear Maxwell theory, the one-loop correction to it due to vacuum polarization, and the four-, six-, and eight-photon contributions in the Heisenberg–Euler weak-field expansion.

and real electron–positron pairs are expected to be produced in a vast amount. This will cause an attenuation of the external field, an intricate feedback mechanism that is picked up in Chapter 7.

1.6.2 Weak-field expansion

For field strengths below the critical value a weak-field expansion can be performed that yields higher order interactions consecutively. The term "weak" is relative, since E_{cr} can be considered a very high field strength such that the weak-field expansion is valid even for the maximal field strengths attainable in the laboratory today, which range up to $10^{-3}E_{\mathrm{cr}}$ [141]. The weak-field expansion of the Heisenberg–Euler Lagrangian (1.1) up to fourth order in $\mathscr{F}$ and $\mathscr{G}$ is given by

$$\mathscr{L}_{\mathrm{HE}} \approx \quad 4\epsilon_0 \frac{E_{\mathrm{cr}}^2 \alpha}{360\pi}\left(4\mathscr{F}^2 + 7\mathscr{G}^2\right) \tag{1.20a}$$

$$-4\epsilon_0 \frac{E_{\mathrm{cr}}^2 \alpha}{630\pi}\left(8\mathscr{F}^3 + 13\mathscr{F}\mathscr{G}^2\right) \tag{1.20b}$$

$$+4\epsilon_0 \frac{E_{\mathrm{cr}}^2 \alpha}{945\pi}\left(48\mathscr{F}^4 + 88\mathscr{F}^2\mathscr{G}^2 + 19\mathscr{G}^4\right), \tag{1.20c}$$

with the vacuum permittivity

$$\epsilon_0 = 8.854 \times 10^{-12}\,\mathrm{A\,s\,V^{-1}\,m^{-1}} \tag{1.21}$$

and the fine-structure constant

$$\alpha = \frac{e^2}{4\pi\epsilon_0\hbar c} \approx \frac{1}{137}\,. \tag{1.22}$$

A Feynman-diagrammatic depiction of the weak-field expansion is given in Figure 1.4.

It should be noted that the Heisenberg–Euler Lagrangian solely depends on the electromagnetic field invariants $\mathscr{F}$ and $\mathscr{G}$, and subsequently (1.20a) is of the order $\mathscr{O}((E/E_{\mathrm{cr}})^4)$. At one-loop order of the quantum corrections to Maxwell's equations, couplings of the vacuum only to even orders of the radiation field are identified as a consequence of Furry's theorem (charge conjugation symmetry) [26, 142].

The effective theory generates the whole perturbative expansion in terms of the external photon legs at once and can be graphically represented by a *dressed* loop, representing the irradiated electron–positron vacuum loop, as in Figure 1.4 [113, 133]. Hence, (1.20a) represents the four-photon contributions, (1.20b) the six-photon contributions, and (1.20c) the eight-photon contributions to the closed double-lined loop. Since the field strengths are constrained to be weaker than the critical field strength, $c|F_{\mu\nu}| < E_{\mathrm{cr}}$, higher order terms can be neglected in the low-intensity

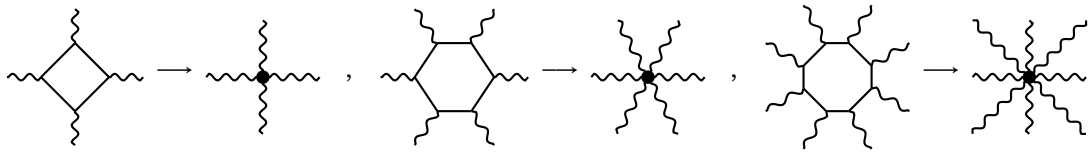


Figure 1.5: Effective vertices for photon–photon coupling in the vacuum due to the Heisenberg–Euler weak-field expansion. Above the Compton scale of the electron, the effective couplings can be described as direct, local interactions.

regime.

It has to be stressed again that pair production is exponentially suppressed in the Heisenberg–Euler Lagrangian. Therefore, in the weak-field limit the properties of the nonlinear vacuum can be described exclusively by radiation fields. The effective photon–photon vertices after integrating out the matter field are shown in Figure 1.5.

For the validity of the Heisenberg–Euler weak-field expansion the regime restriction is therefore two-fold. Energies need be below the electron mass and field strengths below the critical values. Well below these scales, it is solid to employ the Heisenberg-Euler Lagrangian $\mathscr{L}_{\mathrm{HE}}$ in weak-field expansion. This has been proven both analytically [137] and numerically [138].

1.6.3 Modified Maxwell equations

The classical Maxwell–Ampère circuital law in media

$$\partial_t \left(\frac{\vec{E}}{c^2} + \mu_0 \vec{P} \right) = \nabla \times \left(\vec{B} - \mu_0 \vec{M} \right) \tag{1.23}$$

can be compared to the modifications resulting from the Heisenberg–Euler weak-field expansion to infer macroscopic polarization and magnetization terms of the quantum vacuum.

To obtain the nonlinear modifications to the linear Maxwell equations, an explicit rescaling of the electric and magnetic fields $\vec{E}$ and $\vec{B}$ to the critical field strengths is instructive, such that the field strengths are given in units of E_{cr},

$$\vec{E} \to \frac{\vec{E}}{E_{\mathrm{cr}}} \,, \quad \vec{B} \to \frac{\vec{B}}{E_{\mathrm{cr}}} \,. \tag{1.24}$$

From the Lagrangian of the quantum vacuum (1.5) the equations of motion are obtained with the help of the Euler–Lagrange equations with respect to the normalized fields, yielding the modified Maxwell–Ampère law

$$\partial_t \left(\frac{\vec{E}}{c^2} + \mu \frac{\partial \mathscr{L}_{\mathrm{HE}}}{\partial \vec{E}} \right) = \nabla \times \left(\vec{B} - \mu \frac{\partial \mathscr{L}_{\mathrm{HE}}}{\partial \vec{B}} \right) \,, \tag{1.25}$$

with the definition

$$\mu = \frac{\mu_0}{E_{\mathrm{cr}}^2} \,. \tag{1.26}$$

Comparing the modifications due to the effective Heisenberg–Euler theory (1.25) to those in a generic (nonlinear) medium (1.23) is then possible. Effective photon–photon interactions hence imply the appearance of macroscopic nonlinear polarization and magnetization terms of the QED

vacuum given by

$$\vec{P} = \frac{\mu}{\mu_0}\frac{\partial \mathscr{L}_{\mathrm{HE}}}{\partial \vec{E}} = \frac{1}{E_{\mathrm{cr}}^2}\frac{\partial \mathscr{L}_{\mathrm{HE}}}{\partial \vec{E}} \quad \text{and} \quad \vec{M} = \frac{\mu}{\mu_0}\frac{\partial \mathscr{L}_{\mathrm{HE}}}{\partial \vec{B}} = \frac{1}{E_{\mathrm{cr}}^2}\frac{\partial \mathscr{L}_{\mathrm{HE}}}{\partial \vec{B}} \,. \tag{1.27}$$

These can be straightforwardly computed in the weak-field expansion (1.20).

This clarifies that, as described in Section 1.5, the resulting equations of motion supplement Maxwell's linear equations of electromagnetism by nonlinear photon–photon interactions that can be triggered by high-intensity laser pulses.

1.6.4 General approach and motivation

There exists an abundance of perturbative analytical treatments of the Heisenberg–Euler model in weak-field approximation. Since these approaches are limited to approximations and manageable configurations and at the same time experimental requirements for the detection of these signals are high, the need for support from the numerical side is apparent. A numerical algorithm and simulation code for the nonlinear effective light–light interactions in the weak-field expansion of the Heisenberg–Euler theory of quantum electrodynamics is presented in this thesis.

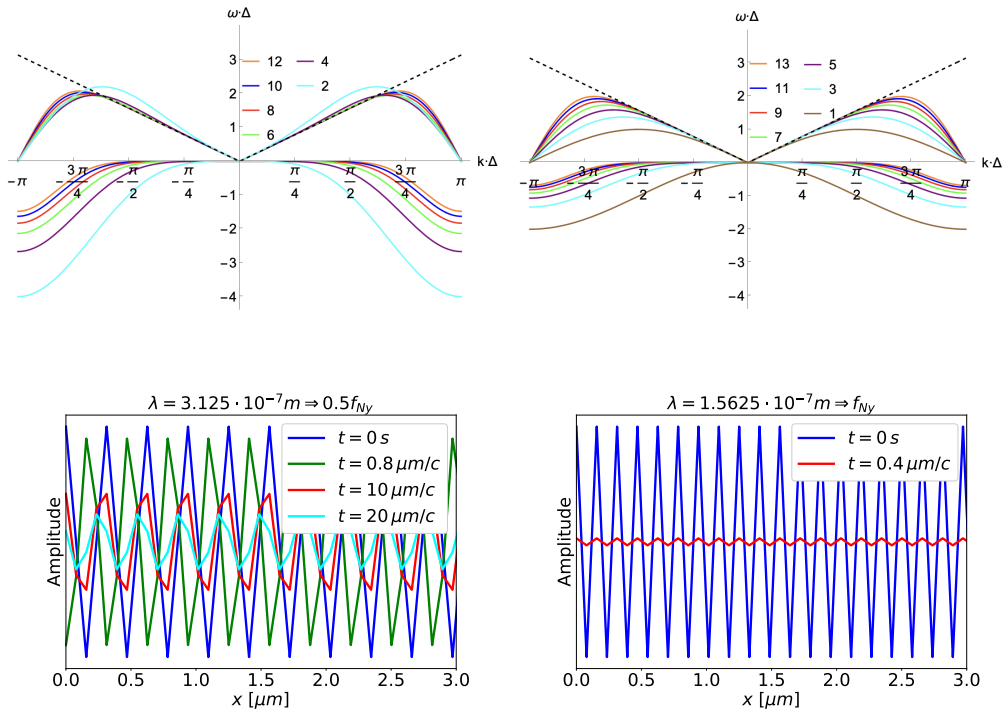
The typical idea is to approach, or *probe*, the vacuum loops via external lines. External perturbations are electromagnetic field lines, i.e., probe photons. Signatures of the quantum vacuum nonlinearities are inscribed in the properties of the outgoing probe photon, the *signal photon*. Throughout the present work so-called probe–pump scenarios described in Section 1.5 will be considered, where a probe pulse is influenced by vacuum nonlinearities induced by a stronger pump pulse.

Since the measurement is performed distant from the interaction region, it is the asymptotic state of a probe that is commonly investigated [111]. Nonetheless, a full picture Heisenberg–Euler solver even enables the time-resolving of nonlinear interactions.

Recent advances in laser and detector technology spur the hope that novel quantum vacuum signals in the all-optical regime can be detected in the near future.

Chapter 2

The Quantum Vacuum Simulation Algorithm



Dispersion relations and damping effects

Links

***Journal of Computational Physics: X* paper**
https://doi.org/10.1016/j.jcpx.2023.100124

***arXiv* paper**
https://doi.org/10.48550/arXiv.2202.09680

***Wolfram Notebook Archive* notebook with *Mathematica* analyses**
https://notebookarchive.org/2022-08-eb2cjxb.

***GitLab* code repository**
https://gitlab.physik.uni-muenchen.de/ls-ruhl/hewes

Code reference
https://gitlab.physik.uni-muenchen.de/ls-ruhl/hewes/-/blob/main/docs/ref.pdf

Highlights

- 13^{th} order of accuracy stable numerical scheme
- Dispersion relation with vacuum-like behavior even for small wavelengths
- Incorporation of a nonphysical modes filter

2.1 Introduction

Wave equations in the nonlinear Heisenberg–Euler weak-field model can only be solved analytically for restrictive assumptions. Almost all analytical approaches to compute nonlinear vacuum effects, as found in the literature cited in Section 1.5, have shortcomings. Consequentially, these attempts are still limited to simple scenarios, impeding experimental verification. Approximations limit the accuracy of predictions and the precision the theory can be tested with. Precision tests require accurate theoretical predictions for arbitary laser field configurations [143].

The present chapter introduces a numerical algorithm for the solution of the modified Maxwell equations in the Heisenberg–Euler weak-field approximation. The work is based on the numerical scheme outlined in one spatial dimension in [111–113] and extended to three dimensions in [144, 145]. The implemented solver for up to three-dimensional simulations is presented in [146–148].

The algorithm is numerically stable and aliasing-free with a well-behaved dispersion relation. Specifically, the crucial dispersion relation of the algorithm renders it: I) real vacuum-like in a large range of wavenumbers, II) stable for any frequency, and III) aliasing-free, i.e, annihilating nonphysical grid modes.

The accuracy of the numerical scheme relies on the assumptions that the fields vary on scales larger than the Compton scale of the electron and the field strengths involved are below the critical field for pair creation in the vacuum. In addition, a viable numerical algorithm is restricted by the computational load it generates, even more so in the realm of high frequencies. The advantage of a numerical approach, however, is that it can principally solve almost any interaction scenario for nearly any arrangement of laser pulses. Limitations are discussed in Chapters 4 and 6.

Outline

In Section 2.2, the numerical scheme that solves the nonlinear Maxwell equations in 3+1 dimensions is outlined. First, the nonlinear modifications to Maxwell's equations due to the Heisenberg–Euler theory are formulated in terms of a matrix partial differential equation. With the help of finite differences for spatial derivatives, the system of partial differential equations is turned into a system of ordinary differential equations. Dispersion effects on the lattice are discussed in Section 2.3. Finally, a comparison to other numerical approaches to the Heisenberg–Euler theory in weak-field expansion is drawn in Section 2.5.

2.2 Equations of motion

2.2.1 A merged partial differential equation

The modified Maxwell–Ampère circuital law (1.23) with the polarization and magnetization terms in (1.27) from Section 1.6.3 is merged with the Maxwell–Faraday law of induction

$$\partial_t \vec{B} = -\nabla \times \vec{E} \,, \tag{2.1}$$

which persists in the nonlinear vacuum, such that a single partial differential equation describing the whole dynamics of the system is formulated. First, the curl of $(\vec{B}-\vec{M})$ can be written as

$$\begin{aligned}\nabla\times(\vec{B}-\vec{M}) = &\underbrace{\begin{pmatrix}0&0&0\\0&0&-1\\0&1&0\end{pmatrix}}_{\mathbf{Q}_x}\partial_x(\vec{B}-\vec{M}) + \underbrace{\begin{pmatrix}0&0&1\\0&0&0\\-1&0&0\end{pmatrix}}_{\mathbf{Q}_y}\partial_y(\vec{B}-\vec{M})\\ &+\underbrace{\begin{pmatrix}0&-1&0\\1&0&0\\0&0&0\end{pmatrix}}_{\mathbf{Q}_z}\partial_z(\vec{B}-\vec{M}) = \sum_{j\in\{x,y,z\}}\mathbf{Q}_j\partial_j(\vec{B}-\vec{M})\,.\end{aligned} \tag{2.2}$$

With the help of the vector

$$\vec{f} = \left(\vec{E}/c, \vec{B}\right)^{\mathrm{T}} \tag{2.3}$$

the Maxwell equations (1.23) and (2.1) can then be expressed as

$$(\mathbf{1}_6+\mathbf{A})\frac{\partial_t}{c}\vec{f} = \sum_{j\in\{x,y,z\}}\mathbf{Z}_j\partial_j\vec{f}\,, \tag{2.4}$$

with the 6×6 identity matrix $\mathbf{1}_6$ and the matrices $\mathbf{A}$ and $\mathbf{Z}_j$ are given by

$$\mathbf{A} = \begin{pmatrix}\mathbf{J}_{\vec{P}}\left(\vec{E}\right) & \mathbf{J}_{\vec{P}}\left(\vec{B}\right)\\ \mathbf{0}_3 & \mathbf{0}_3\end{pmatrix}, \quad \mathbf{Z}_j = \begin{pmatrix}-\mathbf{Q}_j\mathbf{J}_{\vec{M}}\left(\vec{E}\right) & \mathbf{Q}_j-\mathbf{Q}_j\mathbf{J}_{\vec{M}}\left(\vec{B}\right)\\ -\mathbf{Q}_j & \mathbf{0}_3\end{pmatrix}, \tag{2.5}$$

where $\mathbf{0}_3$ is the 3×3 zero matrix and

$$\mathbf{J}_{\vec{P}}(\vec{E}) = \epsilon_0\frac{\partial\vec{P}}{\partial\vec{E}}, \quad \mathbf{J}_{\vec{P}}(\vec{B}) = \mu_0 c\frac{\partial\vec{P}}{\partial\vec{B}}, \quad \mathbf{J}_{\vec{M}}(\vec{E}) = \mu_0 c\frac{\partial\vec{M}}{\partial\vec{E}}, \quad \mathbf{J}_{\vec{M}}(\vec{B}) = \mu_0\frac{\partial\vec{M}}{\partial\vec{B}}\,, \tag{2.6}$$

with matrix elements

$$\left(\mathbf{J}_{\vec{P}}(\vec{E})\right)_{ij} = \epsilon_0\frac{\partial P_i}{\partial E_j}\,. \tag{2.7}$$

Equation (2.4) contains the full dynamics of electromagnetic fields in the weak-field approximation of the Heisenberg–Euler model.

For illustrative purposes, first the linear vacuum is considered by setting $\vec{P}=\vec{M}=0$, leading to

$$\frac{\partial_t}{c}\vec{f} = \sum_{j\in\{x,y,z\}}\mathbf{Z}_j^{\mathrm{lin}}\partial_j\vec{f}\,, \quad \mathbf{Z}_j^{\mathrm{lin}} = \begin{pmatrix}\mathbf{0}_3 & \mathbf{Q}_j\\ -\mathbf{Q}_j & \mathbf{0}_3\end{pmatrix}. \tag{2.8}$$

$\mathbf{Z}_j^{\mathrm{lin}}$ is diagonalized using the matrices

$$\mathbf{R}_x = \frac{1}{\sqrt{2}}\begin{pmatrix}\sqrt{2}&0&0&0&0&0\\0&1&0&0&0&-1\\0&0&1&0&1&0\\0&0&0&\sqrt{2}&0&0\\0&1&0&0&0&1\\0&0&1&0&-1&0\end{pmatrix}, \quad \mathbf{R}_y = \frac{1}{\sqrt{2}}\begin{pmatrix}0&\sqrt{2}&0&0&0&0\\-1&0&0&0&0&-1\\0&0&-1&1&0&0\\0&0&0&0&\sqrt{2}&0\\-1&0&0&0&0&1\\0&0&-1&-1&0&0\end{pmatrix}, \tag{2.9}$$

$$\mathbf{R}_z = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & \sqrt{2} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{2} \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \end{pmatrix},$$

so that for $\mathbf{Z}_j^{\text{lin}}$ the expressions

$$\mathbf{Z}_j^{\text{lin}} = \mathbf{R}_j^{\text{T}} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \mathbf{R}_j = \mathbf{R}_j^{\text{T}} \text{diag}(0,1,1,0,-1,-1) \mathbf{R}_j \tag{2.10}$$

are obtained. The diagonalization helps to identify modes with specific propagation directions, which becomes important in the construction of the numerical scheme. Equation (2.8) becomes

$$\frac{\partial_t}{c} \mathbf{R}_j \vec{f}(x,y,z;t) = \text{diag}(0,1,1,0,-1,-1) \partial_j \mathbf{R}_j \vec{f}(x,y,z;t) \,, \tag{2.11}$$

where

$$\mathbf{R}_x \vec{f} = \frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{2}E_x \\ E_y - B_z \\ E_z + B_y \\ \sqrt{2}B_x \\ E_y + B_z \\ E_z - B_y \end{pmatrix}, \quad \mathbf{R}_y \vec{f} = \frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{2}E_y \\ -B_z - E_x \\ B_x - E_z \\ \sqrt{2}B_y \\ B_z - E_x \\ -B_x - E_z \end{pmatrix}, \quad \mathbf{R}_z \vec{f} = \frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{2}E_z \\ -B_y + E_x \\ B_x + E_y \\ \sqrt{2}B_z \\ B_y + E_x \\ -B_x + E_y \end{pmatrix}. \tag{2.12}$$

Equations (2.11) and (2.12) imply backward propagation in x-direction of the field components $(E_y - B_z)$ and $(E_z + B_y)$, forward propagation of the field components $(E_y + B_z)$ and $(E_z - B_y)$, and non-propagation of the first and fourth component. The components for the other propagation directions can be read off analogously.

2.2.2 Finite differences

For demonstration purposes only the x-direction is considered in the following. In order to transform the partial differential equation into an ordinary one, finite difference approximations are introduced. The spatial derivative ∂_x is thereby replaced by finite differences

$$\partial_x \left(\mathbf{R}_x \vec{f} \right)(x,y,z) \approx \mathscr{D}_x \left(\mathbf{R}_x \vec{f} \right)(x,y,z), \tag{2.13}$$

where

$$\mathscr{D}_x\left(\mathbf{R}_x\vec{f}\right)(x,y,z)=\sum_\nu \frac{1}{\Delta_x}S_\nu\left(\mathbf{R}_x\vec{f}\right)(x+\nu\Delta_x,y,z)\,. \tag{2.14}$$

The S_ν are stencil matrices and Δ_x is the spatial resolution. The stencil matrices S_ν are derived with the help of a Taylor expansion of a generic function $g(x+\nu\Delta_x)$ around $\nu\Delta_x=0$. The Taylor expansion up to accuracy order six is given by

$$\begin{aligned} g(x+\nu\Delta_x) = g(x)+(\nu\Delta_x)g'(x)+\frac{1}{2}(\nu\Delta_x)^2 g''(x)+\frac{1}{6}(\nu\Delta_x)^3 g'''(x)+\frac{1}{24}(\nu\Delta_x)^4 g^{(4)}(x) \\ +\frac{1}{120}(\nu\Delta_x)^5 g^{(5)}(x)+\frac{1}{720}(\nu\Delta_x)^6 g^{(6)}(x)+\mathscr{O}\left((\nu\Delta_x)^7\right). \end{aligned} \tag{2.15}$$

For the finite differences approximations of derivatives, forward and backward steps can be made use of. For the values $\nu\in\{-3,\ldots,3\}$ the expansion in matrix form reads

$$\begin{pmatrix} g(x-3\Delta_x)\\ g(x-2\Delta_x)\\ g(x-\Delta_x)\\ g(x)\\ g(x+\Delta_x)\\ g(x+2\Delta_x)\\ g(x+3\Delta_x) \end{pmatrix} = \begin{pmatrix} 1 & -3 & 9/2 & -9/2 & 27/8 & -81/40 & 81/80\\ 1 & -2 & 2 & -4/3 & 2/3 & -4/15 & 4/45\\ 1 & -1 & 1/2 & -1/6 & 1/24 & -1/120 & 1/720\\ 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 1/2 & 1/6 & 1/24 & 1/120 & 1/720\\ 1 & 2 & 2 & 4/3 & 2/3 & 4/15 & 4/45\\ 1 & 3 & 9/2 & 9/2 & 27/8 & 81/40 & 81/80 \end{pmatrix} \cdot \begin{pmatrix} g(x)\\ \Delta_x g'(x)\\ \Delta_x^2 g''(x)\\ \Delta_x^3 g'''(x)\\ \Delta_x^4 g^{(4)}(x)\\ \Delta_x^5 g^{(5)}(x)\\ \Delta_x^6 g^{(6)}x) \end{pmatrix} \tag{2.16}$$

and can be extended to larger ranges of ν and inverted to obtain approximations for the derivatives taking into account more distant points. Recall that only the first derivative is required for the merged equation of motion (2.4).

To illustrate how the finite difference approximation of the first derivative is calculated, $\nu\in\{0,1\}$ is considered for simplicity. It is obtained

$$\begin{pmatrix} g(x)\\ g(x+\Delta_x) \end{pmatrix} = \begin{pmatrix} 1 & 0\\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} g(x)\\ \Delta_x g'(x) \end{pmatrix}, \tag{2.17}$$

which leads to the first derivative of g given by

$$g'(x)\approx g'_{\nu\in\{0,1\}}=\frac{g(x+\Delta_x)-g(x)}{\Delta_x}\,. \tag{2.18}$$

The derivative $g'_{\nu\in\{0,1\}}$ in (2.18) is called first order forward discrete derivative. In the same way the first order backward discrete derivative $g'_{\nu\in\{-1,0\}}$ is obtained with the help of

$$\begin{pmatrix} g(x-\Delta_x)\\ g(x) \end{pmatrix} = \begin{pmatrix} 1 & -1\\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} g(x)\\ \Delta_x g'(x) \end{pmatrix}, \tag{2.19}$$

where

$$g'_{\nu\in\{-1,0\}}=\frac{g(x)-g(x-\Delta_x)}{\Delta_x}\,. \tag{2.20}$$

In order to obtain specific dispersion properties and numerical stability of the discretized ver-

sion of Equation (2.11), it is necessary to introduce forward and backward finite differences for $\mathscr{D}_x$. Dispersion properties and stability are detailed in Section 2.3.

With Equation (2.14) the full expression for the spatial derivatives for distinct propagation directions is formulated

$$\partial_x \vec{f}(x,y,z) = \mathbf{R}_x^{\mathrm{T}} \partial_x \mathbf{R}_x \vec{f}(x,y,z) \approx \mathbf{R}_x^{\mathrm{T}} \sum_\nu \frac{1}{\Delta_x} S_\nu \left(\mathbf{R}_x \vec{f}\right)(x+\nu\Delta_x, y, z) = \mathbf{R}_x^{\mathrm{T}} \mathscr{D}_x \left(\mathbf{R}_x \vec{f}\right)(x,y,z) \,. \tag{2.21}$$

Setting aside non-propagation of modes, the first three elements of the rotated field vector $\mathbf{R}_x \vec{f}$ represent forward-propagating electromagnetic modes, while the last three components of the latter represent backward-propagating ones, as discussed at the end of Section 2.2.1. This understanding is adopted for the discussion of dispersion relations in Section 2.3.

For the nonlinear vacuum the system of ordinary differential equations is obtained from the merged equation of motion (2.4) and Equation (2.21) to be

$$\frac{\partial_t}{c} \vec{f} = (\mathbf{1}_6 + \mathbf{A})^{-1} \sum_{j \in \{x,y,z\}} \mathbf{Z}_j \mathbf{R}_j^{\mathrm{T}} \mathscr{D}_j \mathbf{R}_j \vec{f} \,. \tag{2.22}$$

The inversion of $(\mathbf{1}_6 + \mathbf{A})$ is an expensive operation. An approximation in terms of a truncated geometric series is not satisfying since also larger nonlinear corrections ought to be taken into account. The problem of inverting the matrix can be reduced from a 6×6-dimensional problem to a 3×3-dimensional one by making use of the block form in (2.5) such that [145]

$$(\mathbf{1}_6 + \mathbf{A})^{-1} = \begin{pmatrix} \mathbf{1}_3 + \mathbf{J}_{\vec{P}}\left(\vec{E}\right) & \mathbf{J}_{\vec{P}}\left(\vec{B}\right) \\ \mathbf{0}_3 & \mathbf{1}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{C}^{-1} & -\mathbf{C}^{-1}\mathbf{J}_{\vec{P}}\left(\vec{B}\right) \\ \mathbf{0}_3 & \mathbf{1}_3 \end{pmatrix} , \tag{2.23}$$

with

$$\mathbf{C} = \mathbf{1}_3 + \mathbf{J}_{\vec{P}}\left(\vec{E}\right) \,. \tag{2.24}$$

This 3×3 matrix can be inverted explicitly.

The finite differences scheme can be implemented on a grid, revealing properties that are evaluated in Section 2.3. An open source numerical solver to be introduced in Chapter 6 is responsible for the time integration of the system of ordinary differential equations (2.22).

2.3 Dispersion relations

2.3.1 A simple example at first and second order

To investigate dispersion effects in the finite differences approximation, an elementary example is worked out in the following. For the values $\nu \in \{-1, 0, 1\}$ the expansion of $g(x)$ in matrix form is given by

$$\begin{pmatrix} g(x-\Delta_x) \\ g(x) \\ g(x+\Delta_x) \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1/2 \\ 1 & 0 & 0 \\ 1 & 1 & 1/2 \end{pmatrix} \cdot \begin{pmatrix} g(x) \\ \Delta_x g'(x) \\ \Delta_x^2 g''(x) \end{pmatrix} . \tag{2.25}$$

It has to be noted that Equation (2.25) is not restrictive to one specific kind of finite difference method. To obtain an expression for the first derivative, the matrix in (2.25) has to be inverted. This leads to

$$\begin{pmatrix} g(x) \\ \Delta_x g'(x) \\ \Delta_x^2 g''(x) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ -1/2 & 0 & 1/2 \\ 1 & -2 & 1 \end{pmatrix} \cdot \begin{pmatrix} g(x-\Delta_x) \\ g(x) \\ g(x+\Delta_x) \end{pmatrix} . \tag{2.26}$$

Assuming that $g(x)$ is one of the two polarizations of an electromagnetic mode that propagates in positive (+) or negative (−) x-direction in 1D, the corresponding equation of motion reads

$$(\partial_t \pm c\,\partial_x)\, g_\pm(x;t) = 0 \, . \tag{2.27}$$

Making use of (2.26), Equation (2.27) becomes for symmetric forward and backward differentiation

$$\partial_t g_\pm(x;t) \pm \frac{c}{2\Delta_x}\left(g_\pm(x+\Delta_x;t) - g_\pm(x-\Delta_x;t)\right) = 0 \, , \tag{2.28}$$

where the spatial derivative of g is accurate up to second order in Δ_x. Equation (2.28) can be solved analytically by a plane wave ansatz

$$g_\pm(x;t) = e^{-i\omega t + i k_\pm x} \, , \tag{2.29}$$

where $k_+ > 0$ and $k_- < 0$, to obtain

$$-(\mathfrak{Re}(\omega) + i\,\mathfrak{Im}(\omega)) \pm \frac{c}{\Delta_x}\sin(k_\pm \Delta_x) = 0 \, , \tag{2.30}$$

$$\Rightarrow \omega = \pm\frac{c}{\Delta_x}\sin(k_\pm \Delta_x) \, . \tag{2.31}$$

The derivative in (2.27) can also be approximated for a forward-propagating mode g_+ by a first order forward finite difference. In this case, Equation (2.28) can be written as

$$\partial_t g_+(x;t) + \frac{c}{\Delta_x}\left(g_+(x+\Delta_x;t) - g_+(x;t)\right) = 0 \, , \tag{2.32}$$

where the solution of (2.32) with $\omega \in \mathbb{C}$ reads

$$-\omega + \frac{c}{\Delta_x}\left(\sin(k_+\Delta_x) + i - i\cos(k_+\Delta_x)\right) = 0 \, , \tag{2.33}$$

$$\Rightarrow \mathfrak{Re}(\omega) = \frac{c}{\Delta_x}\sin(k_+\Delta_x) \, , \tag{2.34}$$

$$\Rightarrow \mathfrak{Im}(\omega) = \frac{c}{\Delta_x}\left(1 - \cos(k_+\Delta_x)\right) \, . \tag{2.35}$$

Further, the derivative in (2.27) for the backward-propagating part can be approximated by a first order backward finite difference. In this case, Equation (2.28) can be approximated for the backward-propagating case as

$$\partial_t g_-(x;t) - \frac{c}{\Delta_x}\left(g_-(x;t) - g_-(x-\Delta_x;t)\right) = 0 \tag{2.36}$$

with the solution

$$-\omega - \frac{c}{\Delta_x}(\sin(k_-\Delta_x) - i + i\cos(k_-\Delta_x)) = 0 \,, \tag{2.37}$$

$$\Rightarrow \mathfrak{Re}(\omega) = -\frac{c}{\Delta_x}\sin(k_-\Delta_x) \,, \tag{2.38}$$

$$\Rightarrow \mathfrak{Im}(\omega) = \frac{c}{\Delta_x}(1 - \cos(k_-\Delta_x)) \,. \tag{2.39}$$

The dispersion relations connect a simulated wave with a specified wavelength to its phase velocity on the lattice. While in the case of symmetric differentiation there is no imaginary part of ω, it does emerge for biased differentiation. The derived dispersion relations are shown in the top row of Figure 2.1.

On a discretized space with the grid spacing given by Δ_x, modes can only be sampled up to a threshold frequency, where the wavelength is twice as long as the grid spacing. The threshold frequency is known as the Nyquist frequency [149]. Here it is given by

$$f_{\mathrm{Ny}} = \Delta_x^{-1}/2 \,, \tag{2.40}$$

corresponding to

$$k \cdot \Delta_x = \pi \,. \tag{2.41}$$

It marks the point where $\mathfrak{Re}(\omega) = 0$ for $k \neq 0$ in all cases. Implications of this barrier are further discussed in Section 2.3.3.

The imaginary part amplifies the wave and thus makes this scheme unstable. By differentiating biased against the propagation direction, however, the imaginary part switches signs and has a damping effect. The calculations are straightforward and the result is shown at the bottom of Figure 2.1. This behavior can be tuned with higher order schemes in order to obtain a more vacuum-like real part and at the same time to defer the imaginary part, such that the latter becomes relevant only for short wavelengths.

As the real part of ω eventually decreases in any scheme when it approaches the Nyquist frequency, the damping of high-frequency modes is an anti-aliasing effect, annihilating nonphysical modes. This is demonstrated in the following sections.

2.3.2 The fourth order scheme

The imbalanced coefficients for the first order discrete derivative in fourth order accuracy in Δ_x, $g'_{\nu\in\{-3,1\}}$ and $g'_{\nu\in\{-1,3\}}$ in the above notation, can be formulated with the help of Equation (2.16). This results in

$$\Delta_x\, g'_{\nu\in\{-3,1\}} = \begin{pmatrix} -1/12 & 1/2 & -3/2 & 5/6 & 1/4 \end{pmatrix} \begin{pmatrix} g(x-3\Delta_x) \\ g(x-2\Delta_x) \\ g(x-\Delta_x) \\ g(x) \\ g(x+\Delta_x) \end{pmatrix} \,, \tag{2.42}$$

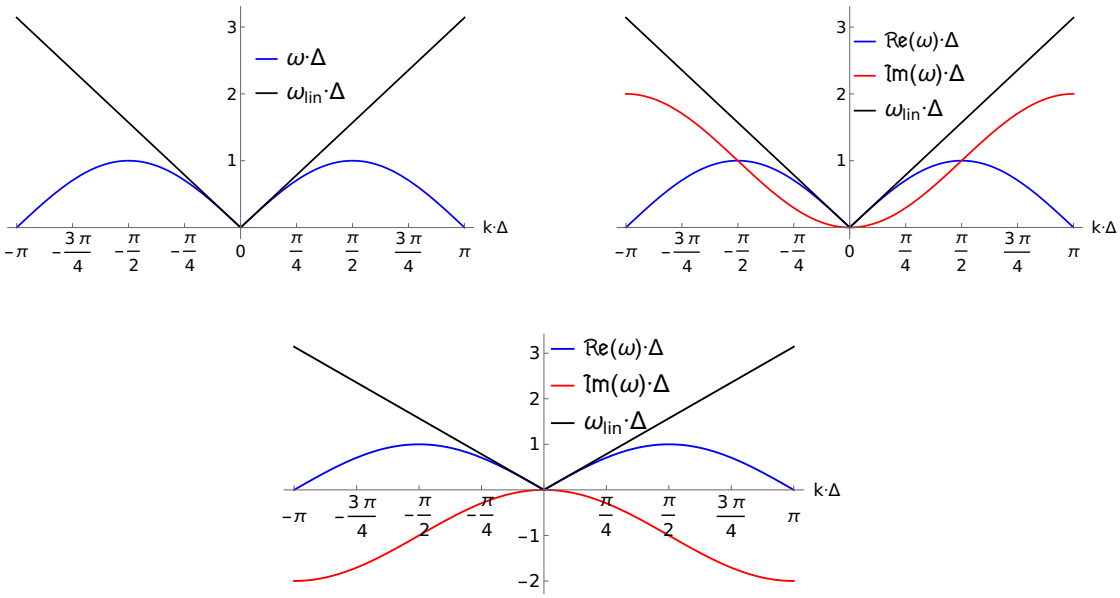


Figure 2.1: Dispersion relations obtained with low-order finite differences for a plane wave propagation. Here $c = 1$. *Top left*: for a second order symmetric forward and backward finite differences scheme. *Top right*: for a first order forward finite difference for the forward-propagating mode and a first order backward finite difference for the backward-propagating mode. *Bottom*: for a first order forward finite difference for the backward-propagating mode and a first order backward finite difference for the forward-propagating mode. The linear vacuum $\omega = c \cdot k$ is given by the black lines with the vacuum speed of light c set to unity. Note that all stencils result in a symmetric dispersion relation. The phase velocity is the same for both directions.

$$\Delta_x g'_{\nu\in\{-1,3\}} = \begin{pmatrix} -1/4 & -5/6 & 3/2 & -1/2 & 1/12 \end{pmatrix} \begin{pmatrix} g(x-\Delta_x) \\ g(x) \\ g(x+\Delta_x) \\ g(x+2\Delta_x) \\ g(x+3\Delta_x) \end{pmatrix} . \tag{2.43}$$

To make the connection to Equation (2.11), g is replaced by the components of $\mathbf{R}_x\vec{f}$. As explained above, $\mathbf{R}_x\vec{f}$ has components propagating in different directions that are differentiated with biases in the respective opposite direction. This implies that the first three components of $\mathbf{R}_x\vec{f}$ are differentiated in the same way as g in Equation (2.42) and the last three components of $\mathbf{R}_x\vec{f}$ are differentiated as g in Equation (2.43). Following these instructions for Equation (2.14) yields

$$\mathscr{D}_x\left(\mathbf{R}_x\vec{f}\right)(x,y,z) = \sum_{\nu=-3}^{3} \frac{1}{\Delta_x} S_\nu \left(\mathbf{R}_x\vec{f}\right)(x+\nu\Delta_x, y, z) \ , \tag{2.44}$$

with the fourth order stencils given by

$$\begin{aligned}
S^4_{+3} &= \mathrm{diag}(1/12, 1/12, 1/12, 0, 0, 0) \ , & S^4_{-3} &= \mathrm{diag}(0, 0, 0, -1/12, -1/12, -1/12) \ , \\
S^4_{+2} &= \mathrm{diag}(-1/2, -1/2, -1/2, 0, 0, 0) \ , & S^4_{-2} &= \mathrm{diag}(0, 0, 0, 1/2, 1/2, 1/2) \ , \\
S^4_{+1} &= \mathrm{diag}(3/2, 3/2, 3/2, 1/4, 1/4, 1/4) \ , & S^4_{-1} &= \mathrm{diag}(-1/4, -1/4, -1/4, -3/2, -3/2, -3/2) \ , \\
S^4_{0} &= \mathrm{diag}(-5/6, -5/6, -5/6, 5/6, 5/6, 5/6) \ .
\end{aligned} \tag{2.45}$$

In view of symmetry and redundancy, it is instructive to rewrite the stencil matrices in terms of its components applied to forward- and backward-propagating modes. The fourth order stencil matrices can be expressed as

$$S^4_\nu = \mathrm{diag}(s^4_b[\nu], s^4_b[\nu], s^4_b[\nu], s^4_f[\nu], s^4_f[\nu], s^4_f[\nu]) \ , \tag{2.46}$$

where the components applied to forward- and backward-propagating modes s^4_f and s^4_b are given by

$$\begin{aligned}
s^4_f\big|_{\nu=-3,\ldots,1} &= \left\{ -\frac{1}{12}, \frac{1}{2}, -\frac{3}{2}, \frac{5}{6}, \frac{1}{4} \right\} , \\
s^4_b\big|_{\nu=-1,\ldots,3} &= \left\{ -\frac{1}{4}, -\frac{5}{6}, \frac{3}{2}, -\frac{1}{2}, \frac{1}{12} \right\} ,
\end{aligned} \tag{2.47}$$

and $s^4_{f/b} = 0$ for out-of-range values of ν. Utilizing the obvious remaining symmetry, the stencil matrix can also be written as

$$S^4_\nu = \mathrm{diag}(-s^4_f[-\nu], -s^4_f[-\nu], -s^4_f[-\nu], s^4_f[\nu], s^4_f[\nu], s^4_f[\nu]) \ , \tag{2.48}$$

for the whole range $\nu = -3, \ldots, 3$.

The solver has implementations of the scheme up to order thirteen. In the ongoing investigations the currently maximal available accuracy is used. Besides the high accuracy, this has another advantage. Considering the fourth order scheme above, the imbalance between steps in the one and the other direction is three to one. In contrast, the thirteenth order facilitates to take seven steps of Δ_x forward and six steps backward in the forward-biased differentiation and vice versa in the backward-biased case. Hence, the imbalance can be kept much smaller. The components of the minimal-bias stencils up to order thirteen are listed in Section 2.3.3.

2.3.3 Dispersion properties of the scheme

Derivation of the dispersion relations

For a single plane wave the Heisenberg–Euler Lagrangian reduces to the Maxwell Lagrangian, since in this case $\mathscr{F} = \mathscr{G} = 0$ holds and there are no vacuum nonlinearities. Inserting a plane wave, without loss of generality propagating on the x-axis,

$$\vec{E}(x;t) = \vec{A}\, e^{-i(\omega t - kx)} \,, \tag{2.49}$$

into the propagation equation (2.22) yields

$$-i\omega \mathbf{1}_6 = \mathbf{Z}_x^{\text{lin}} \mathbf{R}_x^{\text{T}} \sum_\nu \frac{1}{\Delta_x} S_\nu e^{i\nu k\Delta_x} \mathbf{R}_x \,. \tag{2.50}$$

Since the stencil matrices are diagonal, this can be written as

$$-i\omega \mathbf{1}_6 = \mathbf{Z}_x^{\text{lin}} \mathbf{R}_x^{\text{T}} \mathbf{R}_x \frac{1}{\Delta_x} \sum_\nu S_\nu e^{i\nu k\Delta_x} \,, \tag{2.51}$$

which, when multiplied with $\mathbf{R}_x$ from the left, gives

$$-i\,\mathbf{R}_x\,\omega = \text{diag}(0,1,1,0,-1,-1)\mathbf{R}_x \frac{1}{\Delta_x} \sum_\nu S_\nu e^{i\nu k\Delta_x} \,. \tag{2.52}$$

Inserting the stencil matrices expressed in terms of their components applied to forward- and backward-propagating modes s_f and s_b results in

$$-i\,\mathbf{R}_x\,\omega = \text{diag}(0,1,1,0,-1,-1)\mathbf{R}_x \frac{1}{\Delta_x} \sum_\nu \text{diag}\left(s_b[\nu], s_b[\nu], s_b[\nu], s_f[\nu], s_f[\nu], s_f[\nu]\right) e^{i\nu k\Delta_x} \,. \tag{2.53}$$

It can be seen that the equation above holds irrespectively of the axis of propagation since $\mathbf{R}_x$ can be canceled. Hence, the equation becomes

$$\omega\Delta = i\,\text{diag}(0,1,1,0,-1,-1) \sum_\nu \text{diag}\left(s_b[\nu], s_b[\nu], s_b[\nu], s_f[\nu], s_f[\nu], s_f[\nu]\right) e^{i\nu k\Delta} \,. \tag{2.54}$$

There are the distinct cases of forward- and backward-propagating modes, and non-propagating ones. It is obtained for

- a forward-propagating mode (implying $k > 0$):

$$\omega\Delta = \sum_\nu s_f[\nu](-i\cos(\nu k\Delta) + \sin(\nu k\Delta)) \,; \tag{2.55}$$

- a backward-propagating mode (implying $k < 0$):

$$\omega\Delta = \sum_\nu s_b[\nu](i\cos(\nu k\Delta) - \sin(\nu k\Delta)) = \sum_\nu s_f[-\nu](-i\cos(\nu k\Delta) + \sin(\nu k\Delta)) \,; \tag{2.56}$$

  and

- a non-propagating mode (implying $k = 0$) :

$$\omega = 0 \,. \tag{2.57}$$

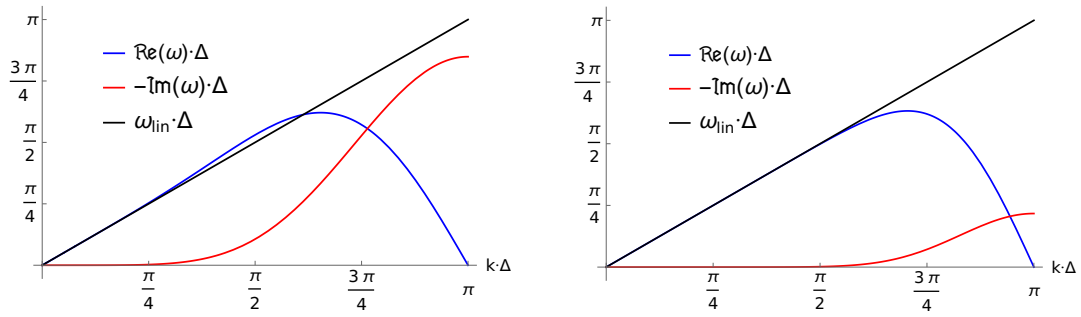


Figure 2.2: Dispersion relations of the numerical scheme for minimally biased finite differences at order four (*left*) and order thirteen (*right*). The black lines represent the real vacuum dispersion relation with $c = 1$. Δ is the grid spacing, the physical distance between lattice points. For better visibility of the symmetric plots only the values for $k \geq 0$ are shown.

At the end of this section the stencil components up to order thirteen are listed and plots of the dispersion relations are shown. The results for orders four and thirteen are visualized and explained in Figure 2.2; the latter being the order used for all simulations in this work, where it is not mentioned otherwise. The minimal resolvable distance Δ is the spacing between the lattice points. It is the decisive parameter to tune the dispersion effects for a given wavelength and is given by the ratio of physical length and lattice points.

The real parts of ω start in the vicinity of the black line and deviate from it for shorter wavelengths or smaller grid resolutions. There is less deviation at higher orders. The imaginary part of ω starts with values close to zero for large wavelengths/high grid resolutions and decreases until the Nyquist frequency is reached. The negative imaginary part in this scheme has the positive effect to promptly annihilate those modes that deviate strongly from the vacuum dispersion relation.

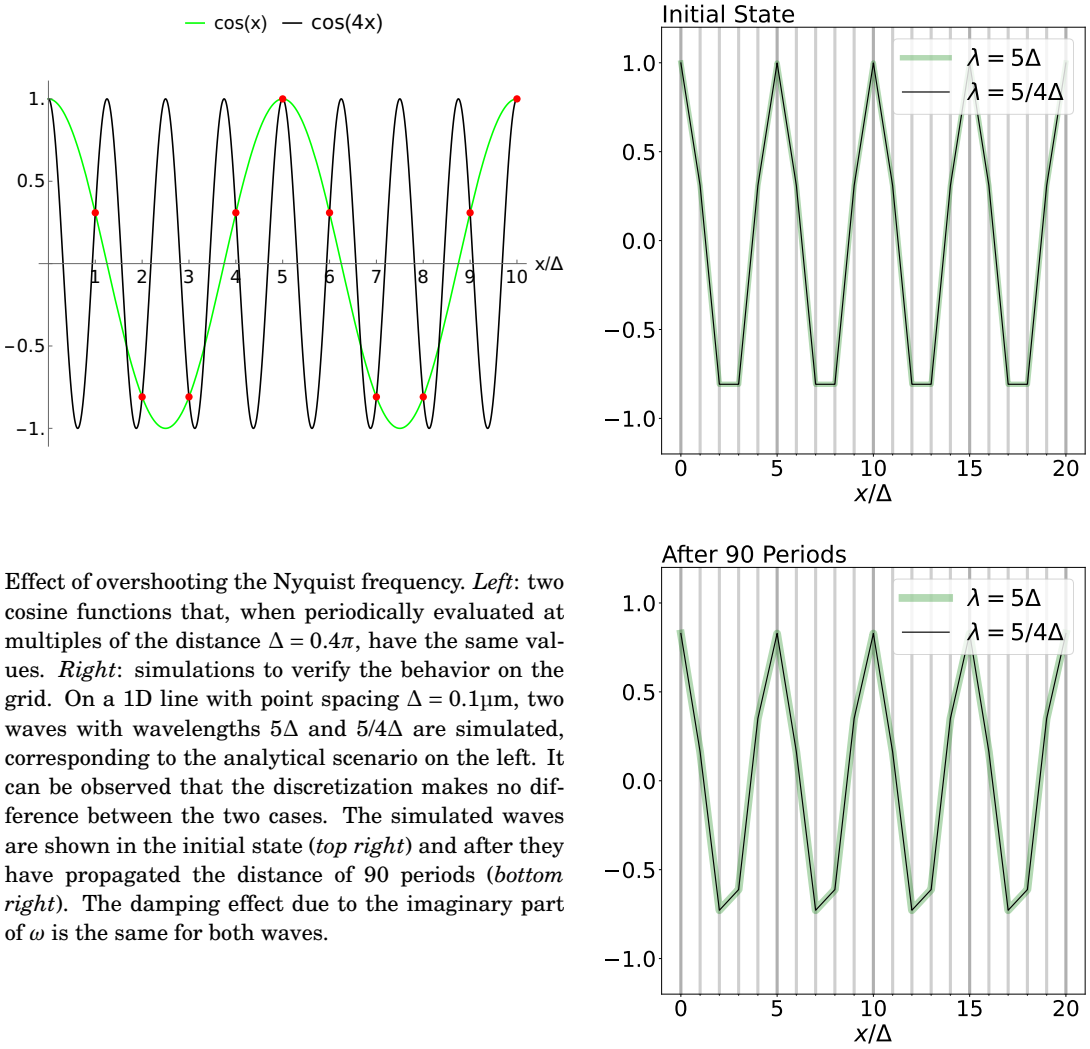
At higher orders, the damping effect of the imaginary part is deferred to shorter wavelengths and is overall smaller. It can further be seen that for higher orders the real part stays closer to the linear vacuum for smaller wavelengths. A relatively strong bias as in the fourth order scheme, (2.42) and (2.43), causes a superluminal phase-velocity in a given $k \cdot \Delta$ range, a critical regime where the dispersion relation deviates remarkably from the linear vacuum. The well-balanced order thirteen scheme has a smaller critical regime and is well-behaved for a larger range of wavelengths.

For small wavelengths the curve showing the real part of ω falls off and the imaginary part of ω causes a damping. As a result, there are two k's for each $\mathfrak{Re}(\omega)$. At the wavelength corresponding to the Nyquist frequency, $k \cdot \Delta = \pi$, $\mathfrak{Re}(\omega)$ is zero, as in the simple scenarios visualized in Figure 2.1.

Stability of the scheme

The high-k values cause nonphysical standing waves with $\mathfrak{Re}(\omega)/k \to 0$. This would cause a self-heating of the system if the damping did not take care of the quick annihilation of these modes. The superluminosity that can be seen in the fourth order finite difference differentiation is thus an acceptable deviation of the real vacuum dispersion to the favor of a damping of nonphysical modes. By increasing the accuracy further and thereby generating a better balance, as in the order thirteen scheme, there is no superluminal range left. Yet, the nonphysical modes filter is still active, and in this way higher order schemes bear clear benefits.

In the numerical investigation it is demonstrated that the damping has noticeable effects already at an earlier stage than a first look at the above plot would suggest. Note that the Nyquist



Effect of overshooting the Nyquist frequency. *Left*: two cosine functions that, when periodically evaluated at multiples of the distance $\Delta = 0.4\pi$, have the same values. *Right*: simulations to verify the behavior on the grid. On a 1D line with point spacing $\Delta = 0.1\mu$m, two waves with wavelengths 5Δ and $5/4\Delta$ are simulated, corresponding to the analytical scenario on the left. It can be observed that the discretization makes no difference between the two cases. The simulated waves are shown in the initial state (*top right*) and after they have propagated the distance of 90 periods (*bottom right*). The damping effect due to the imaginary part of ω is the same for both waves.

Figure 2.3: On the Nyquist frequency.

frequency (2.40), on the other hand, is not the limiting factor for wave modeling in this scheme. In order to accurately time-evolve modes, the grid spacing must be chosen such that $f \ll f_{\mathrm{Ny}}$, c.f. Equation 2.40. A stricter quantification is given below.

Beneficially, the simulation of a high-frequency wave does not overshoot the Nyquist limit, as it would then be sampled on the grid as a wave with lower frequency. This is explicitly demonstrated in Figure 2.3. The relevant frequency scale of the dispersion relation in Figure 2.2 thus ranges only to the Nyquist limit and the scheme is generally stable for any frequency.

Damping of modes

It has to be noted that energy is not conserved on the grid as a consequence of the amplitude damping during the propagation. The imaginary part of ω and with it the damping effect might be small if the grid resolution is high, but there is a trade-off since high grid resolutions are computationally more expensive.

To visualize the effects of the dispersion relation at varying wavelengths the propagation of a

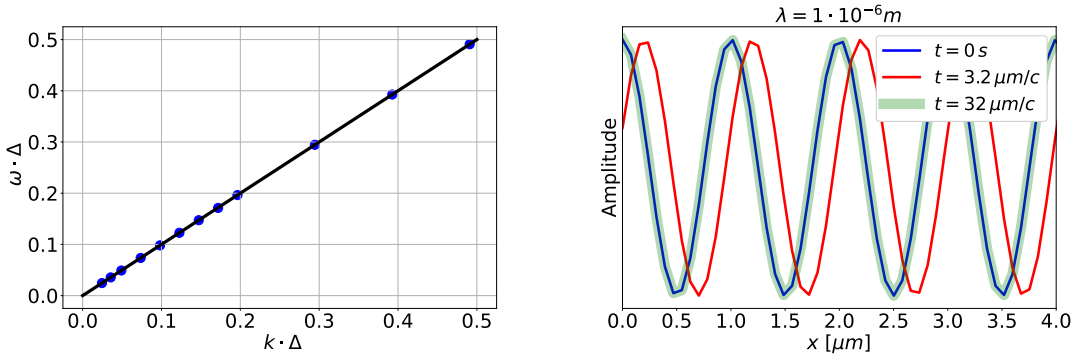


Figure 2.4: Numerical tests of the dispersion for low frequencies. *Left*: numerical results of dispersion relation tests. Simulation results (*blue dots*) are in agreement with the free vacuum (*black line*) for wavelengths from 20 µm to 1 µm. *Right*: the plane wave corresponding to the rightmost point of the left figure with $\lambda = 1\,\mu m$ at different time steps. The wave is not that smoothly modeled anymore but still does not noticeably lose energy in the given amount of time – the amplitude stays the same. Even after 32 periods the wave is perfectly overlapping with its initial state.

plane wave in one direction of a two-dimensional square grid with side length 80 µm, divided into 1024×1024 points, is investigated. The grid resolution is thus given by $\Delta = 80\,\mu m/1024$, $\Delta^{-1} = 128 \times 10^5\,m^{-1}$.

From Figure 2.4 it can be deduced that the waves are well-behaved and quite well-modeled for $k \cdot \Delta \lesssim 0.5$. From Figure 2.5, it can be inferred that already at half the Nyquist frequency the damping is non-negligible for relevant time scales. The Nyquist frequency in this scenario is given by $f_{Ny} = 64 \times 10^5\,m^{-1}$ ($\lambda = 1.5625 \times 10^{-7}\,m$).

A video demonstrating the evolution of a plane wave on the lattice with a wavelength corresponding to half of the Nyquist frequency can be found in the *Mendeley Data* repository [150]. The conclusion is that for a proper modeling and to avert damping effects, one is obliged to adapt the grid resolutions to the lowest wavelength such that

$$\Delta \lesssim 1/12\,\lambda \tag{2.58}$$

for the order thirteen scheme.

While this relation is not a hard limit and can be relaxed in many cases without detriments, it poses a safe rule of thumb for long-time simulations. A sufficiently fine grid resolution is pertinent for a clean analysis of the polarization flipping and harmonic generation effects presented in Chapter 4. A more detailed analysis of the accuracy is provided in Chapter 6.

Since the external ODE solver introduced in Chapter 6 takes care of the time integration with high accuracy, the size of a time step can be defined quite openly. In this work a time step varies in the range of 1 fs to 6 fs, or equivalently 0.3 µm/c to 2 µm/c .

Stencils and dispersion relations up to order thirteen

Solutions to the dispersion relation for all given orders are visualized in Figure 2.6. As already mentioned in the caption of Figure 2.2, the dispersion relation is symmetric in k for all stencils and the phase velocity is equal in both directions. Higher orders defer and decrease the rise of the imag-

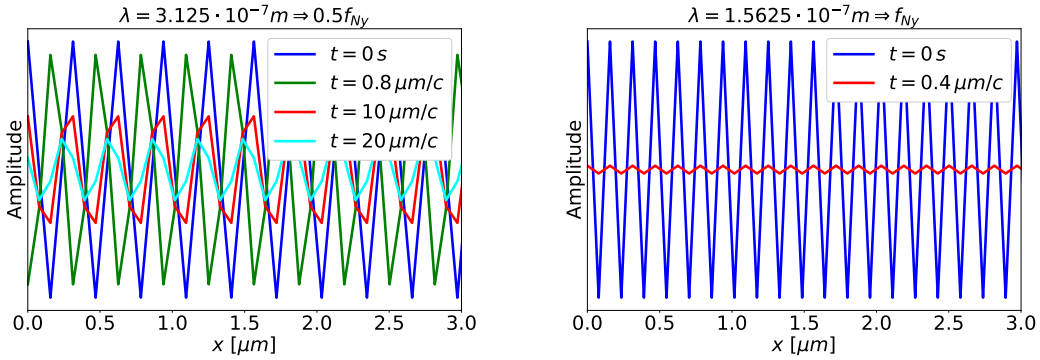


Figure 2.5: Numerical tests of the dispersion for high frequencies. *Left*: a plane wave with $\lambda = 312.5\,\text{nm}$ at different time steps. The grid resolution is given by $\Delta = 80\,\mu\text{m}/1024$ in the propagation direction. In the free vacuum the corresponding frequency is to be associated to half of the Nyquist frequency. 32 periods have passed after $10\,\mu\text{m/c}$ and 64 periods after $20\,\mu\text{m/c}$. The damping and the barely subluminal phase velocity are, after the indicated propagation times, noticeable even at this wavelength. The initial peak position has clearly shifted after $20\,\mu\text{m/c}$. *Right*: a plane wave with $\lambda = 156.25\,\text{nm}$ at different time steps with the same grid settings. In the free vacuum the corresponding frequency is to be associated to the Nyquist frequency. The wave is standing and the damping is strong, resulting in a rapid annihilation.

inary part. A stronger bias causes more absorption and the phase velocity to become superluminal. This can be seen in Figure 2.6 for the even orders of accuracy that are more imbalanced.

For completeness, the elements of the minimally biased stencil matrices S_ν from order one to thirteen, listed in the form of the array components as in Equation (2.47) for the fourth order, are given by

$$
\begin{aligned}
s_f^1|_{\nu=-1,0} &= \{-1, 1\}\,, \\
s_f^2|_{\nu=-2,-1,0} &= \left\{\frac{1}{2}, -2, \frac{3}{2}\right\}\,, \\
s_f^3|_{\nu=-2,\ldots,1} &= \left\{\frac{1}{6}, -1, \frac{1}{2}, \frac{1}{3}\right\}\,, \\
s_f^4|_{\nu=-3,\ldots,1} &= \left\{-\frac{1}{12}, \frac{1}{2}, -\frac{3}{2}, \frac{5}{6}, \frac{1}{4}\right\}\,, \\
s_f^5|_{\nu=-3,\ldots,2} &= \left\{-\frac{1}{30}, \frac{1}{4}, -1, \frac{1}{3}, \frac{1}{2}, -\frac{1}{20}\right\}\,, \\
s_f^6|_{\nu=-4,\ldots,2} &= \left\{\frac{1}{60}, -\frac{2}{15}, \frac{1}{2}, -\frac{4}{3}, \frac{7}{12}, \frac{2}{5}, -\frac{1}{30}\right\}\,, \\
s_f^7|_{\nu=-4,\ldots,3} &= \left\{\frac{1}{140}, -\frac{1}{15}, \frac{3}{10}, -1, \frac{1}{4}, \frac{3}{5}, -\frac{1}{10}, \frac{1}{105}\right\}\,, \\
s_f^8|_{\nu=-5,\ldots,3} &= \left\{-\frac{1}{280}, \frac{1}{28}, -\frac{1}{6}, \frac{1}{2}, -\frac{5}{4}, \frac{9}{20}, \frac{1}{2}, -\frac{1}{14}, \frac{1}{168}\right\}\,, \\
s_f^9|_{\nu=-5,\ldots,4} &= \left\{-\frac{1}{630}, \frac{1}{56}, -\frac{2}{21}, \frac{1}{3}, -1, \frac{1}{5}, \frac{2}{3}, -\frac{1}{7}, \frac{1}{42}, -\frac{1}{504}\right\}\,, \\
s_f^{10}|_{\nu=-6,\ldots,4} &= \left\{\frac{1}{1260}, -\frac{1}{105}, \frac{3}{56}, -\frac{4}{21}, \frac{1}{2}, -\frac{6}{5}, \frac{11}{30}, \frac{4}{7}, -\frac{3}{28}, \frac{1}{63}, -\frac{1}{840}\right\}\,, \\
s_f^{11}|_{\nu=-6,\ldots,5} &= \left\{\frac{1}{2772}, -\frac{1}{210}, \frac{5}{168}, -\frac{5}{42}, \frac{5}{14}, -1, \frac{1}{6}, \frac{5}{7}, -\frac{5}{28}, \frac{5}{126}, -\frac{1}{168}, \frac{1}{2310}\right\}\,,
\end{aligned}
$$

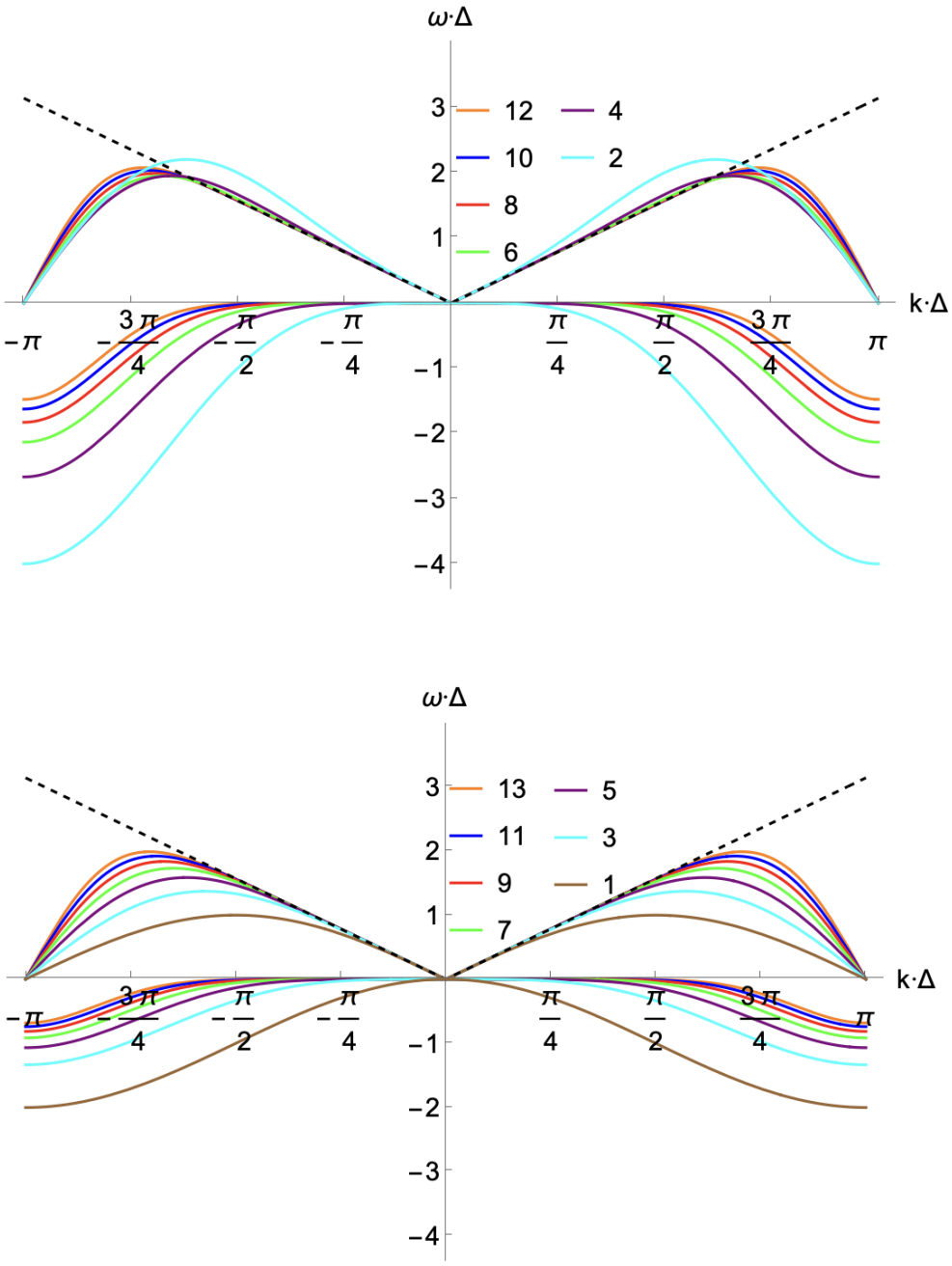


Figure 2.6: Dispersion relations up to order thirteen. Real and imaginary parts are shown in the same color and can be distinguished by their form. The black dotted line represents the real vacuum dispersion relation. *Top*: even orders. *Bottom*: uneven orders.

$$s_f^{12}\big|_{v=-7,\dots,5} = \left\{-\frac{1}{5544}, \frac{1}{396}, -\frac{1}{60}, \frac{5}{72}, -\frac{5}{24}, \frac{1}{2}, -\frac{7}{6}, \frac{13}{42}, \frac{5}{8}, -\frac{5}{36}, \frac{1}{36}, -\frac{1}{264}, \frac{1}{3960}\right\},$$
$$s_f^{13}\big|_{v=-7,\dots,6} = \left\{-\frac{1}{12012}, \frac{1}{792}, -\frac{1}{110}, \frac{1}{24}, -\frac{5}{36}, \frac{3}{8}, -1, \frac{1}{7}, \frac{3}{4}, -\frac{5}{24}, \frac{1}{18}, -\frac{1}{88}, \frac{1}{660}, -\frac{1}{10296}\right\}.$$

If centered (unbiased) stencils were used at even order, that would result in ω having no imaginary part, as discussed in Section 2.3.1. A *Mathematica* [151] analysis of the dispersion relations can be found in [152].

2.4 Overview of the implementation

Simulations are performed on a lattice, which is subdivided into patches that communicate their boundary values to each other. The width of the boundaries are determined by the order of the finite differences scheme, i.e., the neighboring lattice points required for the calculation of spatial derivative approximations. The regions on the patches containing the boundary values will be interchangeably called ghost cells or halos throughout this work.

The algorithm is implemented in *C++*. On a computer, each patch is processed by one core of a processing unit. This approach enables the scalability on high-performance computing systems. The lattice decomposition and the cluster computer communication is detailed in Section 5.3.1.

The lattice forms a class. A member vector of the `Lattice` class is holding the field values $\vec{f}$ of Equation (2.3) for all lattice points. Another class, the `Simulation` class, passes on the construction parameters for the lattice and instructions for other configuration settings at the start of the simulation. This class is responsible for the overall coordination, see Listing 2.1.

```
// Initialize the simulation, set up the cartesian communicator
Simulation sim(patches[0], patches[1], patches[2],
               StencilOrder, periodic); // Simulation object with slicing

// Create the SUNContext object associated with the thread of execution
sim.setPhysicalDimensionsOfLattice(phys_dims[0], phys_dims[1],
                                   phys_dims[2]); // spacing of the box
sim.setDiscreteDimensionsOfLattice(
    disc_dims[0], disc_dims[1],
    disc_dims[2]); // Spacing equivalence to points
sim.initializePatchwork(patches[0], patches[1], patches[2]);
//sim.initializeGhostCells();

// Add em-waves
for (const auto &plane : planes)
  sim.icsettings.addPlaneWave3D(plane.k, plane.p, plane.phi);
for (const auto &gauss : gaussians)
  sim.icsettings.addGauss3D(gauss.x0, gauss.axis, gauss.amp, gauss.phip,
                            gauss.w0, gauss.zr, gauss.ph0, gauss.phA);

// Check that the patchwork is ready and set the initial conditions
sim.start();

// Initialize CVode with abs and rel tolerances
sim.initializeCVODEobject(CVodeTol[0], CVodeTol[1]);

// Configure the time evolution function
TimeEvolution::c = interactions;
```

```
29  TimeEvolution::TimeEvolver = nonlinear3DProp;
30
31  // Configure the output
32  sim.outputManager.generateOutputFolder(outputDirectory);
33  if (!myPrc) {
34    std::cout << "Simulation code: " << sim.outputManager.getSimCode()
35        << std::endl;
36  }
37  sim.outputManager.set_outputStyle(outputStyle);
38
39  // Conduct the propagation in space and time
40  for (int step = 1; step <= numberOfSteps; step++) {
41    sim.advanceToTime(endTime / numberOfSteps * step);
42    if (step % outputStep == 0) {
43      sim.outAllFieldData(step);}
```

Listing 2.1: Excerpt of the newly created coordinating *C++* file. The Simulation object is responsible for all steps and the overall configuration. First, the physical dimensions of the lattice and the resolution in terms of lattice points are defined. Subsequently, the number of patches relevant for parallelization are set, discussed in Chapter 5. Then, the electromagnetic waves are added. With the start of the simulation the settings of the solver are configured and the folder for data output is generated. Finally the waves are propagated and the desired states written to disk.

The initial conditions of the electromagnetic fields are formed by the parameters of the waves the user decides to simulate. The latter are objects of the various electromagnetic wave classes with individual member functions to initialize the six-dimensional field data vector. Another class, called `ICSetter`, is responsible to fill the lattice data vector with the initial field conditions via a loop evaluating the electromagnetic field components on all lattice points.

An auxiliary vector is thereby created for the operations of the scheme discussed in Section 2.2. The auxiliary vector obtains the rotated field data, c.f. Equation (2.12), that are used for the derivative operation with the help of the finite difference scheme at the chosen order, c.f. Equation (2.13). The result is rotated back and stored in a vector holding the data in spatially derived form as in Equation 2.21.

This is finally used in a propagation function that makes use of the outlined scheme to obtain Equation (2.22) with the temporally derived field data on the left, which is contained in a further vector. The time integration is then performed with an external software to be detailed in Chapter 6.

Directed by the settings passed via the `Simulation` class, the `OutputManager` class takes care of writing data to disk. The solver as a software is outlined in Chapter 3, a code reference generated with the help of *Doxygen* [153] is available in the code repository [148].

2.5 Comparison to other approaches

There exist other approaches to simulate all-optical QED vacuum effects. Currently, there are two other groups working in the field of numerical approaches to the QED nonlinear vacuum, one in Jena and one in Lisbon.

The group in Jena makes use of the so-called *Vacuum Emission Picture* [154] as the basis for its algorithm. In this framework the effect-generating laser pulses are propagated by a linear Maxwell solver. Via a scattering amplitude, the vacuum subjected to the lasers forms a source term for outgoing photons that carry signals of nonlinear interactions. Back-reactions into the laser pulses are neglected.

The group in Lisbon constructed a generalized Yee scheme [155] of second order accuracy in space and time. Its dispersion relation is not overall stable.

Only the first order of the weak-field expansion is included in these solvers.

2.5.1 The Vacuum Emission Picture

The algorithm outlined in [156] and the solver presented in [143] are based on the Vacuum Emission Picture. This approach differs substantially from the one presented in the previous sections and there are vast differences in capabilities.

In the Vacuum Emission Picture the fields are split into strong background and weak signal fields $F^{\mu\nu} \to F^{\mu\nu} + f^{\mu\nu}$. The Heisenberg–Euler Lagrangian is expanded to the four-photon order in the weak-field expansion (1.20a) and is subsequently expanded to linear order in the signal field strength,

$$\mathscr{L}_{\mathrm{HE},4\gamma} \approx \frac{\partial \mathscr{L}_{\mathrm{HE},4\gamma}}{\partial F^{\mu\nu}} f^{\mu\nu} \,, \tag{2.59}$$

neglecting terms with more than one signal photon.

The core feature of the Vacuum Emission Picture is the use of transition matrices in the presence of strong fields in the Heisenberg–Euler weak-field approximation. The scattering amplitude maps between the vacuum state $|0\rangle$ and a signal photon state $|\gamma\rangle$ with the interaction Lagrangian (2.59),

$$S = \langle\gamma| \int d^4x \, \frac{\partial \mathscr{L}_{\mathrm{HE},4\gamma}}{\partial F^{\mu\nu}} f^{\mu\nu} |0\rangle \,. \tag{2.60}$$

This approach is very efficient and allows the analysis of experimentally realistic configurations. It provides direct access to relevant quantities such as polarization- and energy-resolved differential numbers of signal photons to be detected outside the interaction volume.

With regard to a numerical implementation, one of the strengths of this formalism is that it essentially only requires the evaluation of Fourier transforms for the calculation of the scattering amplitude. A fast Fourier transform analysis is the essential ingredient of the algorithm. This also requires a grid spacing adapted to the frequencies of the involved electromagnetic fields.

In the Vacuum Emission Picture pump and probe lasers are viewed as macroscopic classical electromagnetic fields whose propagation is governed by the linear Maxwell equations. The strong electromagnetic fields driving the nonlinear effects are thus propagated in the absence of vacuum nonlinearities by means of a Maxwell solver detailed in [157]. The vacuum subjected to the strong electromagnetic fields is interpreted as a source term for outgoing photons. Using the Heisenberg–Euler Lagrangian in weak-field expansion to describe the interaction, the scattering amplitude

(2.60) is obtained. From this perspective, the induced photon states correspond to the signal photons imprinted by the effective nonlinear interaction.

The effects of quantum vacuum nonlinearities are thus encoded in the signal photons emitted from the strong-field region and thereby constitute a distinct signal. The picture can hence be interpreted as describing laser-stimulated signal photon emission from the vacuum. All information about the asymptotic signal photon is carried in the vacuum emission amplitude. In certain configurations excellent signal to background separation can be achieved with the induced photons emitted from the strong-field region to be detected in the field-free region.

The emission of signal photons from a single pulse and colliding pulses are investigated in [143]. Studies of specific laser pulse collision processes with stimulated single photon emission from the vacuum are conducted. Signal photon number estimations can be performed and from the properties of the former conclusions can be drawn on polarization flipping probabilities.

However, it does not permit the investigation of time-dependent phenomena. There is further no feedback on the effect-generating electromagnetic fields themselves by the response of the quantum vacuum taken into account. Hence, there is no notion of pump and probe laser fields in contrast to the scenarios considered throughout the present work.

The Vacuum Emission Picture is capable of making predictions for asymptotic states of ultrashort emission wavelengths, while the solver outlined in Chapter 3 is limited by the affordable grid resolution. In that sense, the Vacuum Emission Picture represents an analytical scale separation at the expense of losing some of the processes, but with the advantage to predict short wavelength photon emission with sufficient accuracy.

Simulations of high-frequency pulses are currently not possible with the Heisenberg–Euler solver detailed in the present work owing to the limitations of the grid. In the outlook in Chapter 7 a plan for the future is outlined to render high-frequency pulses feasible. Relying on the numerical integration of scattering amplitudes, the Vacuum Emission Picture solver on the other hand is optimally suited for this task. As presented in [143], the solver is not yet equipped with a parallelized implementation for distributed computing.

Numerous analytical works are based on the Vacuum Emission Picture [16, 23, 83, 85, 92, 158–162]. In addition, the solver based on the Vacuum Emission Picture has already been employed in some studies of optical signatures of the quantum vacuum [72, 163, 164].

Besides vacuum birefringence, the approach has also been employed to photon–photon scattering as well as merging and splitting processes. The emission process is not restricted to cubic order in the background field [102, 103]. Hence, multi-photon emission processes can also be incorporated in the description.

2.5.2 Modified Yee scheme

As the scheme outlined in Section 2.3, the *QED vacuum polarization solver* of [165] likewise relies directly on the modified Maxwell equations due to the Heisenberg–Euler weak-field expansion, c.f. Section 1.6.3. It makes use of a finite-difference time-domain generalized Yee scheme of second order accuracy in space and time, a modification of the standard algorithm to solve Maxwell's equations [166]. As opposed to that, the accuracy order of the numerical scheme discussed in the previous sections is arbitrary and implementations from the first to the thirteenth order are available, see Chapter 3.

The main problem with the Yee scheme approach is that due to staggering the nonlinearities

at every point require interpolations in space and extrapolations in time [145]. The alternative to staggered fields in the modified scheme is to evaluate all fields at every position to accurately compute the electromagnetic field invariants in the presence of nonlinear couplings. Moreover, the nonlinear propagation of the electric field according to the modified Ampère law (1.25) requires the knowledge of future quantities. In an effort to mainly address the latter problem, a method alike the predictor-corrector methods for integration discussed in Chapter 6, is developed.

In the modified Yee scheme put forward in [165], the Yee scheme is first applied to the linear vacuum to advance the fields and obtain predicted values for the next time step. It is then made use of interpolation methods over all grid points to enable the calculation of the required electromagnetic invariants to be used in the modified Ampère law (1.25) in order to propagate the electric field under the influence of vacuum nonlinearities. The electric field values are updated iteratively, reinserting them into the expressions of the electromagnetic field invariants until the desired accuracy is reached. After convergence, Faraday's law is used to advance the magnetic field. In order to overcome precision asymmetries with respect to field invariants in the Yee grid cells, the fields are evaluated at the positions of higher precision with the help of interpolations. Additional re-interpolations are carried out to compute the invariants at the other grid cell points.

The dispersion relations of the numerical scheme of [165] and those outlined in Section 2.3.3 are fundamentally disparate. The dispersion relation outlined in [165] has an imaginary part that can lead to the amplification of nonphysical modes and requires a high grid resolution for a given wavelength. This might be the reason why results have been presented only in one and two spatial dimensions.

The dispersion relations shown in Figure 2.6 have imaginary parts that always damp nonphysical modes and can afford lower grid resolution at high integration orders. As per the discussed properties of the dispersion relation the solver of the present work can be considered very efficient. Moreover, Chapter 5 details that the implementation is well scalable on cluster computers.

The authors of [165] state that their solver is numerically stable in spite of the amplifying imaginary part. To this end, the grid has to be configured with an ample resolution. The scheme discussed in Section 2.3 is in contrast overall numerically stable and by virtue of higher orders the accuracy is kept for lower grid resolutions in relation to the simulated wavelengths. While in [165] simulations with the small spatial grid spacings of $k \cdot \Delta = \pi/100$ are performed, in the present work $k \cdot \Delta = \pi/6$ is sufficient. The authors of [165] nevertheless believe that their approach is computationally more efficient than the one presented in this work.

Due to the fact that the simulations in [165] are performed on an HPC system and that the authors claim the possibility of massively parallel simulations, it can be supposed that the implementation is scalable as well. However, for 1D and 2D simulations, which they present, no supercomputer would be required for the solver of the present work – resource occupation is taken up in Chapter 3.

In [165] basically similar simulations as conducted in Chapter 4 are presented. In the future the authors plan to simulate higher order processes and 3D simulations as well.

First, the authors check the refractive indices in 1D in an eternal background. Different theoretical 1D setups for birefringence in pulse collisions are investigated. Harmonic generation is simulated in 2D with coaxial and orthogonal pulses, yet only with four-photon processes, which marks the main difference to the simulations of Chapter 4 in the end result. Remarkably, they calculate up to the fifth harmonic analytically and as laser experts the authors discuss real world

experimental influences on the pulse parameters, such as laser misalignment.

It is made use of the same technique as in Chapter 4 of artificially increasing the field strengths in comparison to analytical benchmarks for simulations of vacuum birefringence. The use of high field strengths is beneficial to resolve the nonlinear effects from the numerical noise.

2.5.3 Conclusion

At the bottom line, it can be concluded that the Heisenberg–Euler solver is principally capable of capturing more interesting physics in comparison to the VEP solver, while it still has practical resolution limits that render some important simulations in the high-frequency realm unfeasible.

The VEP is not dynamic but benefits from an analytical separation of scales. Hence, it is able to make predictions for the probability of photon emission for ultra short wavelengths, while it lacks the nonlinear physics of interacting power pulses.

The Heisenberg–Euler solver in the weak-field approximation presented in Chapter 3 can principally describe all conceivable interaction scenarios between the probe and power pulses. For a complete picture the full nonlinear dynamics of the vacuum is required.

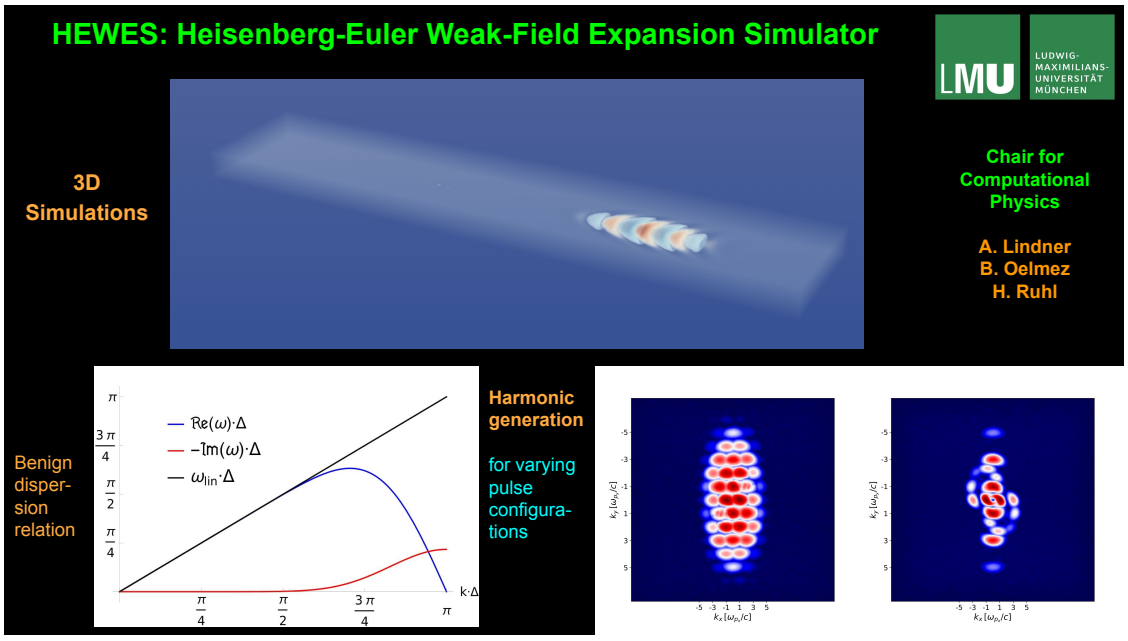
The solvers in [165] and [143] both include only the four-photon order of the weak-field expansion of the Heisenberg–Euler interaction, while the present work outlines an implementation for up to six-photon interactions. Some all-optical nonlinear vacuum effects are tiny. Hence, high-order, high-precision numerical solvers may bear advantages.

High-dimensional and high-resolution grids are subject to the curse of dimensionality. Some applications require the modeling of high-frequency waves and therefore extraordinary high resolutions. These easily reach beyond the limits of any computing system and thus extrapolation techniques have to be employed. It is therefore beneficial to rely on an algorithm with a very vacuum-like dispersion relation for high frequencies to minimize the required grid size. The beneficial dispersion properties outlined in Section 2.3 form the strongest contrast to the QED vacuum polarization solver of [165].

An implementation for distributed computing systems of the scheme discussed in the present work is at hand and allows costly simulations in full three spatial dimensions plus time. Such have not yet been presented with the help of the QED vacuum polarization solver of [165].

Chapter 3

HEWES: The Heisenberg–Euler Weak-Field Expansion Simulator



Graphical abstract of **HEWES: Heisenberg–Euler weak-field expansion simulator** [148]

Links

***Software Impacts* paper**
https://doi.org/10.1016/j.simpa.2023.100481

***Papers With Code* paper**
https://physics.paperswithcode.com/paper/hewes-heisenberg-euler-weak-field-expansion

***Software Impacts* code repository**
https://github.com/SoftwareImpacts/SIMPAC-2023-33

Code metadata

Current code version	v0.2.5
Link to code/repository	https://gitlab.physik.uni-muenchen.de/ls-ruhl/hewes
Link to Reproducible Capsule	https://codeocean.com/capsule/3187285/tree
Legal Code License	BSD 3-Clause License
Code versioning system used	git
Software code languages, tools, and services used	C++20, (MPI-3.1, OpenMP v4.5)
Compilation requirements, operating environments & dependencies	CMake ≥ v3.21
Link to code reference	https://gitlab.physik.uni-muenchen.de/ls-ruhl/hewes/-/blob/main/README.md https://gitlab.physik.uni-muenchen.de/ls-ruhl/hewes/-/blob/main/docs/ref.pdf
Support email for questions	and.lindner@physik.uni-muenchen.de

Table 3.1: Code metadata

3.1 Introduction

This chapter presents the solver as an open source software. It constitutes a highly accurate numerical solver for the leading weak-field Heisenberg–Euler corrections to classical Maxwell theory arising as a consequence of quantum vacuum polarization. The solver is named *HEWES*, for "Heisenberg–Euler Weak-Field Expansion Simulator".

The solver relies on the algorithm put forward in Chapter 2. A short recapitulation is in order. It is equipped with a high-order finite differences scheme for the approximation of spatial derivatives on a lattice. The modified Maxwell equations, partial differential equations (PDEs) in space and time, are thereby turned into ordinary differential equations (ODEs) in time. Time integration is performed by an external solver package.

The present code implementation reaches an accuracy of up to order thirteen in the numerical scheme and notably takes into account up to six-photon interactions, c.f. Section 1.6.2 with Figure 1.4. Depending on the chosen order of the numerical scheme, it possesses an almost linear vacuum dispersion relation even for smaller wavelengths. This allows the use of comparatively small grids. Moreover, an imaginary part in the dispersion relation annihilates nonphysical modes, see Section 2.3 with Figures 2.2 and 2.6.

Since theoretical approaches are limited to approximations and manageable configurations, and the experimental requirements for the detection of these signals are high, the need for support from the numerical perspective is apparent. In preference to analytical calculations, the numerical solver is agnostic to the specific scenarios and thus in principle superior in most situations. It has other practical limitations that have been discussed in Sections 1.6 and 2.5. Shifting these limits is part of the work detailed in Chapter 5 and future research outlined in Chapter 7.

The simulator is envisaged to become a tool seconding experimentalists efforts with simulation data. With the working and established simulator at hand, expedient simulations can be conducted in the future.

The code works in one to three spatial dimensions. Whereas 1D and 2D results can be easily attained on personal computers, the numerical load for meaningful full-scope 3D simulations requires the use of supercomputers. The code is therefore tuned for high-performance computing, see Chapter 5.

A code documentation is created with the help of *Doxygen* [153]. Besides the linked, public repository, there is a non-public repository for development and testing, which is employed for the analyses in Chapters 5 and 6.

Outline

A short description of the software is given in Section 3.2 with the usage outlined in Section 3.3, and to be detailed in the incorporated README file in Section 3.4. The installation process, issues, and an online code platform for researchers, where the code is available for direct use, are explained in Section 3.5.

3.2 Description

Hyperparameter values at the user's option determine the overall accuracy. These are the order of the finite difference scheme – ranging from one to thirteen – the lattice resolution, and the error

tolerances of the employed ODE solver.

The *SUNDIALS* [167, 168] package is used for the numerical solution of the resulting system of nonstiff, nonlinear differential equations for $\vec{f}$ in Equation (2.22). Explicitly, the *CVODE* solver [169] from the *SUNDIALS* family of solvers is employed, configured to use the implicit Adams method (Adams-Moulton formula) in conjunction with a fixed-point iteration [170, 171]. This is detailed in Chapter 6.

The numerical time integration error is controllable with *CVODE* by setting relative and absolute integrator tolerances per user-defined step size. *CVODE* adapts its internal time step sizes according to the system's dynamics guided by the tolerances. Larger time steps are performed in quiet regions and shorter steps in highly dynamic regions. Errors per step accumulate to a global error. Tolerances are set to below 10^{-12} for the simulations presented in Chapter 4.

The computation time is approximately independent of the stencil order, but the correct physical solution is rather approached with a higher stencil order. This is demonstrated in Chapter 6, within an investigation of trade-offs concerning scheme order, solver tolerances, grid resolution, and performance. Only for simulations on distributed systems the order of the numerical scheme becomes relevant for the communication load, which is discussed in Section 3.4 and in more detail in Chapter 5.

Output data are written at each user-defined time step optionally into convenient comma-separated values (*CSV*) files or a compact and efficient file in binary format. A *CSV* output file contains six columns for the field components E_x, E_y, E_z, B_x, B_y, and B_z; each column containing all grid values of the corresponding field component. A binary output file aligns all six field components for every lattice point.

Cluster computer communication for large-scale simulations with distributed memory can be achieved with multiprocessing, guided by the Message Passing Interface (*MPI*) [172] on a virtual Cartesian topology. To this end, an *MPI* implementation has to be available, see Section 3.5. The implementation of multiprocessing in the code is detailed in Chapter 5. The boundary conditions are periodic.

Use of *OpenMP* [173] is likewise optional to enforce more vectorization and to enable multithreading. The latter is beneficial for performance only at a scale of about 1000 compute cores. Details are discussed in Chapter 5.

The software is coded in *C++* with features up to the *C++20* standard [174]. To this end, it has undergone some modernization, partly making use of tools. A short overview can be found in Appendix B.

3.3 Usage

This section gives an overview of the usage, with details provided in the README file in Section 3.4.

Execution

Executable *Bash* and *Windows Powershell* example run scripts are provided in the software repository. An excerpt of a *Bash* run script for a 3D simulation is given in Listing 3.1.

```
1 #!/bin/bash
2
```

```
# Script to run executables built with CMake in the "src/build" directory
export EXECUTABLE="./build/hewes"

# Exporting MPI and OpenMP environment variables
export MPI_NUM_PROCESSES=64
export OMP_NUM_THREADS=8

# Specify the parameters
parameters=(\
    '# ---- General simulation settings -----'\
    $SCRATCH/outputs '# output directory'\
    3 '# spatial dimensions of simulation space'\
    1.0e-12 '# CVode relative tolerance'\
    1.0e-12 '# CVode absolute tolerance'\
    13 '# stencil order'\
    80e-6 '# physical length of the simulation box in x-direction'\
    80e-6 '# physical length of the simulation box in y-direction'\
    20e-6 '# physical length of the simulation box in z-direction'\
    1600 '# number of lattice points in x-direction'\
    1600 '# number of lattice points in y-direction'\
    200 '# number of lattice points in z-direction'\
    8 '# patches in x-direction'\
    8 '# patches in y-direction'\
    1 '# patches in z-direction'\
    3 '# linear vacuum (0), four-photon (1), six-photon (2), 4- and 6-photon (3) processes'\
    10e-6 '# simulation time in meters'\
    10 '# number of simulation steps performed by CVode'\
    10 '# output step multiples'\
    binary '# output style; binary or csv'\
...
    '# ---- Parameters for 2D/3D Gaussian pulses -----'\
    1 '# use of 2D/3D Gaussian pulse(s) yes (1) or no (0)'\
    2 '# number of 2D/3D Gaussian pulses; 0, 1, or 2'\
    40e-6 '# center of 2D/3D Gaussian 1 in x-direction'\
    40e-6 '# center of 2D/3D Gaussian 1 in y-direction'\
    10e-6 '# center of 2D/3D Gaussian 1 in z-direction'\
    1. '# x-component of normalized direction from which 2D/3D Gaussian 1 approaches the center'\
    0. '# y-component of normalized direction from which 2D/3D Gaussian 1 approaches the center'\
    0.05 '# amplitude of 2D/3D Gaussian 1'\
    0. '# polarization rotation from TE-mode (z-axis) in mulitples of pi/4'\
    2.3e-6 '# taille of 2D/3D Gaussian 1'\
    16.619e-6 '# Rayleigh length of 2D/3D Gaussian 1'\
    2e-5 '# shift from center in negative propagation direction of 2D/3D Gaussian 1'\
    0.45e-5 '# beam length of 2D/3D Gaussian 1'\
    40e-6 '# center of 2D/3D Gaussian 2 in x-direction'\
    40e-6 '# center of 2D/3D Gaussian 2 in y-direction'\
    10e-6 '# center of 2D/3D Gaussian 2 in z-direction'\
    0. '# x-component of normalized direction from which 2D/3D Gaussian 2 approaches the center'\
    1. '# y-component of normalized direction from which 2D/3D Gaussian 2 approaches the center'\
    0.05 '# amplitude of 2D/3D Gaussian 2'\
    0. '# polarization rotation from TE-mode (z-axis) in mulitples of pi/4'\
```

```
    2.3e-6 '# taille of 2D/3D Gaussian 2'\
    16.619e-6 '# Rayleigh length of 2D/3D Gaussian 2'\
    2e-5 '# shift from center in negative propagation direction of 2D/3D Gaussian 2'\
    0.45e-5 '# beam length of 2D/3D Gaussian 2'\
)

# Run it
mpirun -np $MPI_NUM_PROCESSES $EXECUTABLE "${parameters[@]}"
```

Listing 3.1: Excerpt of a *Bash* run script for a 3D simulation. This is a minimal version. The number of *MPI* processes and *OpenMP* threads are defined, to be detailed in Chapter 5. A number of parameters to configure the setting must be provided, starting with the lattice construction, accuracy and output decisions, followed by the parameters of the electromagnetic waves.

The passed time for each simulation step after starting is written to stdout, see Listing 3.2.

```
Simulation code: 23-03-11_10-09-43

Step 1    Elapsed time:  18.780211s

Step 2    Elapsed time:  35.193428s

Step 3    Elapsed time:  51.673806s

Step 4    Elapsed time:  68.332443s

Step 5    Elapsed time:  84.671380s
```

Listing 3.2: Excerpt of the shell output for a standard 2D simulation as to produce the results to be shown in Section 4.5. The execution of a run script first produces the starting time step, which forms the name of the simulation and the folder where the output data are written to. For each simulation step the number of seconds wall clock time are written to stdout.

General settings

Simulations can be performed on a one- to three-dimensional lattice. It can be decided whether to simulate in the linear Maxwell vacuum, the linear vacuum plus four-photon interactions, the linear vacuum plus six-photon interactions, or the linear vacuum plus four- and six-photon interactions.

The accuracy of the *CVODE* integrator and the order of the numerical scheme can be defined. For the absolute and relative error tolerances of the *CVODE* solver, 10^{-12} or lower are suitable choices, see Chapter 6. The implicit Adams method is preset to use the highest available order (twelve). The number of steps performed within the total physical propagation time should be chosen such that the step size is not larger than 2 µm. To fully exploit the beneficial dispersion properties, it is recommended to use the highest available order (thirteen) of the scheme.

Implementations of Gaussian laser pulses are available and their parameters can be configured. Higher than realistic amplitudes are commonly used in order to reduce the numerical noise, as mentioned in Section 2.5. This procedure is applied in Chapter 4.

The configuration of the grid and the decomposition into *MPI* processes becomes relevant in higher dimensions. For example, a 3D simulation to produce the results shown in Chapter 4, Figure 4.23 can be performed on a grid with $1400 \times 1400 \times 200$ points in less than four hours on about 400 compute cores. The output size amounts to nearly 20 gigabytes for one time step. A

weak scaling test for such 3D simulations is demonstrated in Section 5.3.1. The computational load scales linearly with the grid size and the simulation time varies only slightly when the load is equally distributed on up to about 1000 cores.

Postprocessing

The postprocessing of the data for the present work is done with the help of *Python* scripts, *Jupyter notebooks* [175] employing the *SciPy* library [176], and with the help of *Mathematica* [151] and *ParaView* [177]. *Python* modules are provided in the code repository to read the electromagnetic field components from the *CSV* or binary file into *NumPy* arrays for the ensuing analysis. Example analyses are provided in the code repository and in a *Mendeley Data* repository [150].

3.4 *HEWES* README

The Heisenberg–Euler Weak-Field Expansion Simulator is a solver for the all-optical QED vacuum. Vacuum polarization, due to omnipresent quantum fluctuations, supplements Maxwell's linear equations of electromagnetism by nonlinear photon–photon interactions. HEWES solves the nonlinear equations of motion for electromagnetic waves in the weak-field limit of the Heisenberg–Euler effective theory of QED with up to six-photon processes.

There is a paper that outlines the algorithm [147], a paper that introduces the software [148], and a *Mendeley Data* repository [150] with extra and supplementary materials.

Required software

CMake is used for building and a recent *C++* compiler version is required since features up to the *C++20* standard are used.

An *MPI* implementation supporting the *MPI* 3.1 standard is strongly recommended to make use of multiprocessing.

OpenMP is optional to enforce more vectorization and enable multithreading. The latter extra layer of parallelization is useful for performance only when a very large number of compute nodes is occupied.

The *CVODE* solver is fetched on-the-fly through *CMake*.

If *CVODE* (or the whole *SUNDIALS*package) is installed manually: The `SUNDIALS_DIR` variable in the `CMakeLists.txt` has to be set to the installation directory.

Version 6 is required, the code is presumably compliant with the upcoming version 7. Enable *MPI* and *OpenMP*, if desired. For optimal performance the `CMAKE_BUILD_TYPE` should be "Release".

3.4.1 Short user manual

In order to build the executable with *CMake*, execute, e.g., in the `src` directory, `cmake -S. -Bbuild` and then `cmake --build build`. On *Windows* a subsequent installation of the *SUNDIALS* modules is required. With `CMake` this can be done via `cmake --install build --config Debug`. The installation type has to be "Debug" with *MSVC* attributable to an issue with *SUNDIALS*. *MSMPI* is required, even though it only complies with *MPI*-2.0.

There is full control over all high-level simulation settings via command line arguments.

- First, the general settings are specified:
  - The path to the project directory. Therein, a `SimResults` folder is created and therein a folder named after the timestamp of the start of the simulation.
  - Whether to simulate in 1D, 2D, or 3D.
  - The relative and absolute integration tolerances of the *CVODE* solver. Recommended values are between 1e-12 and 1e-16.
  - the order of accuracy of the numerical scheme (the stencil order). An integer in the range 1–13 can be chosen.
  - The physical side lengths of the grid in meters.
  - The number of lattice points per dimension.
  - The slicing of the lattice into patches (relevant only for 2D and 3D simulations, automatic in 1D). This determines the number of patches and therefore the required distinct processing units for *MPI*. The total number of processes is given by the product of slices in any dimension. Note: In the 3D case patches should be chosen cubic in terms of lattice points. This is decisive for computational efficiency.
  - Whether to simulate in the linear vacuum (0), on top of the linear vacuum only 4-photon processes (1), only 6-photon processes (2), or 4- and 6-photon processes (3).
  - The total time of the simulation in units c=1, i.e., the distance propagated by the light waves in meters.
  - The number of time steps that will be solved stepwise by *CVODE*. In order to keep interpolation errors small, this number should not be chosen too small. One micro meter is appropriate.
  - The multiple of steps at which the field data will be written to disk.
  - The output format. It can be *CSV* (comma-separated values) or binary. For *CSV* format the name of the files written to the output directory comprise the time step and the writing process in the form `{step_number}_{process_number}.csv`. For binary output all data per step are written into one file and the name of the file is given by the step number.
- Second, the electromagnetic waves are chosen and their parameters specified. It can be chosen between plane waves (not much physical content, but useful for checks) and implementations of Gaussian pulses in 1D, 2D, and 3D. To identify which command line argument is exactly which parameter, see the comments in the short example *Bash* run scripts which are preconfigured for 1D, 2D, and 3D simulations. (One example is also provided as a *Windows Powershell* script.) Amplitudes are given in units of the critical field strength (Schwinger limit). Position and propagation parameters on the y- and z-axis are only effective if the grid has an extend in the corresponding dimension. A description of the wave implementations is given in the *Doxygen*-generated code reference. Note that the 3D Gaussians, as they are implemented up to now, are propagated only in the xy-plane. More waveform implementations will follow in subsequent versions of the code.

The boundaries are periodic. It has to be kept in mind that in 2D and 3D simulations the number of *MPI* processes has to coincide with the actual number of patches, as described above. An error will be thrown at startup otherwise.

If the program was built with *OpenMP* support, the environment variable `OMP_NUM_THREADS` needs to be set.

It can be useful to save the run script along with the output as a log of the simulation settings for later reference.

stdout and stderr should be monitored during the run (or redirected into files). The starting timestamp, the process steps, and the used wall times per step are printed on stdout. Errors are printed on stderr.

Note: Convergence of the employed *CVODE* solver cannot be guaranteed and issues of this kind can hardly be predicted. On top, they are even system-dependent. Piece of advice: Only decimals should be used for the grid settings and initial conditions, not, e.g., `sqrt` expressions. *CVODE* warnings and errors are reported on stdout and stderr, respectively.

Note on simulation settings

A typical starting point is to use two Gaussian pulses in 1D colliding head-on in a probe–pump setup. For this event, a high-frequency probe pulse has to be specified with a low amplitude and a low-frequency pump pulse with a high frequency. Both frequencies should be chosen to be below a sixth of the Nyquist frequency to minimize nonphysical dispersion effects on the lattice. The amplitudes should be below unity, the critical field strength, for the weak-field expansion to be valid.

Arising higher harmonics can then be investigated in frequency space via a Fourier analysis. The signals from the higher harmonics can be highlighted by subtracting the results of the same simulation in the linear Maxwell vacuum, such that only the nonlinear effects are left. Choosing the probe pulse to be polarized with an angle to the polarization of the pump, one may observe a fractional polarization flip of the probe in response of the nonlinear interaction. It has to be decided beforehand which steps should be written to disk for the ensuing analysis.

Example scenarios of colliding Gaussians are preconfigured for any dimension in the example scripts. Sensible configurations are listed in tables in Chapter 4.

Note on resource occupation

The computational load depends mostly on the grid size and resolution. The order of accuracy of the numerical scheme and *CVODE* are rather secondary, except for simulations running on many processing units. In the latter case, the communication load plays a major role, which in turn depends on the order of the numerical scheme. This is because the order of the scheme determines how many neighboring grid points are taken into account for the finite differences derivatives.

Simulations in 1D are relatively cheap and can easily be run on a modern notebook within some seconds. The output size per step is less than a megabyte. Simulations in 2D with about one million grid points are still feasible for a personal machine and take only a number of minutes. The output size per step is in the range of some dozen megabytes.

Sensible simulations in 3D require large memory resources and therefore need to be run on distributed systems. This implies an increased communication load. Even hundreds or thousands of cores can be kept busy for many hours. The output size rapidly amounts to hundreds of gigabytes for just a single state, if a high resolution is chosen. This hurdle forms a practical limit to the grid resolution.

Some scaling tests are shown in Section 5.3.1.

If the output is in binary form, the size can be easily calculated. Per step, it is given by the number of grid points times six (the number of field components) times 8 bytes.

Note on the output analysis

The field data are either written in *CSV* format to one file per *MPI* process, the ending of which (after an underscore) corresponds to the process number, as described above. This is the simplest solution for smaller simulations and a portable way that also works fast and is straightforward to analyze. Or, the option strictly recommended for larger write operations, in binary format with a single file per output step. Raw bytes are written to the files as they are captured in memory.

The latter option is more efficient and achieved with the help of *MPI IO*, and hence only possible if *MPI* is used. However, there is no guarantee of portability; postprocessing/conversion is required. The file name is given by the step number.

A `SimResults` folder is created in the chosen output directory if it does not exist and therein a folder named after the starting timestamp of the simulation (in the form `yy-mm-dd_hh-MM-ss`) is created. This is where the output files are written into.

There are six columns in the *CSV* files, corresponding to the six components of the electromagnetic field: E_x, E_y, E_z, B_x, B_y, B_z. Each row corresponds to one lattice point. Postprocessing is required to read in the files in order. A *Python* module taking care of this is provided. Likewise, another *Python* module is provided to read the binary data of a selected field component into a *NumPy* array. For its use, the byte order of the reading machine has to be the same as that of the writing machine.

More information describing settings and analysis procedures used for actual scientific results are given in Chapter 4 and a collection of corresponding analysis notebooks are uploaded to a *Mendeley Data* repository [150]. Some concise example *Python* analysis scripts can be found in the examples in the respository. The first steps therein demonstrate how the simulated data is correctly read in from disk to *NumPy* arrays using the provided `get field data` module. Harmonic generation in various forms is sketched as one application exhibiting nonlinear quantum vacuum effects. Analyses of 3D simulations are more involved due to large volumes of data. A script with the purpose to extract the ratio of polarization flipped photons of a laser pulse as a consequence of vacuum birefringence can also be found in the examples. Visualization requires tools like *ParaView* [177].

3.5 Installation

Automated building

Automated building is new to code version 0.2.2. The build process of the application is automated with the *CMake* build generator [178], which is platform- and compiler-agnostic with great portability. The installation is performed with two simple commands, as described in the README file.

For a simple project like this, *CMake* may introduce complexity and problems for the user without real advantages over a simple *Makefile* as it was used before. One could argue to rather keep it simple. However, the ease of use becomes even more pronounced with the on-the-fly dependency installation and the outstanding cross-platform compatibility. The installation has been successfully tested on various different hardware architectures and compilers.

Without its use, *SUNDIALS* would have to be manually installed by the user, in the specific configuration used for the project. Experience showed that this alone can pose an obstacle.

Furthermore, *CMake* enables straightforward optional use of *MPI* and *OpenMP*, based on their availability on the system and the user's choice. With this flexibility and automation, it is possible to run simulations not only on dedicated systems, but on any device provided with at least a sufficiently up-to-date *C++* compiler and *CMake* installation.

As explained in the README file in Section 3.4, *MPI* is recommended, not only on cluster computers. Therefore, an *MPI* implementation should be installed on the device such that *CMake* can find it. On an HPC system, the desired and required software needs to be loaded first, see Appendix A.

To render the code *Windows*-compatible, preprocessor definitions are used that create a specific version. The code is therefore customized to some extend to fit different systems and software.

Once the executable is built, the high-level settings are straightforwardly controlled with command line arguments.

Using *HEWES* on *Windows* systems

Employing *CMake* is especially comfortable for *Windows* systems, where the build system differs substantially from *Unix*-based ones. Installing requirements can be more difficult on the former and some software might not be available in the desired version.

Since *MSMPI* (*Microsoft MPI*) as of writing only complies with the older *MPI* 2.0 standard and *MPI* is crucial for performance, some *MPI*-related code parts have to be replaced with *MSMPI*-compliant routines for *Windows* systems. This is achieved with the help of preprocessor directives.

As mentioned in the README, owing to a known bug the *SUNDIALS* library only installs in "Debug" mode on *Windows* at the time of writing. Accordingly, the whole build and installation processes are performed in "Debug" mode, which degrades the performance.

Reproducible code capsule

Code Ocean [179] is a digital laboratory, a platform for computational research. It aims to overcome the problem of differing computing environments in order to make computational experiments rigorously reproducible. To this end, container technology is used, realizing a persistent and reproducible computing environment, the *Reproducible Compute Capsule*.

A software project can be released with access to the scientific community by depositing the code in a reproducible capsule verified by *Code Ocean*. Upon verification, the software is added to the *Code Ocean Open Science Library*. Registered scientists can execute the code directly in the cloud or download the self-contained capsule.

Code Ocean enables collaboration and versioning. There is a reproducible compute capsule of *HEWES* published on *Code Ocean* [180], preconfigured to run a 1D simulation by hitting "Reproducible Run", see Figure 3.1.

The capsule showcases how the code can be executed even on slightly outdated host systems. The installation from source of required software components, such as a sufficiently recent *C++* compiler and *CMake* version, are demonstrated for a *Linux* distribution in the "environment" section.

A highlighted run script leads through the subsequent execution steps. Hence, the reproducible compute capsule also functions as a tutorial. Researches are provided with a sufficient amount of free computing hours to conduct their own simulations in the cloud.

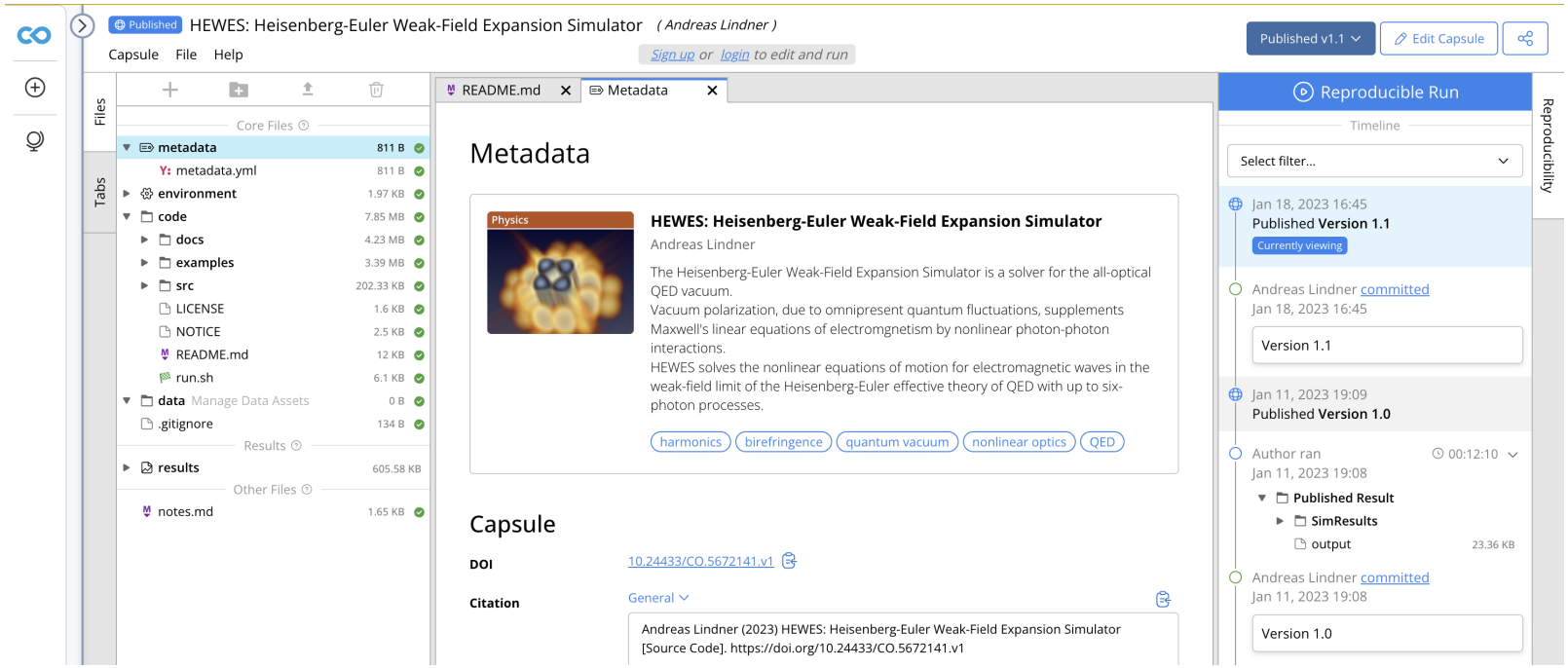
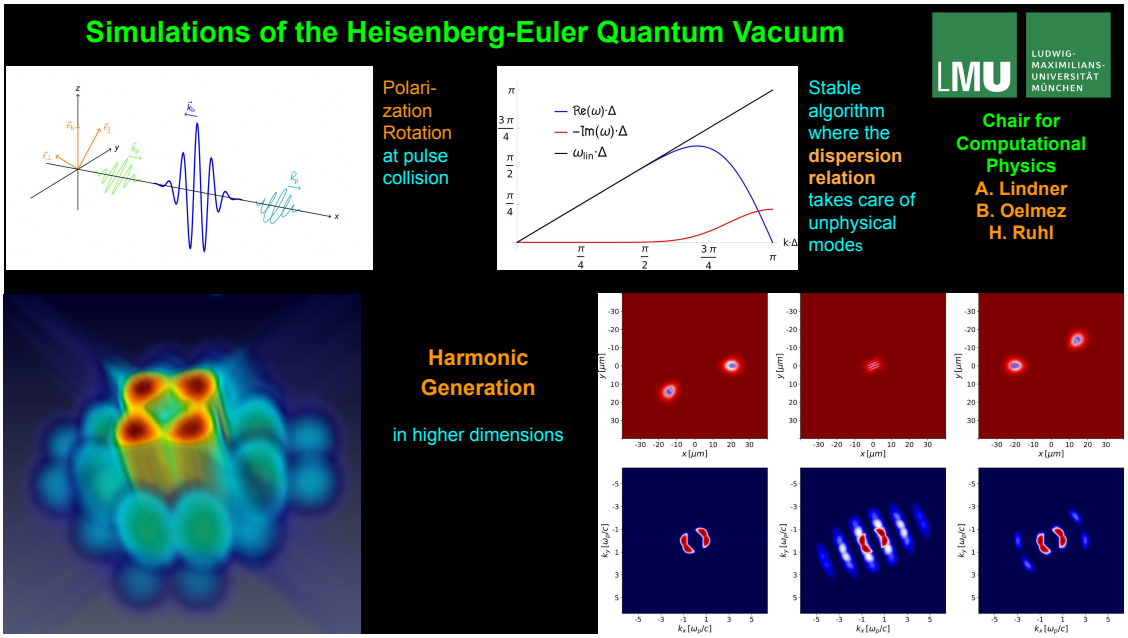


Figure 3.1: Screenshot of the Compute Capsule on *Code Ocean*. On the top right, a reproducible run can be started. It is preconfigured to a 1D simulation. The output is stored under "results" on the left. In that left panel the code can be browsed and the software environment be listed.

Chapter 4

All-Optical Quantum Vacuum Simulations



Graphical abstract of **Nonlinear simulations of the quantum vacuum in the Heisenberg–Euler weak-field expansion** [147]

Links

Research Unit FOR 2783: Probing the Quantum Vacuum at the High-Intensity Frontier
http://www.quantumvacuum.org

Reproducible Compute Capsule on *Code Ocean*
https://doi.org/10.24433/CO.5672141.v1

***Mendeley Data* Repository with analysis scripts and extra material**
https://doi.org/10.17632/f9wntyw39x.3

Highlights

- Universality with respect to pulse configurations in contrast to analytical treatments
- Scalability on distributed computing systems and 3D capability
- Inclusion of up to six-photon processes in the Heisenberg–Euler weak-field expansion

4.1 Introduction

As outlined in Chapter 1, the QED vacuum can be understood as a nonlinear, polarizable medium. Accordingly, nonlinear optical effects should in principle be observable, but the violations of the superposition principle for electromagnetic fields become noticeable only at unparalleled high intensities of the involved light sources.

Experimental and financial hurdles for the detection of such effects are thus very high. Facilities need to be equipped with high-intensity laser pulses and likewise ultra sensitive detectors. To gain a clear vision, such endeavors are nowadays backed by computer simulations. Consequently, advanced numerical frameworks are relevant to simulate future QED experiments. The present simulation project is part of a research unit dedicated to the investigation of the nonlinear optical properties of the quantum vacuum.

The capabilities of *HEWES* are demonstrated by solving various scenarios of nonlinear vacuum phenomena and by successfully cross-checking with analytical results where they exist. The prominent probe–pump approach outlined in Section 1.5 is the main constellation to be considered for the detection of nonlinear vacuum effects.

Tiny nonlinear effects can be successfully extracted by a proper postprocessing of the simulation results with remarkable accuracy. Up to six-photon processes can be simulated, where a useful feature of the code is that the different order processes can be simulated separately. The contributions of different orders of the weak-field expansion can thus be singled out. Full 3D capability has been worked out and first results can be presented.

The numerical predictions of the presented solver are cross-checked with analytical predictions for birefringence as calculated in [82, 83] and the generation of higher harmonics as predicted in [111].

Outline

In Section 4.2 the phase velocity reduction of a probe pulse propagating through a strong electromagnetic background is successfully compared to analytical predictions for sufficiently large background field strengths.

In Section 4.3 the phenomenon of vacuum birefringence is investigated with the help of probe–pump setup simulations. The theoretical prediction for the polarization flipping probability in plane wave backgrounds and its scaling with various parameters is verified. An extrapolation of the results to wavelengths in the x-ray regime is performed.

In Section 4.4 the capability of simulations to predict higher harmonics generated by vacuum nonlinearities is demonstrated. It is shown that in 1D available analytical results are accurately reproduced. Examples of harmonic generation in 2D and 3D simulations are presented in Section 4.5. The latter are hard, if not impossible, to obtain by analytical means and are used for comparison with the previous 1D simulations. Higher-dimensional simulations enable the investigation of complicated scenarios that cannot be solved analytically in the near future but are extremely pertinent to master the experimental challenges.

4.2 Phase velocity in a strong background

A probe plane wave is propagated along the x-axis,

$$\vec{E}(x;t) = \vec{A}_p \cos(kx - kvt) \,, \tag{4.1}$$

through a linearly polarized strong electromagnetic background with field strength A_b. The polarization of latter breaks the isotropy of space, giving rise to different refractive indices [74, 75, 117, 181]

$$n_\pm = 1 + \frac{\alpha}{45\pi}(11 \pm 3)\frac{A_b^2}{E_{\text{cr}}^2} = 1 + \delta n_\pm \tag{4.2}$$

for a probe polarization orthogonal (+) and parallel (-) to the background polarization. Since Equation (4.2) only takes into account the four-photon interaction contribution, i.e., it neglects all but the first nonlinear term in the weak-field expansion (1.20), the results are verified turning off six-photon processes in the simulations.

The resulting phase velocity change v_{nli} from the vacuum speed of light, given by

$$\frac{v_{\text{nli}}}{c} = \frac{v}{c} - 1 = \frac{1}{n_\pm} - 1 = -\frac{\delta n_\pm}{1 + \delta n_\pm} \,, \tag{4.3}$$

can be extracted with the help of a Fourier analysis of a time-propagated wave. When the passed time of propagation is chosen to be an integer multiple of λ, it is obtained with $l \in \mathbb{N}$ [145]

$$\vec{E}(x; l \cdot \lambda/c) = \vec{A}_p \cos(2\pi x/\lambda - 2\pi l\, v_{\text{nli}}) = \vec{A}_p \cos(2\pi(x/\lambda - l\, v_{\text{nli}})) \,. \tag{4.4}$$

The nonlinear phase velocity contribution can then be extracted from the phase after a spatial Fourier transformation evaluated at λ^{-1}. This results in

$$\frac{v_{\text{nli}}}{c} = -\frac{1}{2\pi l} \arg\left(\text{FT}[E(x;t_l)](\lambda^{-1})\right) \,, \quad \text{with} \quad c \cdot t_l = l \cdot \lambda \,. \tag{4.5}$$

The spatial Fourier transformation in Equation (4.5) can be replaced by a Fast Fourier Transformation in the analysis.

To analyze the phase velocity variation numerically, the background field strength is varied. In a second step the relative polarization of the waves is changed from parallel to orthogonal. The configurations are given in Table 4.1. The total simulation time is chosen to be 200 μm/c, conveniently divided into 100 steps of 2 μm – the chosen wavelength of the probe wave.

It can be seen that the nonlinear interactions give note to themselves in a reduction of the phase velocity of the simulated waves in Figure 4.1. For sufficiently large background field strengths the numerical values are in very good agreement with the analytical predictions.

4.3 Vacuum birefringence

Polarization, or helicity, flipping of a fraction of photons in a probe pulse propagating through a strong background pump pulse is a result of vacuum birefringence. The origin of the effect is again the breaking of the isotropy of space by the polarization of the strong background. The refractive

Table 4.1: Settings for phase velocity variation tests. The background amplitudes and the relative polarizations are varied. The large wavelength of the background manifests itself as an ever-persistent static background.

Grid	Length	100 µm
	Lattice Points	1000
Background	$\vec{A}$	$(0,3,0)\times10^{-6}E_{\mathrm{cr}}$ up to $(0,0.9,0)E_{\mathrm{cr}}$
	λ	1 Pm
Probe	$\vec{A}$	$(0,1,0)\times10^{-6}E_{\mathrm{cr}}$ and $(0,0,1)\times10^{-6}E_{\mathrm{cr}}$
	λ	2 µm

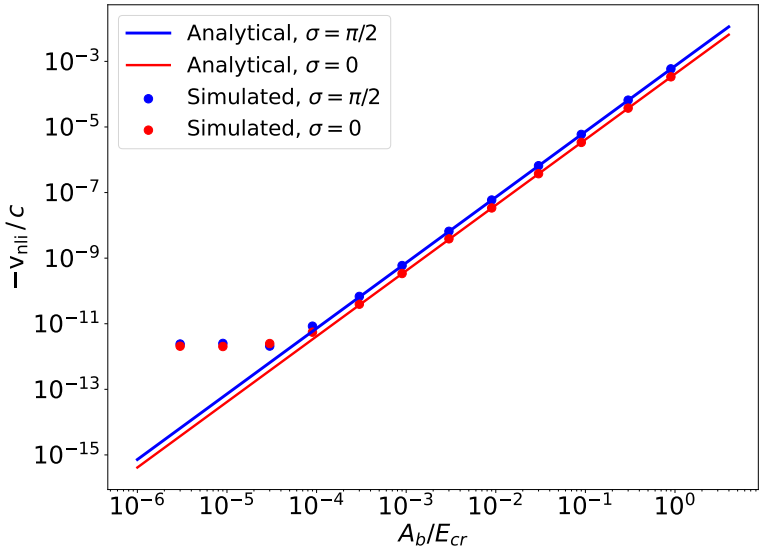


Figure 4.1: Nonlinear contribution to the phase velocity slow-down in a background with varying field strength. The simulation results converge to a value of $v_{\mathrm{nli}} \approx 2\times10^{-12}$ for small background field strengths. This is the phase velocity reduction caused by numerical errors, which are getting larger than the physical effect for background field strengths $A_b < 10^{-4}E_{\mathrm{cr}}$. Deviations from the analytical expectation are higher for lower background field strengths. The error at $A_b = 3\times10^{-4}E_{\mathrm{cr}}$ is still 6.1% for parallel and 4.5% for orthogonal relative polarization of the probe. This difference originates in the probe wave not being a perfect "probe" as in the idealized theoretical scenario and thus is contributing with its own polarization of $A_p = 3\times10^{-6}E_{\mathrm{cr}}$. The values for background field strengths larger than $10^{-4}E_{\mathrm{cr}}$ have a mean absolute percentage error, c.f. Equation (6.15), of 1.8% for parallel relative polarization and 1.2% for orthogonal relative polarization of the probe.

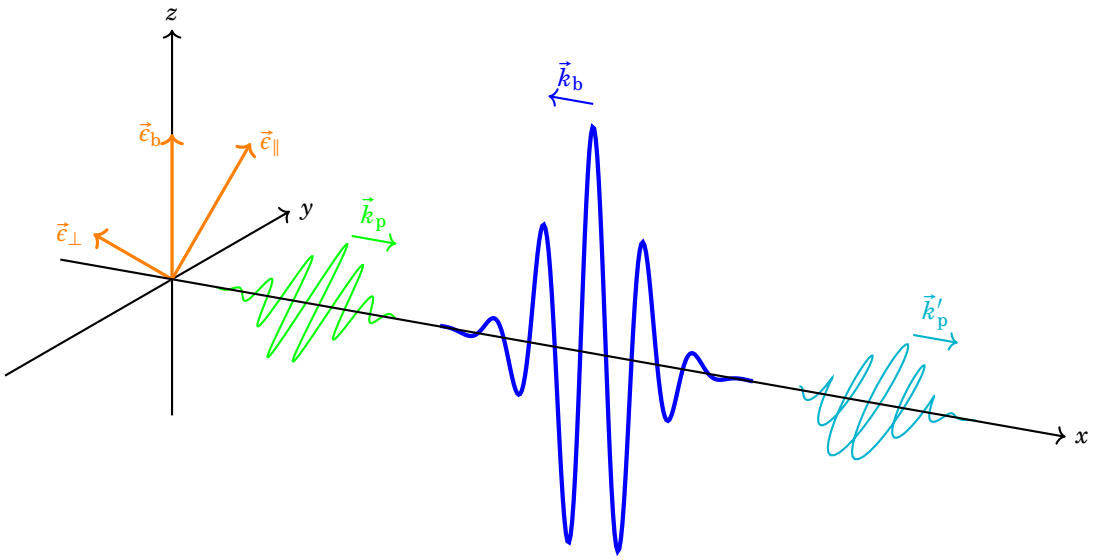


Figure 4.2: Qualitative depiction of the electric fields in a coaxial probe–pump experiment for the measurement of vacuum birefringence. The probe (*green*) traverses the counter-propagating pump (*blue*), experiencing a polarization rotation due to different refractive indices. The originally linearly polarized probe is afterwards marginally elliptical (*turquoise*). The effect is depicted significantly exaggerated for visibility. This effect is results from the fact that the isotropy of space for the charged particle–antiparticle fluctuations in the vacuum is broken by the polarization of the strong background pulse. The coupling of these fluctuating particles in turn to the probe pulse results in different refractive indices for the polarization modes of the probe, as given in Equation (4.6).

indices from above,

$$n_- = 1 + \frac{8\alpha}{45\pi}\frac{E^2}{E_\mathrm{cr}^2} \quad \text{and} \quad n_+ = 1 + \frac{14\alpha}{45\pi}\frac{E^2}{E_\mathrm{cr}^2}\,, \tag{4.6}$$

generate a difference in optical path length for the probe pulse polarization components parallel and orthogonal to the pump polarization, which results in *bi*refringence. On a microscopic level, a portion of the probe pulse's quanta flip their polarization. Macroscopically, the overall polarization experiences a tiny rotation. There are facilities constructed with the purpose to detect vacuum birefringence [182, 183].

A typical probe–pump scenario devised for the observation of helicity flips is sketched in Figure 4.2. A probe pulse propagates through a strong low-frequency pump field in which spatial isotropy is broken. While propagating through a pump field a fraction of probe pulse photons flips their polarization by 90° which results in a tiny ellipticity of an initially linearly polarized probe pulse. A corresponding simulation configuration showing one polarization direction is shown in Figure 4.3.

In order to backtest the numerical solver, firstly, in Section 4.3.1, parametric checks of the flipping probability as derived in [82] are performed. The settings given in Table 4.2 are chosen to this end with Gaussian pulses given by Equation (4.8) below. With that result being verified, secondly, in Section 4.3.2, the parametric scaling properties are made use of to extrapolate results to wavelenghts in the x-ray regime. This is compared to a calculation for a realistic polarization flipping scenario calculated in [83]. The parameters for that setup are listed in Table 4.3.

For both cases the normalized vectors parallel and orthogonal to the initial probe polarization

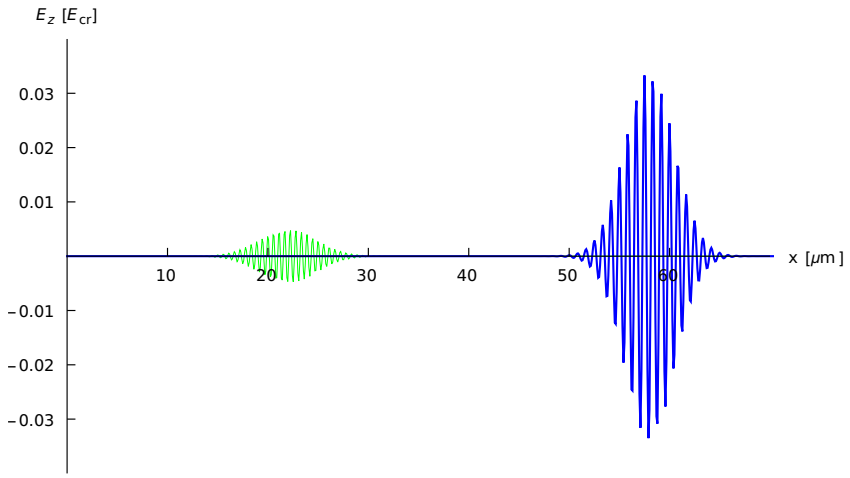


Figure 4.3: Sketch of the pulse configuration for the simulation of polarization flipping as shown in Figure 4.2. On the left is the weaker probe pulse which propagates to the right; on the right is the strong pump pulse propagating leftwards. Note that in actual simulations (parameters in Table 4.2) the probe field strength and wavelength are significantly smaller. Adjustments are made here for better visibility.

are given by

$$\vec{\varepsilon}_\parallel = (0, 1/\sqrt{2}, 1/\sqrt{2}) \quad \text{and} \quad \vec{\varepsilon}_\perp = (0, -1/\sqrt{2}, 1/\sqrt{2}) \,. \tag{4.7}$$

For the simulations, 1D Gaussian pulses are used in the form

$$\vec{E} = \vec{A}\, e^{-(\vec{x}-\vec{x}_0)^2/\tau^2} \cos\left(\vec{k}\cdot\vec{x}\right) \,, \quad \text{with} \quad \vec{k} = \frac{2\pi}{\lambda}\hat{k} \,, \quad \text{and} \quad \vec{B} = \hat{k} \times \frac{\vec{E}}{c} \,, \tag{4.8}$$

where the vector $\vec{A}$ comprises amplitude and polarization, $\vec{x}_0$ is the center of the pulse, τ its width, λ the wavelength, and $\hat{k}$ the unit propagation direction vector.

Pulses are implemented in space without explicit time dependence. With spatial derivatives via the finite differences scheme, the ODE system in time (2.22) from Chapter 2 is formulated, which is solved by *CVODE* for the time evolution. For convenience, the normalized vector $\hat{k}$ indicating the propagation direction is stated in the parameter tables.

Since analytical estimates for polarization flipping such as in [83] make use of a photon picture, while the numerical simulations presented in the present work propagate coherent modes, the mapping

$$\frac{N_\perp}{N} = \frac{\hbar\omega N_\perp}{\hbar\omega N} = \frac{\mathrm{E}_\perp}{\mathrm{E}_{\mathrm{tot}}} \tag{4.9}$$

is used. The energies in the respective polarization directions are proportional to the electric field strength projections squared,

$$\mathrm{E}_\perp \sim \sum_{x_i \in \mathscr{C}} \left(\vec{E}(x_i)\cdot\vec{\varepsilon}_\perp\right)^2 \,, \quad \mathrm{E}_\parallel \sim \sum_{x_i \in \mathscr{C}} \left(\vec{E}(x_i)\cdot\vec{\varepsilon}_\parallel\right)^2 \,, \quad \mathrm{E}_{\mathrm{tot}} = \mathrm{E}_\perp + \mathrm{E}_\parallel \,. \tag{4.10}$$

All other factors in (4.9) cancel out. It has to be mentioned that Equation (4.9) implies that the frequencies of the signal photons equal that of the probe pulse. However, as is shown in Section 4.4, the nonlinear interaction results in a small fraction of photons with altered frequency.

4.3.1 Vacuum birefringence – parametrical dependencies

For a probe pulse coaxially counter-propagating to a plane wave background field, the polarization flipping probability, taking into account again only up to four-photon interactions, in the low-energy approximation, is given by [82]

$$P_{\text{flip}} = \frac{N_\perp}{N} = \frac{\alpha^2}{255\,\lambda_p^2} \sin^2(2\sigma) \left(\int dx \, \frac{A_b(x)^2}{E_{\text{cr}}^2} \right)^2 , \tag{4.11}$$

where σ is the initial angle between the probe and pump polarizations, λ_p the wavelength of the probe pulse, and A_b the amplitude of the background pulse. The propagation direction of the probe is assumed to be perpendicular to the pump polarization and the probe field strength to be negligible compared to the pump. The probability directly translates to the flip ratio,

$$N_\perp = P_{\text{flip}} \cdot N \, . \tag{4.12}$$

Formula (4.11) yields all the parametric dependencies for the probability of polarization flips and indirectly excludes other parameters. There is a the strong dependence on the optical path of the pump pulse and on the probe wavelength. Notably, the ratio is independent of the shapes of the pulses. Limitations of the above formula for focused background pulses are discussed in the following section. In 1D simulations the background can be modeled as a Gaussian pulse. To investigate the scaling properties of the numerical solver the settings in Table 4.2 are used, where only those parameters affecting Equation (4.11) are actually relevant. A time-resolved flipping process for those parameters is depicted in Figure 4.4. The results of the parametric scaling tests are visualized in Figure 4.5. There is perfect agreement between the 1D simulation results and formula (4.11).

Neglecting the signals for $\sigma = 0, \pi/2$, where $P_{\text{flip}} = 0$ analytically, which cannot be respected in a relative error calculation when the true values serve as baseline, the mean absolute percentage errors for each scaling test are below 0.1%. A *Mathematica* [151] analysis of the parametric scaling properties of Equation (4.11) and the comparison to numerical results can be found in [152].

With these scaling properties being verified in the algorithm, hereinafter the analytical result in case (a) of [83], where a small probe pulse traverses a strong pump field, is compared to an extrapolation of simulation results.

4.3.2 Vacuum birefringence – extrapolation to the x-ray regime

Simulations of birefringence effects are computationally expensive in higher dimensions for the small wavelengths and field strengths targeted in experiments. Making use of the scaling properties in Equation (4.11), the phenomenon of birefringence can still be predicted for the parameters accessible in planned near future experiments by simulating numerically feasible, quasi-1D setups with consecutive extrapolation.

To this end this approach is used to reproduce an analytical result for a coaxial probe–pump setup with Gaussian laser pulses in a realistic scenario by considering case (a) of [83]. In this case the radius of the probe pulse is taken to be much smaller than the waist of the pump beam, such that the probe does not sense the transverse structure of the pump. This scenario thus amounts to a 1D case. The settings for the simulation of the scenario described in case (a) in [83] are given in

Table 4.2: Parameters for probe and pump beams chosen to test the parametric dependencies of polarization flipping in Equation (4.11). The probe wavelength, the pump field strength, and their relative polarizations are varied to obtain the parametric scaling results of Figure 4.5.

Grid	Length	80 µm
	Lattice Points	60×10^3
Pump	$\vec{A}$	$(0,0,34) \times 10^{-3} E_{\mathrm{cr}}$
	$\hat{k}$	(-1,0,0)
	λ	800 nm
	$\vec{x}_0$	58 µm
	τ	3.5 µm
Probe	$\vec{A}$	$(0,50,50) \times 10^{-6} E_{\mathrm{cr}}$
	$\hat{k}$	(1,0,0)
	λ	25 nm
	$\vec{x}_0$	22 µm
	τ	4.0 µm

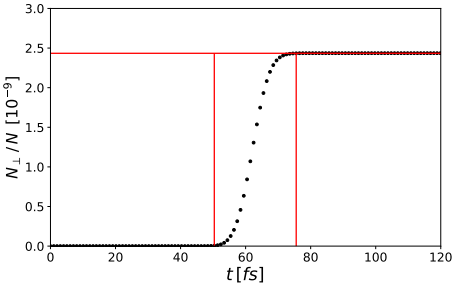


Figure 4.4: Time evolution of the polarization flipping ratio for the parameters presented in Table 4.2. A simulation time of 30 µm divided into 100 steps, so 1 fs per step is used. The settings used here provide an interaction time of 25 fs, indicated by the red vertical lines. The red horizontal line corresponds to the asymptotic relative flip ratio.

Table 4.3: Parameters for probe and pump beam adapted to [83]. The pump field strength is obtained as the square root of the ratio of intensity to critical intensity. The pump pulse duration is 30 fs (2τ in the $1/e^2$ criterion).

Grid	Length	80 µm
	Lattice Points	80×10^3
Pump	$\vec{A}$	$(0,0,0.34) \times 10^{-3} E_{\mathrm{cr}}$
	$\hat{k}$	(-1,0,0)
	λ	800 nm
	$\vec{x}_0$	58 µm
	τ	4.5 µm
Probe	$\vec{A}$	$(0,50,50) \times 10^{-6} E_{\mathrm{cr}}$
	$\hat{k}$	(1,0,0)
	λ	96 pm
	$\vec{x}_0$	22 µm
	τ	3.4 µm

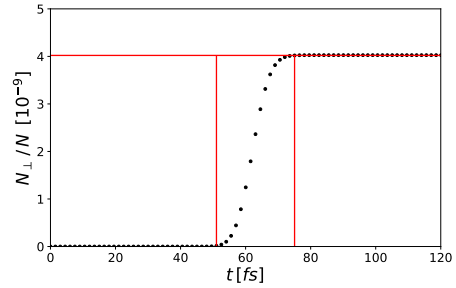


Figure 4.6: Time evolution of the polarization flipping probability for an adaptation of the parameters presented in Table 4.3. One time step corresponds to 1.5 fs. The adaptations are: The pump field strength is magnified by a factor of 100 to $34 \times 10^{-3} E_{\mathrm{cr}}$ in order to reduce numerical noise. The probe wavelength is enlarged to a computationally acceptable value of 25 nm. The distance between the vertical red lines is the total interaction time $t_I = 24$ fs. The horizontal red line denotes the asymptotic flip ratio.

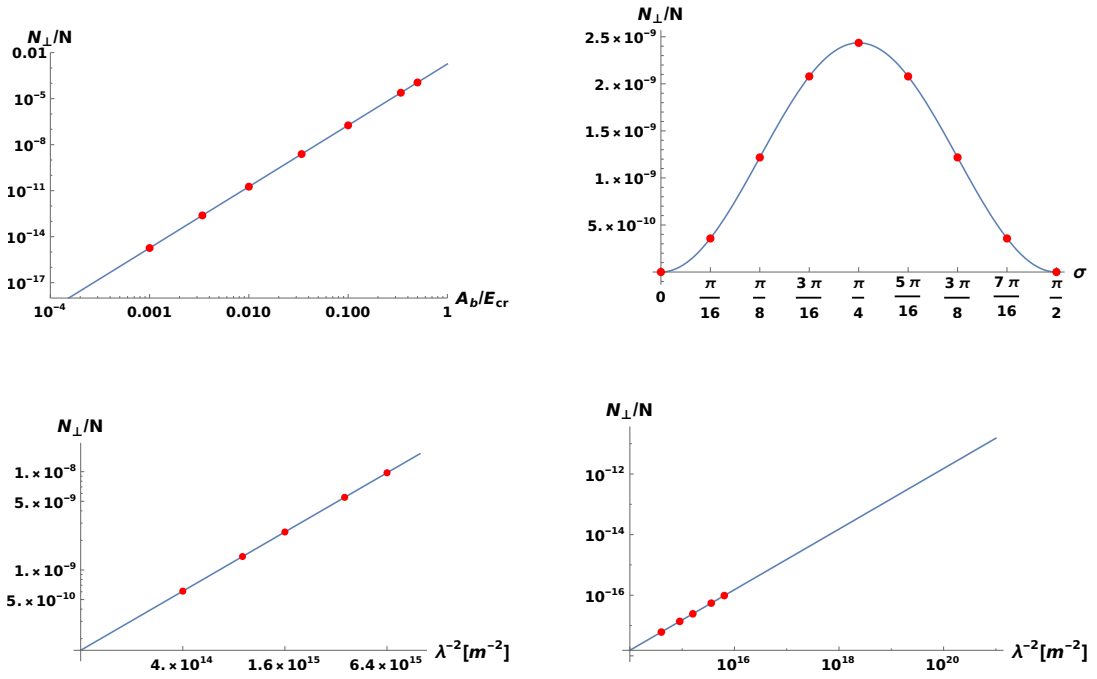


Figure 4.5: Scaling of the polarization flipping probability with variations of the parameters given in Table 4.2. The solid lines are the analytical curves obtained with the help of Equation (4.11). The red dots are simulation results. *Top left*: varying the background field strength. *Top right*: varying the relative polarization angle of probe and pump. *Bottom left*: varying the probe wavelength. *Bottom right*: combining the scaling of the pump field strength and probe wavelength; A_b is scaled down by a factor of 100 compared to the parameter in Table 4.2 and simultaneously an extrapolation to frequencies in the x-ray regime is performed. This last way of combining scaling properties is useful in order to extrapolate to relevant probe frequency regimes, while keeping the numerical accuracy high and the computational load low.

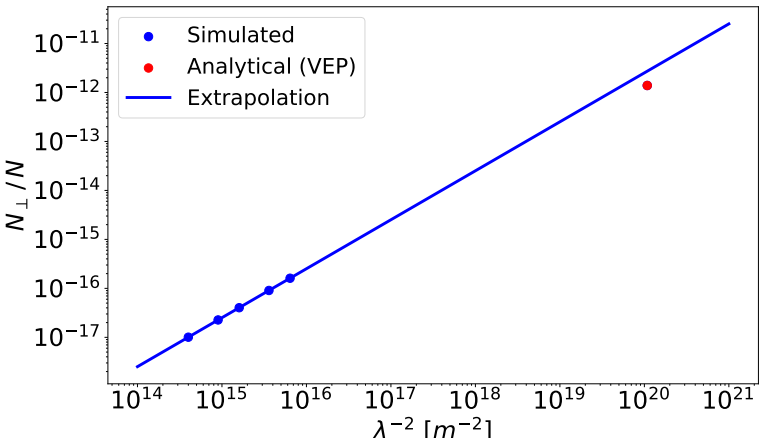


Figure 4.7: Extrapolation of polarization flipping ratios to the x-ray regime and comparison to a result obtained via the vacuum emission picture in [83] (*red dot*). The blue line is an extrapolation of simulation results (*blue dots*) with various probe wavelengths.

Table 4.3. The calculation in [83] is performed in the vacuum emission picture [154].

Some parameters in [83], devised for experimental verification, impair the numerical approach. First, the pump field strength is too low to extract the flipping process from the numerical noise. To combat this, the field strength scaling properties can be used to simulate with a larger background amplitude. Second, the probe wavelength of $\lambda_p = 96\,\mathrm{pm}$ is in the x-ray regime to amplify the effect, c.f. Equation (4.11), and is therefore problematically small for modeling on a discrete grid. An extremely fine grid would be necessary to model that pulse. To evade computations that expensive, the wavelength scaling properties can be made use of. The resulting extrapolation is thus a combination of two scaling methods in the way shown in the bottom right of Figure 4.5 and described in that caption.

Figure 4.7 shows that the extrapolated flipping ratio of 2.72×10^{-12} is almost twice as high as the flipping ratio of 1.39×10^{-12} obtained in [83]. This is attributable to the neglect of a longitudinally localizing term with the Rayleigh length in the pulse form in Equation (4.8), c.f. the higher-dimensional Gaussian pulse in Equation (4.30). This yields an estimated further suppression of about a factor of two. Accordingly, the value obtained at the corresponding probe frequency via formula (4.11) is 2.73×10^{-12} and agrees with the simulations.

It is of course possible to carry out simulations of birefringence processes in higher spatial dimensions with increased computational load. Part of a parallel project is the investigation of polarization flipping in extreme cases in 1D and 2D. Full 3D simulations are left for future work. A principal simulation scenario for future prospects regarding polarization flipping is given at the end of the chapter in Figure 4.26.

4.4 Harmonic generation

To further crosscheck the solver, the prominent probe–pump scenario for the detection of nonlinear vacuum signatures shown in Figure 4.8 is considered again with two head-on colliding pulses, a strong background pulse and a weaker probe pulse. For this analysis, the former pulse is assumed to have zero frequency. The initial settings are listed in Table 4.4.

Approximate analytical results for this scenario are derived in [111, 113]. The effective vertices for four- and six-photon scattering in Figure 4.9 (a) can produce outgoing photons with higher frequency by photon merging. For example, in Figure 4.9 (b) two probe photons and a zero-frequency

Table 4.4: Initial settings to observe harmonic generation in 1D simulations, see Figure 4.8.

Grid	Length	300 µm
	Lattice Points	4000
Pump	$\vec{A}$	$(0,20,0)\times 10^{-3}E_{\mathrm{cr}}$
	$\hat{k}$	(-1,0,0)
	λ	1 m
	x_0	200 µm
	τ	12.8 µm
Probe	$\vec{A}$	$(0,5,0)\times 10^{-3}E_{\mathrm{cr}}$
	$\hat{k}$	(1,0,0)
	λ	2 µm
	x_0	100 µm
	τ	10 µm

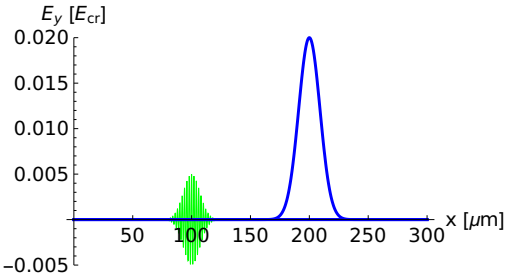
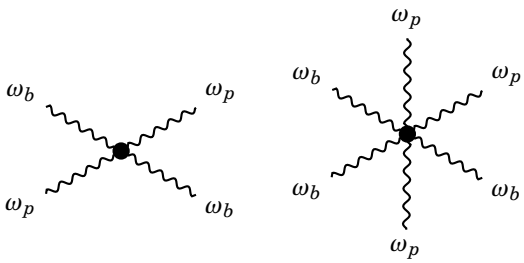
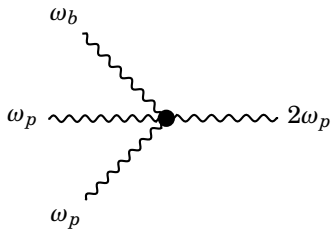


Figure 4.8: Visualization of the pulse configuration to detect higher harmonics with a zero-frequency background pulse, see Table 4.4.



(a) Effective vertices for four- and six-photon scattering.



(b) Example of high-harmonic generation with a zero-frequency background.

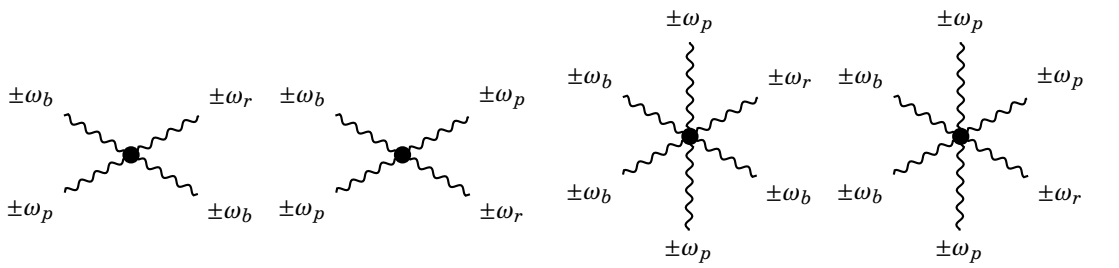
Figure 4.9: Feynman diagrams for harmonic generation with a probe (*subscript p*) and background (*subscript b*) wave. In (b) there is an implicit time axis from left to right.

background photon merge into an outgoing photon with frequency $2\omega_p$. The possible contributions of two-wave scattering that result from the first orders of the weak-field expansion are listed in Figure 4.10.

In the present case of $\omega_b = 0$, there are for four-photon processes

- the scattering of a background and a probe photon contributing with one photon to the zeroth harmonic ($\omega_r = 0$), also called dc component, and with one to the first harmonic ($\omega_r = \omega_p$), also called fundamental harmonic;
- two background photons and one probe photon merging to produce a photon of the fundamental harmonic ($\omega_r = \omega_p$); and
- two probe photons and one background photon merging to produce a photon of the second harmonic ($\omega_r = 2\omega_p$).

For six-photon processes it is obtained



(a) $\omega_r \in \omega_b, 2\omega_p \pm w_b, \omega_p, \omega_p \pm 2\omega_b$

(b) $w_r \in \omega_b, 3\omega_b, \omega_p + 3\omega_b, 2\omega_p \mp 2\omega_b, \omega_p, \omega_p \mp 2\omega_b, 3\omega_p \mp 2\omega_b$

Figure 4.10: Allowed vertices of Figure 4.9 with resulting frequencies ω_r for (a) four-photon and (b) six-photon processes in a probe–pump setup [113]. ± indicates in-/outgoing photons. Further restrictions are posed upon the asymptotic states by energy conservation.

- the sheer scattering of background and probe photons contributing to the dc component and the fundamental harmonic;
- two background and two probe photons merging and producing one photon contributing to the second harmonic and one contributing to the dc component;
- two background and two probe photons merging and producing two photons contributing to the fundamental harmonic;
- two background photons and three probe photons merging and producing a photon contributing to the third harmonic ($\omega_r = 3\omega_p$); and
- the merging of three background and two probe photons producing a photon contributing to the second harmonic.

A visualization of the contributions at selected points in time is provided with Figure 4.11.

These simulations are strictly 1D and the use of plane waves leads to strong constraints. It can be seen that the asymptotic contribution to the second harmonic is only due to six-photon processes. That is because wave-mixing – resulting pulses that are combined of photons of both pulses – is not allowed asymptotically. The reason behind this is energy conservation, since for coaxial pulses and a photon resulting of wave-mixing it is found

$$k_r^\mu = n_p\,\omega_p\,(1,\hat{k}) + n_b\,\omega_b\,(1,-\hat{k}) \quad \text{and} \quad (k_r^\mu)^2 = 0 \;\Rightarrow n_p\,n_b \stackrel{!}{=} 0\,, \tag{4.13}$$

where n_p and n_b are the numbers of the contributing photons and $\hat{k}$ is the unit propagation direction vector of the probe pulse. These states are thus only visible in the overlap position. The six-photon process, on the other hand, can produce second harmonics without wave-mixing, see the second point for six-photon processes above.

As discussed in the context of birefringence in Section 4.3, the 1D case corresponds to a simplified handling of the experimentally relevant scenario of counter-propagating pulses.

For the highest generated harmonic, the short-lived third harmonic, the rule-of-thumb resolution limit for the grid (2.58) is slightly exceeded. This comes without noticeable accuracy problems as is demonstrated in the following Sections.

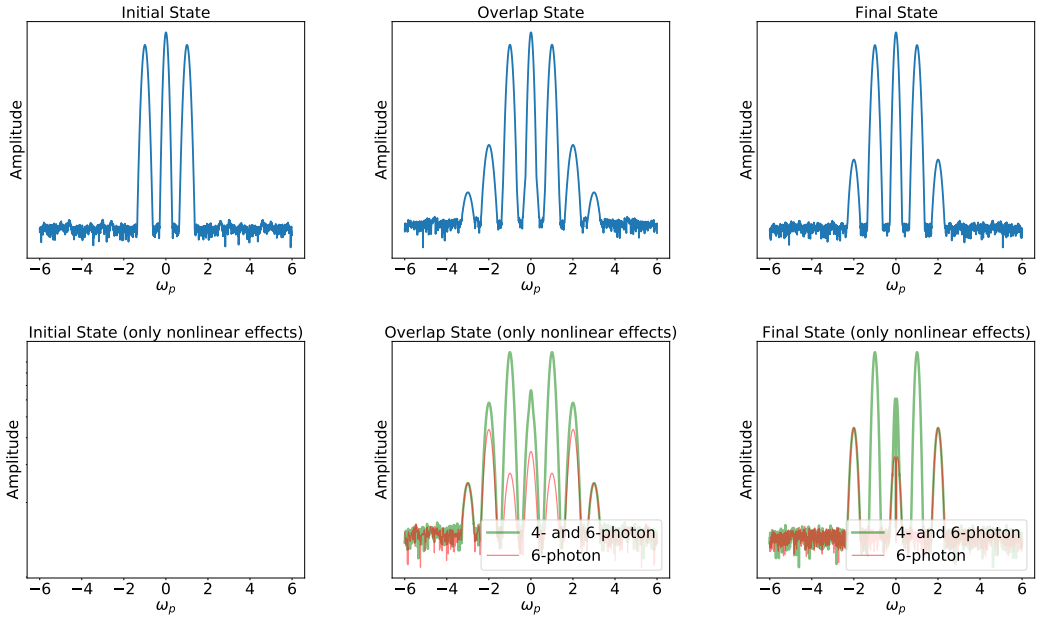


Figure 4.11: Log-scale plot making the higher harmonics visible in frequency space. *Top*: a full simulation in the linear vacuum supplemented by four- and six-photon nonlinear interactions. *Bottom*: after subtraction of the linear vacuum. The initial state contains only the main signals of the pulses with no nonlinear interaction yet present. The overlap state denotes the time step where the pulses are directly overlapping, at the final state they have separated again – the asymptotic field is left. It can be deduced that the third harmonic and the asymptotic part of the second harmonic are solely ascribable to six-photon processes.

4.4.1 Harmonic generation – analytical results

Analytical methods in [111, 113] contain a derivation of iterative solutions to the nonlinear equations of motion for zero-frequency backgrounds. With the probe (p) and background (b) pulses as time-dependent 1D Gaussian pulses with the parameters of Table 4.4 the pulses are given by, c.f. Equation (4.8),

$$\vec{E}_p(x;t) = \vec{\epsilon}_p A_p \, e^{-(k_p^\mu x_{\mu p})^2/(\omega_p \tau_p)^2} \cos(k_p^\mu x_{\mu p}) \quad \text{and} \quad \vec{E}_b(x;t) = \vec{\epsilon}_b A_b \, e^{-(k_b^\mu x_{\mu b})^2/(\omega_b \tau_b)^2} \,, \tag{4.14}$$

with $k_j^\mu x_{j\mu} = \omega_j t - \vec{k}_j \vec{x}_j$. The shifted coordinates of probe and background field read

$$\vec{x}_p = (x - 100\,\mu\text{m}, 0, 0) \quad \text{and} \quad \vec{x}_b = (x - 200\,\mu\text{m}, 0, 0) \,. \tag{4.15}$$

Writing a combined electric field as

$$\vec{E} = \vec{E}_p + \vec{E}_b \,, \tag{4.16}$$

the inhomogeneous wave equation in 1D can be written as

$$(\partial_t^2/c^2 - \partial_x^2)\vec{E} = T[\vec{E}] \,. \tag{4.17}$$

The source term $T[\vec{E}]$ comprises the nonlinear Heisenberg–Euler interactions. Decomposing the electric field into

$$\vec{E} = \vec{E}^{(0)} + \vec{E}^{(1)} + \dots \,, \tag{4.18}$$

where $\vec{E}^{(0)}$ solves the free vacuum wave equation $(\partial_t^2/c^2 - \partial_x^2)\vec{E}^{(0)} = 0$, an iterative procedure is obtained [111, 113] in which

$$(\partial_t^2/c^2 - \partial_x^2)\vec{E}^{(1)} = T[\vec{E}^{(0)}] \, . \tag{4.19}$$

With the polarization for both fields given by $\vec{\epsilon} = (0,1,0)$ and defining the shorthands

$$\kappa_p = \frac{k_p^\mu x_{\mu p}}{\omega_p \tau_p} \quad \text{and} \quad \kappa_b = \frac{k_b^\mu x_{\mu b}}{\omega_b \tau_b} \, , \tag{4.20}$$

it is then obtained for the solution to the nonlinear wave equation (4.19) at the first iterative order for [111, 113]

- the dc component:

  the overlap field

  $$\vec{E}_{0,o}^{(1)} = -\frac{8\alpha}{180\pi} A_p^2 \vec{E}_b(x;t) e^{-2\kappa_p^2} \vec{\epsilon} \tag{4.21}$$

  and the asymptotic field

  $$\vec{E}_{0,a}^{(1)} = \frac{8\alpha}{180\pi} A_p^2 \sqrt{\frac{\pi}{2}} \frac{\tau_p \kappa_b}{\tau_b} (1 + \mathrm{erf}(\sqrt{2}\kappa_b)) A_b \, e^{-\kappa_b^2} \vec{\epsilon} \, ; \tag{4.22}$$

- the fundamental harmonic:

  the overlap field

  $$\vec{E}_{1,o}^{(1)} = -\frac{8\alpha}{90\pi} A_p e^{-\kappa_p^2} \vec{E}_b(x;t)^2 \cos(k_p^\mu x_{p\mu}) \vec{\epsilon} \tag{4.23}$$

  and the asymptotic field

  $$\vec{E}_{1,a}^{(1)} = \frac{8\alpha}{90\pi} A_p e^{-\kappa_p^2} A_b^2 \sqrt{\frac{\pi}{2}} \omega_p \tau_b \, \frac{1 + \mathrm{erf}(\sqrt{2}\kappa_b)}{2} \sin(k_p^\mu x_{p\mu}) \vec{\epsilon} \, ; \tag{4.24}$$

- the second harmonic:

  the overlap field

  $$\vec{E}_{2,o}^{(1)} = -A_p^2 e^{-2\kappa_p^2} \left[\frac{8\alpha}{180\pi} \vec{E}_b(x;t)^2 + \frac{96\alpha}{630\pi} \vec{E}_b(x;t)^3 \right] \cos(2k_p^\mu x_{p\mu}) \vec{\epsilon} \tag{4.25}$$

  and the asymptotic field

  $$\vec{E}_{2,a}^{(1)} = \frac{96\alpha}{315\pi} A_p^2 e^{-2\kappa_p^2} A_b^3 \sqrt{\frac{\pi}{3}} \omega_p \tau_b \, \frac{1 + \mathrm{erf}(\sqrt{3}\kappa_b)}{2} \sin(2k_p^\mu x_{p\mu}) \vec{\epsilon} \, ; \text{ and} \tag{4.26}$$

- the third harmonic:

  the overlap field

  $$\vec{E}_{3,o}^{(1)} = -\frac{96\alpha}{1260\pi} A_p^3 e^{-3\kappa_p^2} \vec{E}_b(x;t)^2 \cos(3k_p^\mu x_{p\mu}) \vec{\epsilon} \tag{4.27}$$

  and the asymptotic field

  $$\vec{E}_{3,a}^{(1)} = 0 \, . \tag{4.28}$$

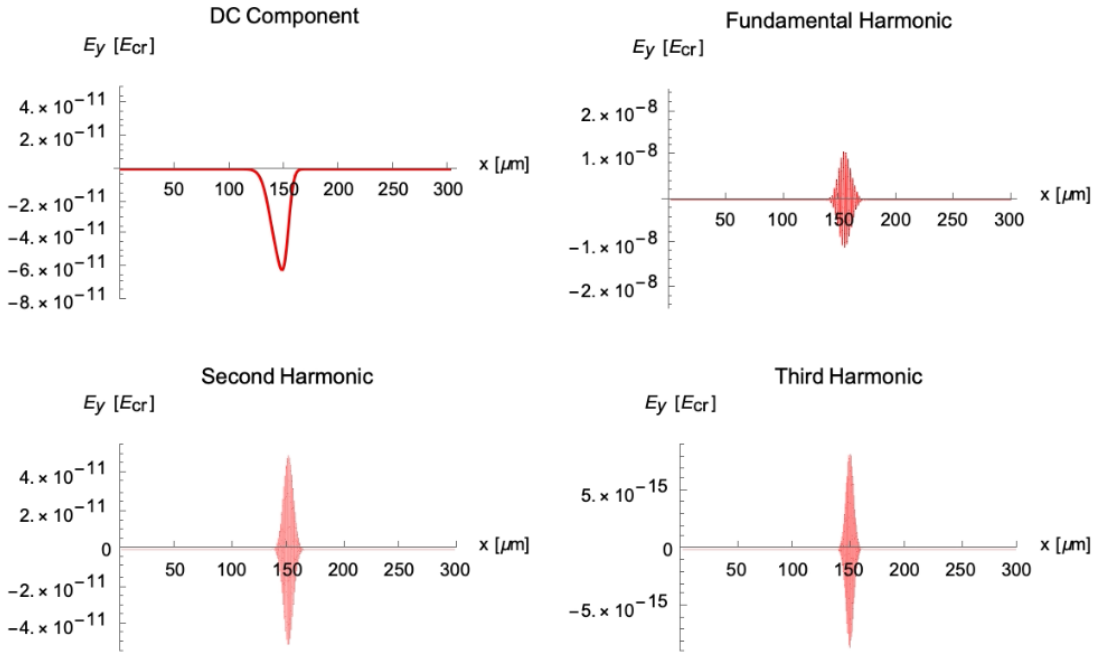


Figure 4.12: Thumbnail of animations available in the *Mendeley Data* repository [150] showing the time evolution of the various harmonics in 1D resulting form a pulse collision with the specifications of Table 4.4 with nonlinear four-photon and six-photon interactions.

Using the values from Table 4.4, the solid lines in Figure 4.13 are obtained. Those are employed to scrutinize the correctness of the simulation results.

A *Mathematica* [151] analysis of the analytical results for harmonic generation can be found in [152]. Animations of the arising and evolution of the various harmonics are provided in the *Mendeley Data* repository [150] (thumbnails and description in Figure 4.12).

4.4.2 Harmonic generation – simulation results

To put the focus on the nonlinear contributions to the generation of harmonics, each setting is simulated three times with varying combinations of interactions included: once including only the nonlinear effects of four- and six-photon contributions; once including only six-photon processes; and once excluding all nonlinear interactions, i.e., keeping only the linear vacuum. By subtracting the dynamics in the linear vacuum from the full dynamics, the higher order processes of the weak-field expansion are extracted. Furthermore, the simulations of six-photon processes permit to isolate their sole contribution, as is shown in Figure 4.11. It is a fruitful feature of the simulation code that the contributions of four- and six-photon diagrams can be turned on and off to make them separately visible.

In order to extract the amplitudes of the various arising harmonics and their time evolution, their respective frequencies have to be filtered in Fourier space and then transformed back to position space [145]. The amplitude is then obtained as the maximum norm. Its time evolution can be observed, in this case with a chosen resolution of a time step of 1 μm/c. The results for the zeroth harmonic, first harmonic, second, and third harmonic are displayed in Figure 4.13.

As can be seen in Figure 4.13, there is good agreement between the analytical approximation

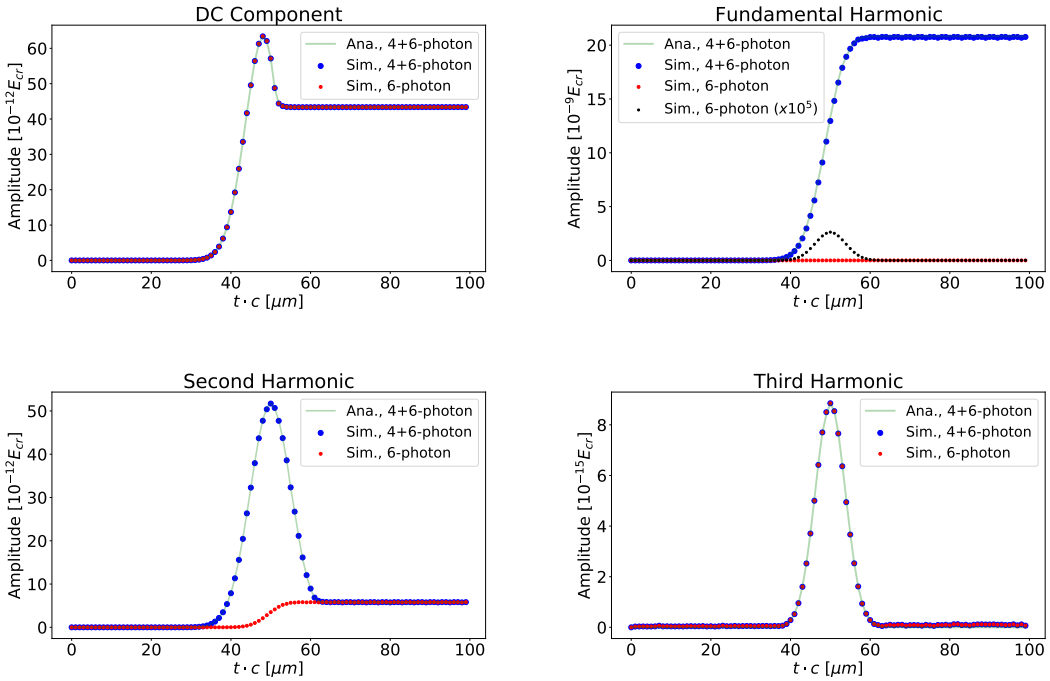


Figure 4.13: Amplitude evolution of the nonlinearly generated harmonics. Shown are the simulation results for harmonics caused by four- and six-photon processes combined (*blue dots*) and by only six-photon processes (*red dots*), all without the linear vacuum contributions. The analytical approximations summarized in (4.21)-(4.28) are underlying (*light green curve*).

and the simulation results. Small systematic errors are unavoidable in view of the back and forth Fourier transformation and slicing of frequency ranges. The mean absolute percentage errors of the simulation results are calculated to be less than 1% in the regions where the amplitudes are non-vanishing.

Asymptotic states are constrained by energy-momentum conservation, while the overlap state has a richer spectrum. The overlap spectrum becomes more pronounced and versatile when the pulses collide at a non-zero angle in higher dimensions. Some results are demonstrated in Section 4.5, where also the zero-frequency background restriction is relaxed. Signals, which are degenerate in the case of a non-zero-frequency pulse, split up in that case.

Simulations in higher dimensions provide a powerful means to analyze varying collision configurations. Situations that pose no further difficulty to the numerical code are considerably hard to cope with analytically. Employing simulations of the solver it is possible to track harmonic frequencies in time and space for scenarios of arbitrary pulse parameters – with the only restrictions posed by the applicability of the Heisenberg–Euler weak-field expansion and, of course, computational feasibility.

4.5 Higher-dimensional simulations

For simulations in 2D a special adaptation of 3D Gaussian pulses is used to model the diffusion behavior. The pulse is assumed to propagate along the z-axis. The widening of the beam with

respect to the longitudinal coordinate z is given by the waist

$$w(z) = w_0 \sqrt{1 + \left(\frac{z - z_0}{z_R}\right)^2} \tag{4.29}$$

with w_0 the waist of the beam at position z_0, where the amplitude is $1/e$ of the initial value. The cross-sectional area in 1D is a point, in 2D is a line, and in 3D is an area. The field intensity scales with $w_0/w(z)$ and with the surface area $\sim z^2$ in 3D. Lateral dispersion has to be taken into account for the 2D Gaussian pulses, where the surface scales as $\sim z$. Hence, the factor $w_0/w(z)$ appearing as prefactor in the Gaussian pulses gets a square root in the lower-dimensional case. The pulse can thus be written as

$$\vec{E}(r, z(t)) = A\,\vec{\epsilon}\,\sqrt{\frac{w_0}{w(z)}}\, e^{-(r/w(z))^2}\, e^{-((z-z_\tau)/\tau_z)^2} \cos\left(\frac{k\,r^2}{2R(z)} + \zeta(z) - kz\right) . \tag{4.30}$$

It is further defined

- the parameter A determining the peak pulse amplitude and the polarization $\vec{\epsilon}$;
- the distance to the propagation axis (here taken to be z) $r = \sqrt{x^2 + y^2}$;
- the wavenumber $k = 2\pi/\lambda$;
- the pulse width in z-direction τ_z (pulse duration) and the envelope center z_τ ; and
- the Rayleigh length $z_R = \pi w_0^2/\lambda$ as the longitudinal distance from z_0 at which the waist has increased by a factor of $\sqrt{2}$, which is contained in
  - the Gouy phase $\zeta(z) = \arctan(z/z_R)$, and
  - the radius of curvature $R(z) = z(1 + (z_R/z)^2)$.

The settings used for 2D simulations, which are confined to the xy-plane for propagation, are listed in Table 4.5.

For coaxial pulse collisions there is a correspondence to 1D with respect to the harmonics generated. Comparing a head-on collision in 1D (Figure 4.11) and 2D (Figure 4.14), a similar frequency spectrum is found at the different states in time. Contributions of different orders in the weak-field expansion are demonstrated again (Figure 4.15), with the same reasoning as in Section 4.4. The main difference stems from the fact that the two conceptually equal pulses have the same *non-zero* frequency ω_p. There is still a degeneracy of signals present by virtue of the equal frequencies.

A feature that is not present in 1D simulations is a lateral broadening of the pulses in frequency space. This effect arises during the interaction and remains, as can be seen in the simulation videos 4.20 [150]. The presence of a lateral beam profile of the pulses is a prerequisite to invoke this outcome. This results in outgoing signal photons with transverse momentum components, effectively yielding a diffraction effect [80]. Diffraction spreading opens up the opportunity to detect signal photons off the beam axis with a background free measurement. The scattering of polarization-flipped signal photons outside the forward cone of a probe beam may thus constitute an essential key ingredient for the detection of vacuum birefringence [83].

Correspondingly, from the position space point of view, the transversal momenta might imply a slight focusing of the pulses. This can be explained with the lensing effect of a power pulse that creates a refractive index influencing the propagation speed and direction, as detailed in Section

Table 4.5: Settings for 2D simulations with two conceptually equal Gaussian pulses. The wavelength is obtained via $\lambda = \pi w_0^2/z_R$. The Rayleigh length and waist are chosen such that the wavelength equals one micrometer.

Grid	Square Size	80 µm × 80 µm
	Lattice Points	1024×1024
Pulse 1	$\vec{\epsilon}$	(0,0,1)
	A	$50\times10^{-3}E_{\mathrm{cr}}$
	$\hat{k}$	(-1,0,0)
	λ	1 µm
	w_0	2.3 µm
	z_R	16.619 µm
	z_τ	20 µm
	τ_z	4.5 µm
Pulse 2	Same parameters as for pulse 1 but varying propagation direction and polarization	see Figures 4.14-4.22

4.2. With a lower refractive index at the outer waist regions of at least one of the pulses, light passing through the strong-field zone experiences a phase velocity change comparable to the one in a convex lens. Focusing of light by light is an intriguing topic to be further investigated with the help of adequate simulations with tailored pulse parameters.

Going beyond coaxial pulses by varying the collision angle and thereby lifting the degeneracy of frequencies of the harmonics, geometry effects with rich spectra in 2D simulations can be observed, see Figures 4.16–4.19 for perpendicularly propagating and colliding pulses as well as for a collision angle of 135°. Most signals vanish again in the asymptotic state. The off-axis contributions occur as consequences of the field spatio-temporal inhomogeneities [165].

The asymptotic harmonics, which can be seen in the right frames of the frequency plots, are caused by the self-interactions of the 2D Gaussian pulses [110, 165]. Time-resolving the processes (see the simulation video referenced in Figure 4.20) reveals that these harmonics arise immediately as the dynamics begin, directly after the initial configuration shown at the lower left of the figure, and thus already before the pulses overlap. Both the four- and six-photon processes contribute to the asymptotic signals with $|\vec{k}| = \omega_p/c$ and $|\vec{k}| = 3\omega_p/c$.

Six-photon processes contribute to all harmonics, in the overlap as well as in the asymptotic states. In the overlap states various merging processes become directly visible. A more precise analysis shows that the four-photon spectrum in the overlap states is even richer. Some signals, however, are of the order of accumulating numerical errors that depend on a number of factors, see the discussion in Chapter 6. Such errors appear, e.g., at the corners of the frequency plots for four-photon and six-photon processes, irrespective of the pulse alignment.

The signals of asymptotic harmonics generated by four-photon interactions are aligned on the initial propagation axes, while six-photon processes seem to generate a tiny off-axis twist. Note that the initial symmetry of the two-pulse system is thereby conserved.

Giving the pulses different polarization directions, the harmonics can be tagged. The frequency space of the simulations visualized in Figures 4.21 and 4.22 show again collision angles of 90° and

135°, but the pulse propagating from the left is now polarized along the E_y-direction. Hence, the shown polarization components can be identified with one of the two pulses, and thus also the harmonics.

Ultimately, in order to take into account all geometry effects, simulations have to be conducted in full three spatial dimensions. A demonstration of configurations similar to those in 2D as shown in Figure 4.16, in 3D yields results visualized in Figure 4.23. Pulses with frequencies differing by a factor two colliding collinearly produce the results of Figure 4.24. By virtue of the differing frequencies expressly rich harmonics spectra can be observed. The corresponding 1D case with a breakdown of the frequencies is demonstrated in the examples provided in the code repository [148]. Colliding the two pulses at an angle of 135° generates the harmonics shown in Figure 4.25. It can be seen that the weakest generated signals have about the magnitude of numerical artifacts.

A promising configuration for the detection of a nonlinear vacuum response is given by the prominent probe–pump laser pulse collision setup shown in Figure 4.26. While vacuum birefringence effects in 2D are simulated in a parallel project, even the employed supercomputing system described in Appendix A does not provide enough computing power to for such simulations in 3D as a consequence of the small probe wavelengths required to enhance the effect, c.f. Equation (4.11). Ways to overcome the obstacle of extremely large 3D grids are discussed in Chapters 7 and 8.

Simulation results in higher dimensions extend the horizon to yet undiscovered terrain that is particularly relevant for experiments. Predictions for light-by-light scattering in general should be possible in all conceivable interaction scenarios between probe and pump with the help of the solver [147].

The presented simulations in one spatial dimension could be cross-checked against analytical results and are in perfect agreement with the latter. On the other hand, the simulation results in higher dimensions have no direct analytical counterparts to compare with. A comparable accuracy can be expected.

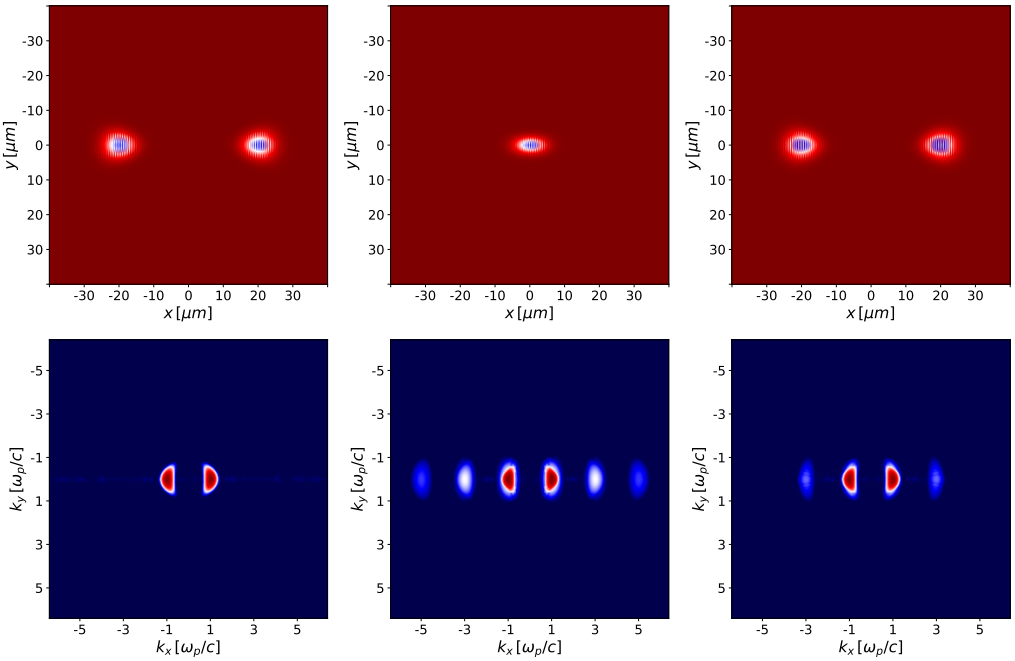


Figure 4.14: Coaxially colliding pulses with the same polarization. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: position space. *Bottom*: frequency space.

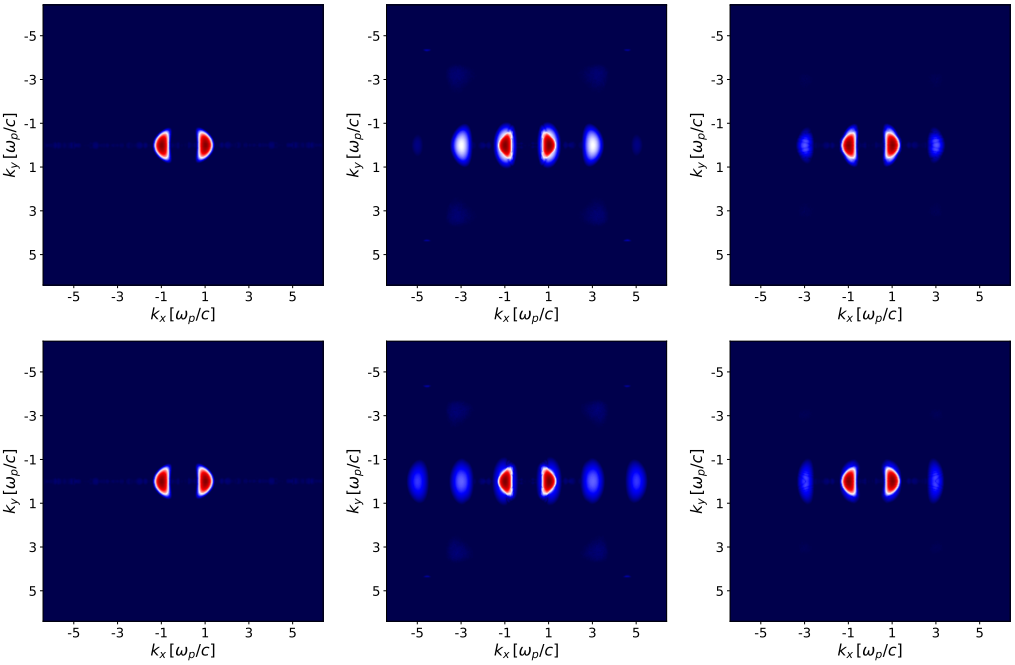


Figure 4.15: Frequency space of coaxially colliding pulses with the same polarization. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: only four-photon diagrams included. *Bottom*: only six-photon diagrams included.

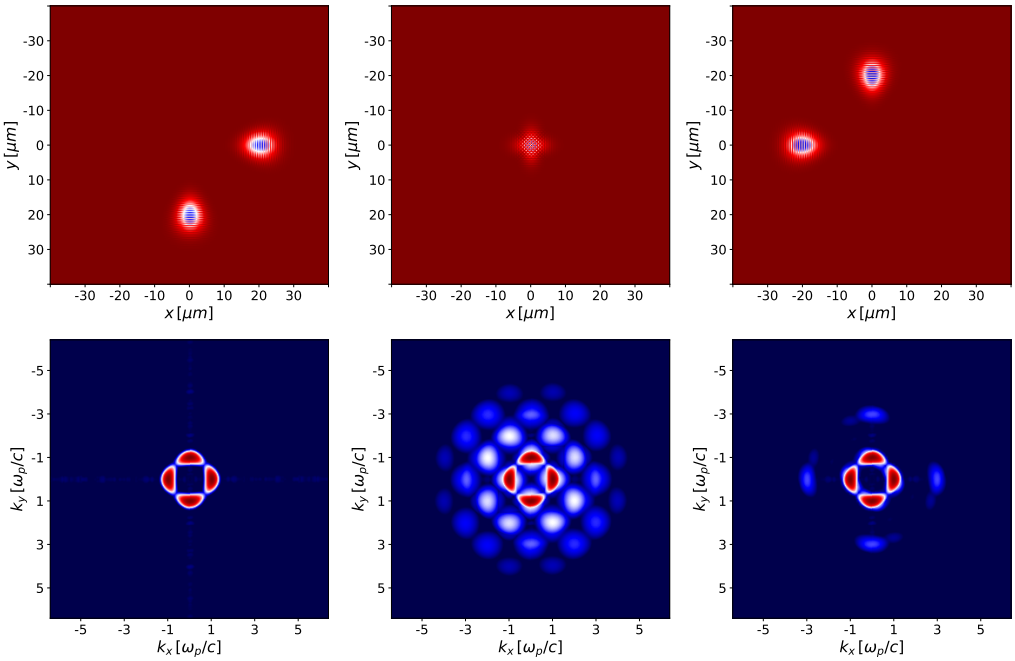


Figure 4.16: Perpendicularly colliding pulses with equal polarization. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: position space. *Bottom*: frequency space.

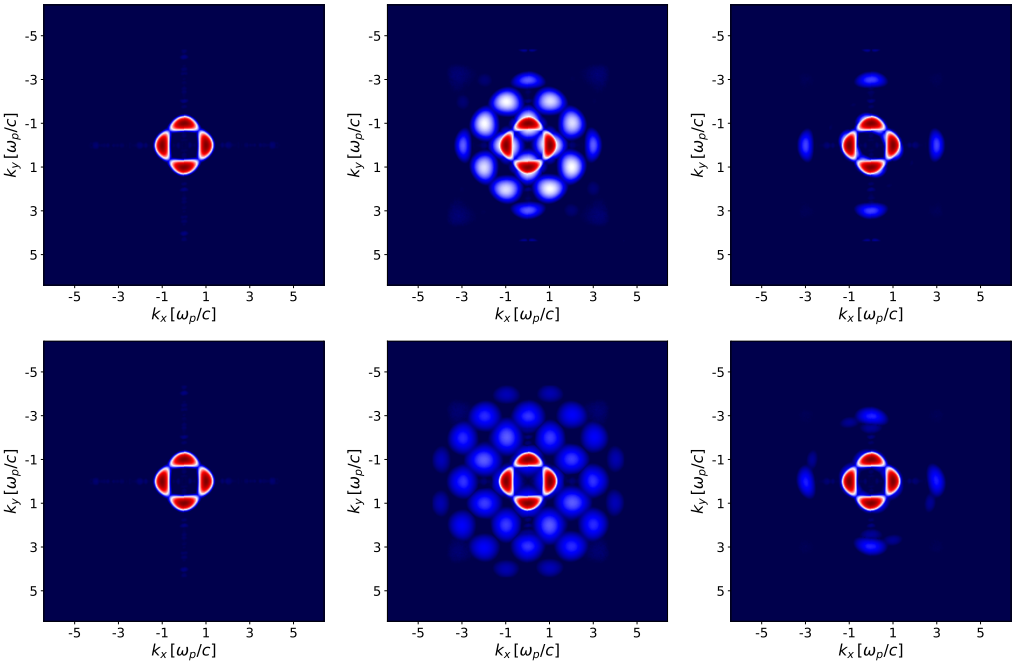


Figure 4.17: Frequency space of perpendicularly colliding pulses with same polarization. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: only four-photon diagrams included. *Bottom*: only six-photon diagrams included.

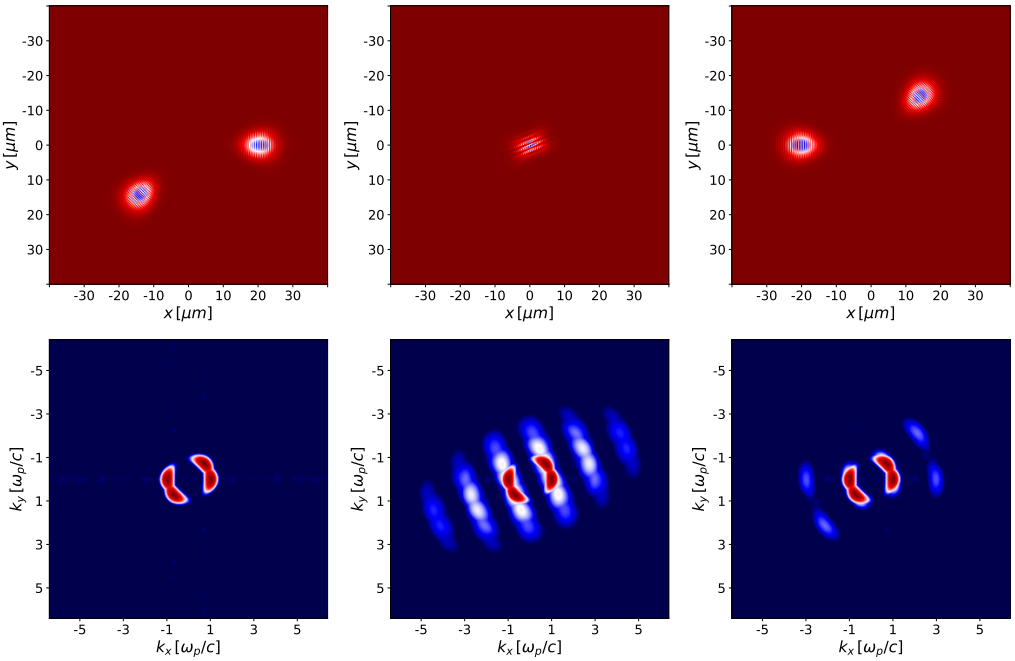


Figure 4.18: Pulses with the same polarization colliding at an angle of 135°. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: position space. *Bottom*: frequency space.

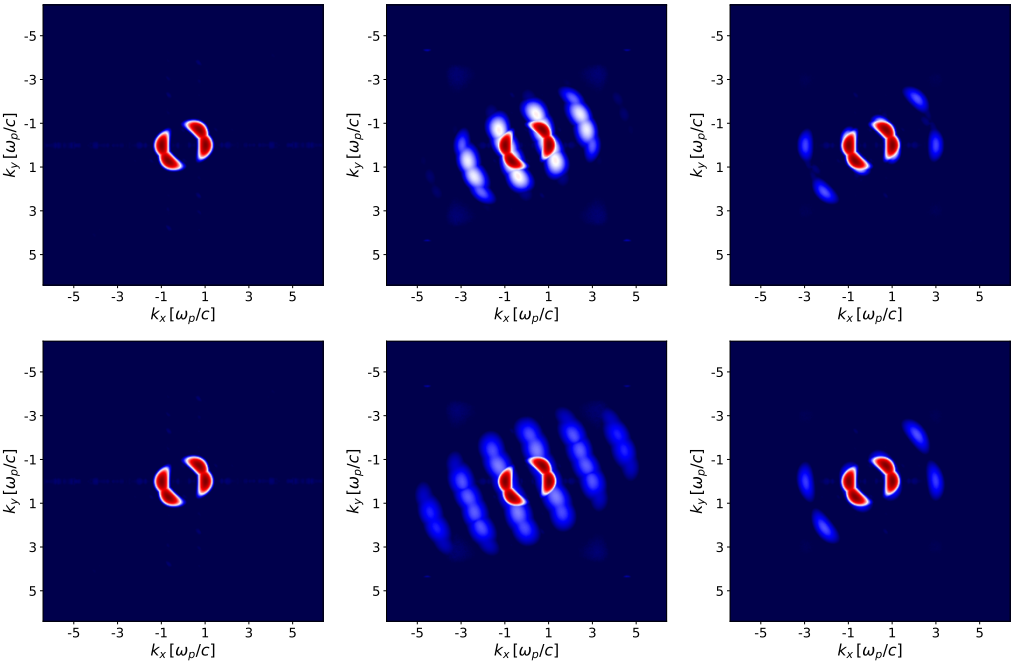


Figure 4.19: Frequency space of pulses with the same polarization colliding at an angle of 135°. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: only four-photon diagrams included. *Bottom*: only six-photon diagrams included.

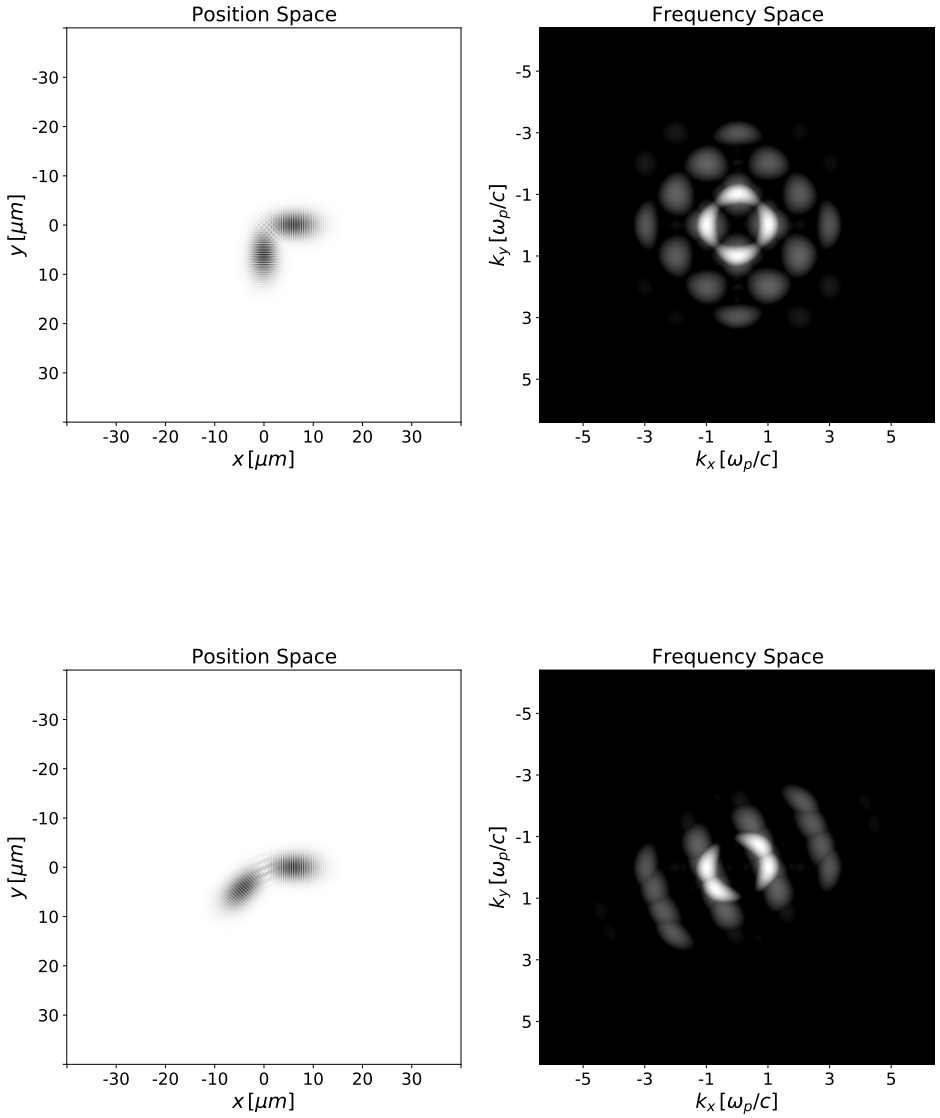


Figure 4.20: Thumbnail of simulation videos for two pulses colliding at an angle of 90° and 135° provided in the *Mendeley Data* repository [150]. 40 µm propagation distance are divided into 100 time steps. Note the self-interaction contributions for the higher-dimensional Gaussian pulses in contrast to plane waves. Gaussian pulses in one dimension amount to plane waves, but in higher dimensions there are self-interactions of a single Gaussian pulse since in that case $\mathscr{F} \neq 0$. Therefore, the self-interactions already at the first time step cause the slightly wiggling 3ω harmonics to become visible before the pulse overlap. From the overlap position onward, these harmonics are static in frequency space.

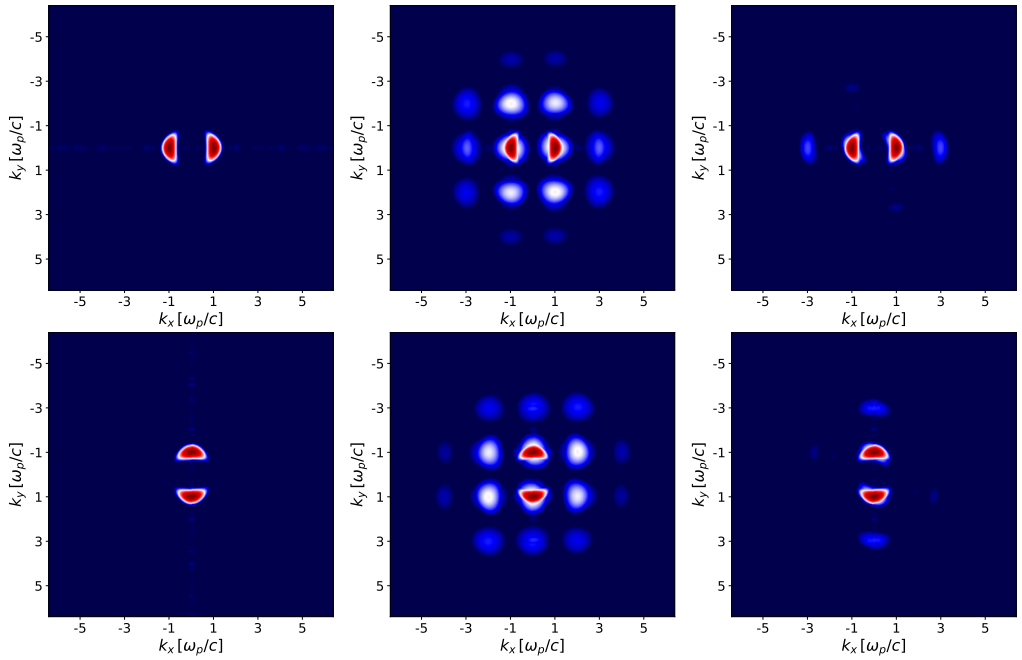


Figure 4.21: Frequency space of perpendicularly colliding pulses with orthogonal relative polarization. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: the E_z component is shown. *Bottom*: the B_z component is shown.

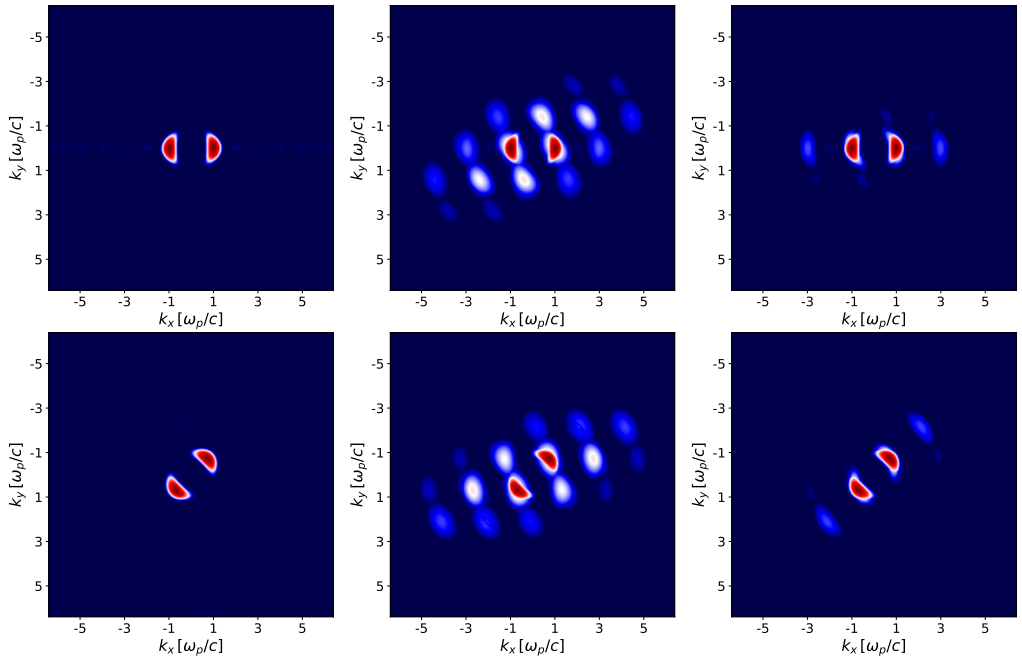


Figure 4.22: Frequency space of pulses colliding at an angle of 135° with orthogonal relative polarization. The left plots show the initial state, those in the middle the overlap state, and the right ones the final state. *Top*: the E_z component is shown. *Bottom*: the B_z component is shown.

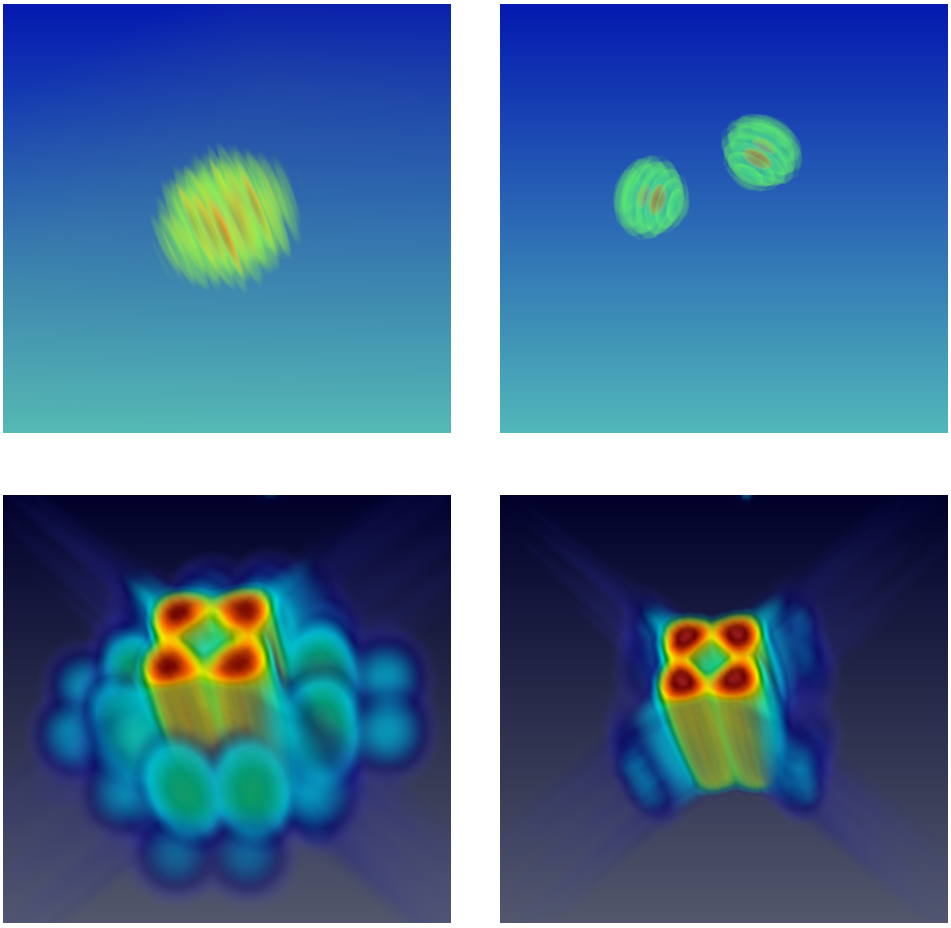


Figure 4.23: 3D simulation of two perpendicularly colliding Gaussian pulses. The visualizations on the left show the overlap state, the visualizations on right the final state. *Top*: position space. *Bottom*: frequency space.

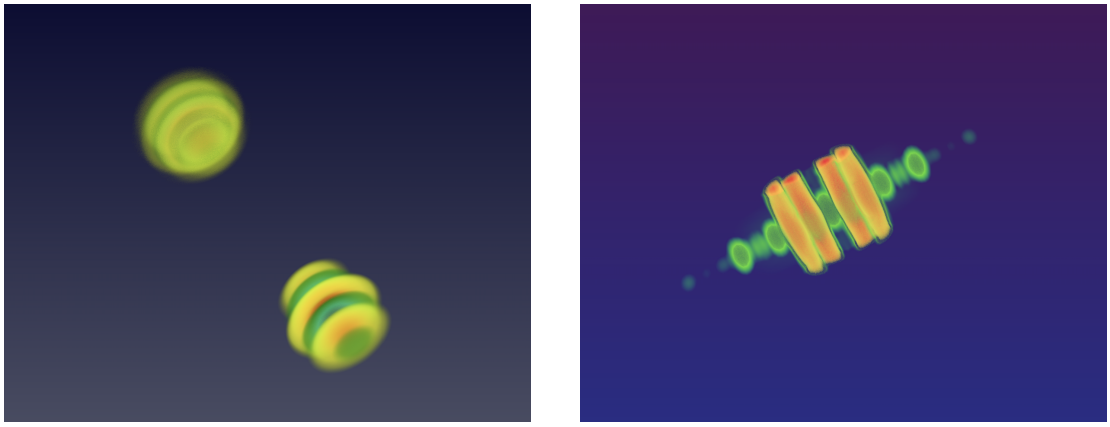


Figure 4.24: 3D simulation of two coaxially colliding Gaussian pulses with different frequencies. *Left:* initial pulse configuration. *Right:* harmonics spectrum at the overlap position.

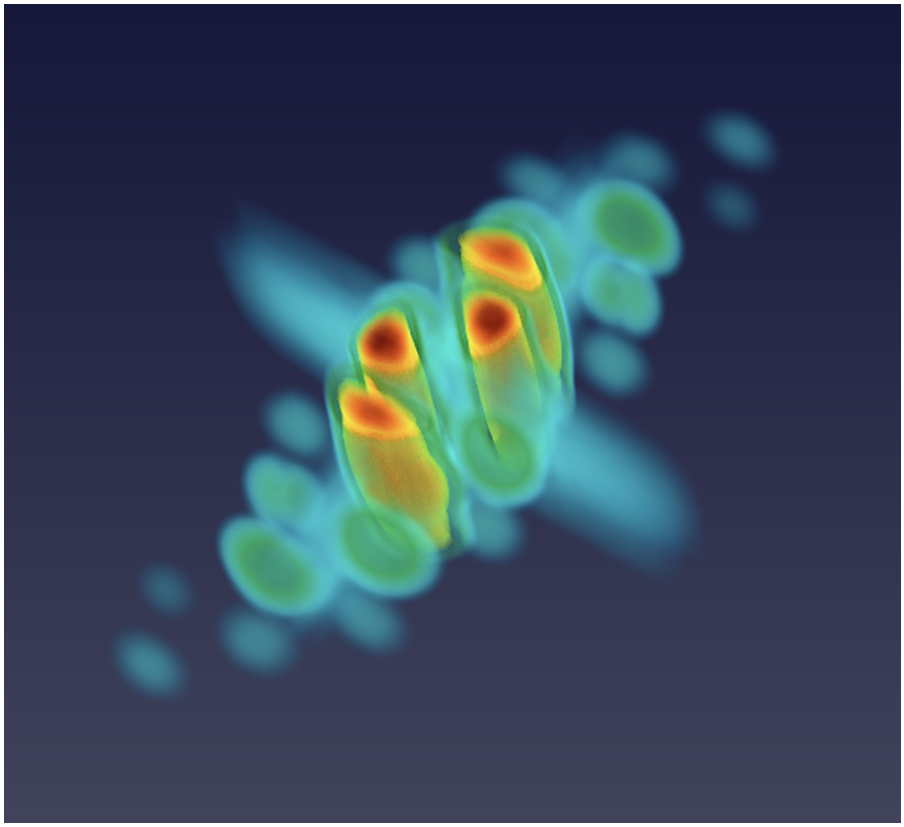
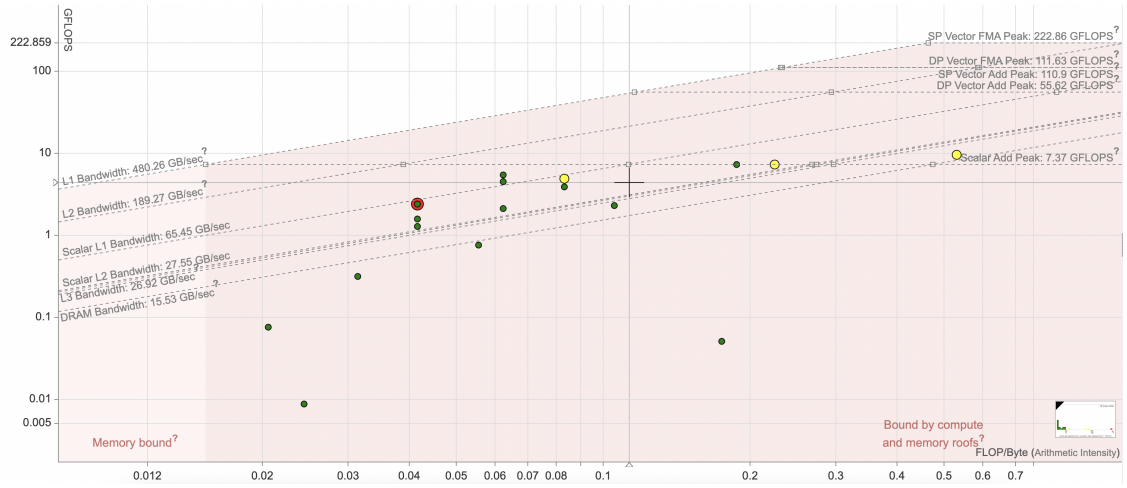


Figure 4.25: Rich harmonics spectrum in 3D. This spectrum is generated by the two Gaussian pulses of Figure 4.24 colliding at an angle of 135°. There are numerical remnants present on the coordinate axes, where the zero values in frequency space are located and small numerical errors accumulate. These can also be observed in Figure 4.23.

Figure 4.26: Probe–pump collision in a flat-box shaped simulation space in order to detect a polarization rotation of the probe pulse. The probe pulse on the left is tiny in comparison to the strong pump on the right.

Chapter 5

Performance Optimization



Roofline model

"*If you optimize everything you will always be unhappy.*" – Donald E. Knuth

5.1 Introduction

Computation has been optimized advancing from the first code versions of [111, 113] to [144, 145], but communication has not. While the code has been optimized with respect to smartly performed computations and memory access at the derivative operations in comparison to the very first versions, focus was not put on large scale performance.

The code version from [144, 145] already performed decisively better than the one of [111, 113] and enabled computations in 2D, but 3D simulations have not been conductible. One reason is the implementation of parallelism in the code. *MPI* communication was performed with the use of standard calls, not taking advantage of more advanced functionality. Moreover, the communication pattern was a self-constructed, hard-coded, and iterative procedure. No effort has been made to further increase the parallelism of the code.

This chapter describes how the solver is turned into a modern, large-scale, high-performance computing (HPC) application and as such is scalable and therewith able to simulate relevant pulse collisions in 3D. Performance optimization nowadays often means parallelization in various forms.

Performance optimization on all levels of an HPC system, the core-, node-, and cluster-level, is outlined. To this end, the code is vectorized and the memory access is improved both on the software side and by making use of hardware affinity. Most importantly, since the code is eventually limited by the *MPI* communication speed when distributed over a large number of processes, the *MPI* communication style is updated. An *MPI* virtual Cartesian topology is implemented to incorporate the actual structure of the simulation space and advanced techniques such as non-blocking communication replace the standard procedures for the exchange of boundary values between the patches of the lattice.

In order to combat communication overheads for simulations on a large number of nodes, the parallelization scheme has to be extended to a hybrid multiprocessing plus multithreading form. This approach is advantageous for massively distributed simulations in order to reduce both waste of memory and the communication overhead caused by a large number of *MPI* processes A further option would be making use of the *MPI-3* shared memory model.

The output data sizes of 3D simulations easily amount to several terabytes. Consequently, the file output needs to be efficiently parallelized and the format optionally changed from simple comma-separated-values to a lightweight and efficient binary form.

Parallelization techniques are, however, only half the way to a good performance, since optimization is highly system dependent. In order to improve the scalability of the solver and reduce performance variations, the software needs to match to the computer architecture and the hardware components. This goes under the name of “hardware affinity”. In the present case it is achieved on the one hand with the use of an *MPI* virtual topology fitted to the underlying physical problem in order to reduce communication resources. On the other hand, the hybrid parallelization with both multiprocessing and multithreading urges the user to explicitly “pin” the patches of the lattice to certain cores of the computer.

Benchmarks of the revamped simulator are provided.

Outline

In Section 5.2 high-performance computing systems, such as the one employed for the larger simulations conducted for the ongoing investigations, are introduced. Performance measurements of

the solver are conducted in Section 5.3 and approaches using parallelization techniques for performance optimization at various levels are detailed in Section 5.4.

5.2 High-performance computing

Clock rates are no longer increasing but Moore's law in the sense of transistor count still holds through parallelization [184]. A list of the 500 best performing supercomputers with respect to their ability to solve a set of linear equations, the *LINPACK* benchmark [185], can be found at the *TOP500* project [186], where latest news, databases, and statistics about the topic are available. The evolution of the list shows an extremely expensive and fast race for machines with the highest computing power.

The reason is that science, and consequentially technological development as a whole, today rely on computing. Computer simulations via numerical methods are pervasive in science to assist or replace experiments and make predictions on complex systems in order to gain insights into theoretical models. The highest-developed countries require the best-performing computing systems in order to maintain their leading positions in science. In its 59^{th} edition of May 2022 the exascale barrier (10^{18} floating point operations per second) was first officially broken on the high-performance *LINPACK* benchmark by the *Frontier* system at Oak Ridge National Laboratory (ORNL). Purported Chinese exascale systems have not been subjected to independent benchmarks. The breaking of that barrier is an achievement opening up all kinds of possibilities for the advancement of science and engineering.

Performance in HPC relies on using many processors simultaneously, where the single ones are commonly not particularly fast. Supercomputers are clusters of computers that put emphasis on speedy links between its constituent computers and fast memory access. Notebooks on the other hand rather have few but powerful cores, since for everyday tasks a large number of cores will hardly be necessary or usable.

The single self-contained computers in a cluster are called *nodes*, each equipped with an operating system. Nodes on the cluster can communicate with each other via node-interconnects. A node commonly contains a number of sockets consisting of compute cores, some levels of cache layers and local memory attached. Cores might be further divided into hardware threads [187]. A visualization is provided with Figure 5.1. All nodes further have access to shared parallel file systems.

Optimization on such composite systems is more intricate for the user, since both machine and code have to be tuned for parallelization. The interplay between large scale parallel code and the HPC system can become complex. The various levels to be taken into account are outlined over the following sections.

It is increasingly common to furthermore offload work onto accelerators like GPUs that can have significantly better performance than CPUs for certain problems. Since the HPC system used for the present work, the *KCS* system at the Arnold Sommerfeld Center for Theoretical Physics in Munich, does not contain accelerators, *HEWES* is still lacking GPU support. The *KCS* is described in Appendix A.

The future of computing is increasingly distributed and nowadays clearly affects also research and development outside of the scientific community. The global supercomputer market experiences vast growth rates through the high demand in new AI-supported technologies in many fields

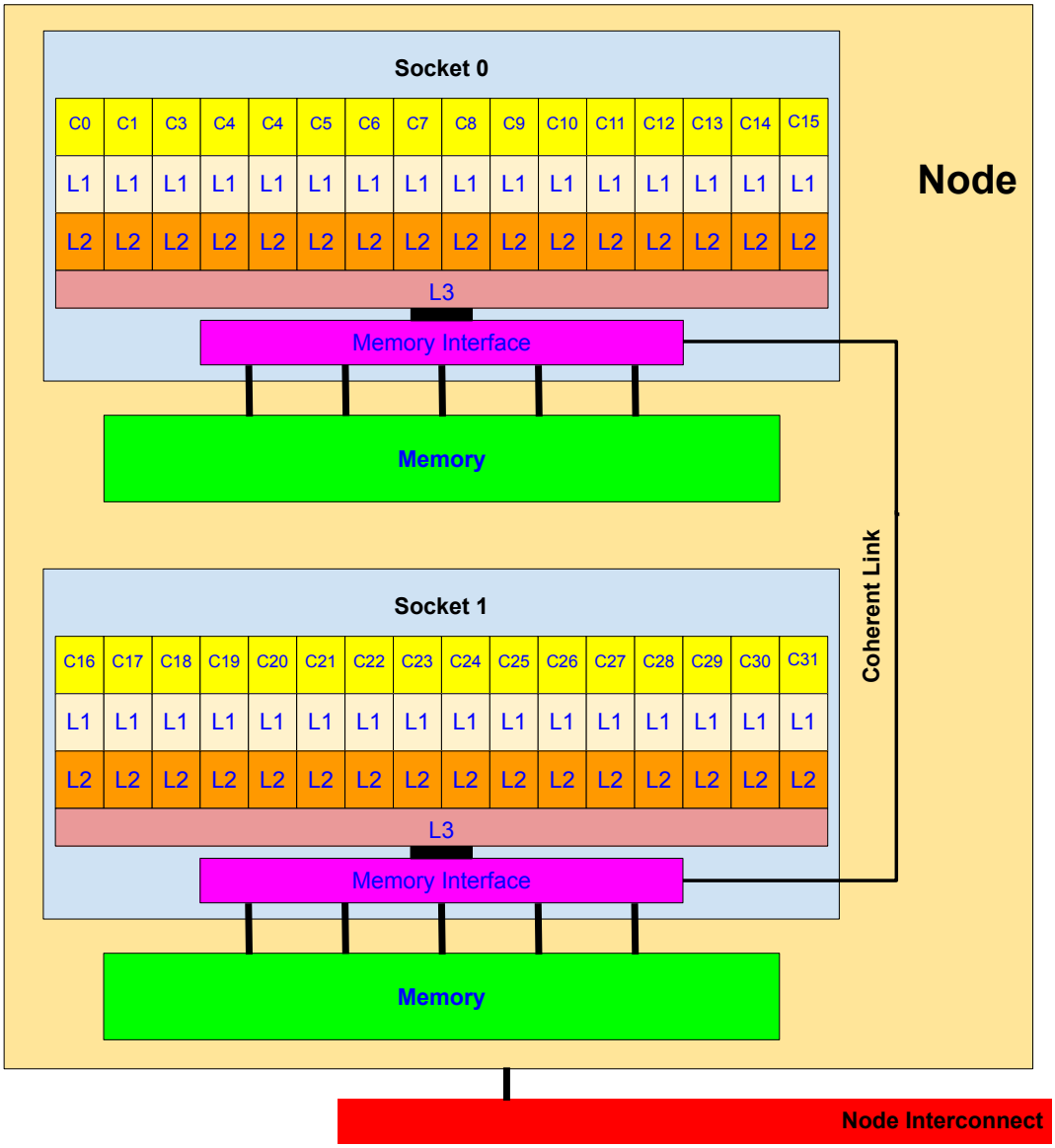


Figure 5.1: HPC architecture. The figure shows one node. It is connected to the other nodes on the cluster via the node interconnect. Here, the node consists of two sockets with an attached memory unit. Each socket contains sixteen cores (C), c.f. Figure 5.2, with their individual level one (L1) and level two (L2) caches. Cache levels are detailed in Section 5.2.1. The level three (L3) cache is shared among the cores of a socket. Caches are connected to the local memory via a memory interface. Cache coherence, discussed in Section 5.2.3, is maintained through a coherent link between the latter. This is the architecture of the *KCS*, see Appendix A.

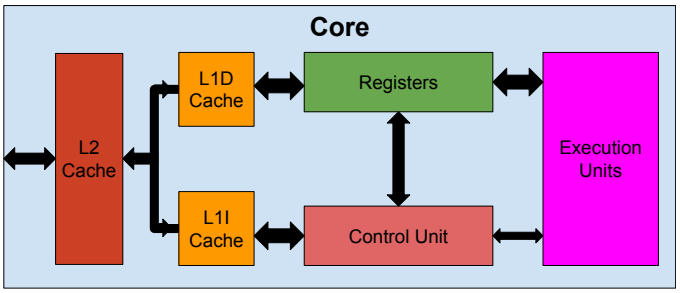


Figure 5.2: Rough structure of a compute core. The data is transported trough the caches to the registers and the execution units. There are different types of L1 caches, the data (d) cache and the instruction (i) cache. A control unit coordinates the instructions.

of research as well as in the entertainment industry with prominent technologies like chatbots and the metaverse. All these technologies require unprecedented computing power. Progress is pushed hard by big technology companies and governments. Even for the success of quantum computing, a conjunction with classical high-performance computing is indispensable.

Alike a self-fulfilling prophecy, use cases become even more versatile taking into account that supercomputers are no longer in privileged use of research institutes and a small number of large companies. Cloud computing has made its way into supercomputing and even into the top rankings in the *TOP500* list. There is a movement that can be considered as "democratization of HPC" [188] as a consequence of the digital transformation. Large scale process simulation and the creation of digital prototypes are no longer inaccessible to companies and startups that cannot afford an own supercomputing machine with HPC on demand in the cloud [189].

5.2.1 Memory access

Computer architecture is plagued by bottlenecks and in most cases data transfer is the number one limiting factor. Floating point operations per second (FLOPs) are, like the pixels of a digital camera, the lighthouse metric for quality. Yet, overall performance depends on which is slower, the processor or the data transfer. Disk, memory and network throughput all have been increasing way more slowly than the computing power in terms of FLOPs over the years [190].

Caches

Intermediate levels of cache layers, normally three, are built into the CPU chips to act as buffers between memory units and the processing unit with the purpose to remedy the slow memory performance, c.f. Figure 5.1. Instructions are performed by the cores only on data in their registers, which are connected only to the top level cache. See Figure 5.2 for a rough outline of the structure of a core.

Caches are faster in transfer to the CPU register but have smaller capacities. In order to increase the throughput, algorithms make sure that caches are holding those data items the processor is most likely to ask for next. However, these algorithms are not perfectly apt for all applications and rely on, e.g., a last-recently-used policy. There are various styles of implementations of the cache levels both in terms of design and algorithm [187].

Usually, there are three levels of cache layers. The latencies of level one (L1) to level three

(L3) cache range from 10^{-9}s to 10^{-8}s, with bandwidths of 10^{12} to 10^{11} bytes per second [191]. The register only communicates directly with the L1 cache. Further information on the caches of the *KCS* system employed for large simulations of the present work are given in Appendix A in Listing A.3 and Figure A.1.

The common way of designing the layers is such that L1 and L2 caches are private to the cores and a much larger L3 cache is shared among the socket. This yields low latencies on the one hand and avoids data contention between caches (different cores overwriting cache data of others) on the other. Shared data in L3 caches among cores improves the effective bandwidth. Everything in the L1 cache is also contained in the L2 cache, etc., down the hierarchy.

Registers can only access the level one cache in the form of complete cache lines of usually 64 bytes size. Accessing a byte anywhere in a line fetches the whole line. This reduces latency, but has the downside that crossing a cache line incurs another whole access operation. Every unused byte in the line forms a *loss* event. In consequence, it is crucial to exploit the whole cache line in order not to waste bandwidth.

Locality

Oftentimes the main purpose of optimizations is to organize fast memory access through the cache layers. This is achieved by managing the local and temporal locality of data, since caches most likely hold lately used and nearby data. If the processor finds the data item it wants right at the time in its cache, this is called a *cache hit*. The important term *cache miss* on the other hand expresses that the load or store instruction does not find the desired data in the first cache level and thus cache line transfers between cache levels have to be performed, or even worse, from the main memory [187]. To fetch new data into the cache, other data have to be evicted. If the searched data lie in the main memory of another socket or node this can lead to a severe drop in performance, as explained in Section 5.2.3.

Everything about memory is about locality. Caching effects can be hard to predict but there exist dedicated cache analyzers like *Cachegrind* and *Callgrind* from the *Valgrind* toolset [192]. Improved memory management involves [191]

- blocking of data, i.e., splitting data structures such that access sweeps are small, in order to exploit cache reuse opportunities;
- easing the hardware prefetcher's work with simple to predict access patterns in order to help hide memory latency; and
- copying to reduce chances of data eviction from cache.

Sometimes, memory access instructions can be reduced by increased (perhaps redundant) computation. Memory access analyses for *HEWES* are performed in Section 5.3.4.

5.2.2 Core-level

On the core-level, in order to leverage the increasing transistor count, multiple levels of parallelism are available [187].

Instruction-level parallelism

Instruction-level parallelism such as pipelining and superscalarity make use of the fact that instructions are split into smaller, simpler tasks that each take the same amount of time, one compute cycle. Every computation is split into single tasks like multiplying mantissas, adding exponents, etc. Moreover, the instruction execution itself can be split into fetching the instruction from cache, decoding and finally executing it.

The efficiency of pipelining stems from the fact that these tasks can be overlapped for different instructions and data. With different units assigned to different instructions like adding or multiplying, loading or storing, the instruction stream is parallelized. In this way all stages of functional units are kept active throughout the processing time.

Modern processors are capable of executing more than one instruction per cycle. Superscalarity means performing multiple instructions per cycle.

Exploiting instruction-level parallelism is mostly carried out by the compiler or processor.

Single instruction multiple data

Producing one result for each single arithmetic instruction is called *scalar operating*. Nowadays, register sizes are large enough to hold more than one floating-point or integer number of a given precision. Single instruction multiple data (SIMD) *vectorization* leverages the wide register size by executing the same kind of operation concurrently on multiple data. Vectorization is crucial because often enough the modern arithmetic and floating point units of the cores are underutilized.

A modern fused multiply-add vector unit with a 512-bit register can operate on chunks of sixteen single-precision (SP) or eight double-precision floating-point numbers, or eight 64-bit or sixteen 32-bit integers within 512-bit vectors at a time. on the *KCS* system at LRZ, the full size of the registers can be exploited using the Advanced Vector Extension 512 instruction set architecture (*Intel® AVX-512 ISA*) for the *Intel® Xeon® Gold 6130* CPUs of type *Skylake*, see Appendix A.

It has to be kept in mind that the bandwidth limits vectorization capabilities. Full SIMD benefit is only possible for data in the L1 cache since for lower levels the data transfer is too slow. SIMD vectorization, which is detailed in Section 5.4.1, is achieved with compiler options and directives.

Simultaneous multithreading

Simultaneous multithreading (SMT), or hyperthreading, divides one physical core into two logical cores with separate use of registers, pipelines and control units. This can oftentimes be advantageous, since the data loading from cache is due to latency too slow to keep the execution units occupied, which comes at the cost of idle cores. Employing SMT, the caches are shared between two logically distinct registers and control units, while the number of execution units stays the same. There are different forms of implementation of SMT.

As is the case for many applications, the use of hyperthreading showed to have no positive influence on the speed of execution in the present project. This is probably due to the competition of threads for data on the shared caches, leading to more cache misses.

General multithreading lets a multitude of logical cores perform computations piece-wise in parallel. This is detailed in the Section 5.4.2.

5.2.3 Socket-, node-, and cluster-level

Cores share a memory bus for access to the main memory, with the cache hierarchy in between. Oftentimes, there is a local memory unit attached to each socket. A socket thus forms a shared memory island, or locality domain, on the cluster on which multithreading can be ideally employed, e.g., with the help of an API like *OpenMP* [173]. Multithreading means the parallel execution of code on a number of cores/threads using the same memory address space, further discussed in Section 5.4.2.

Multiprocessing on the other hand is the parallel execution of code with separate memory per process. Parallelization libraries for multiprocessing, e.g., implementations of the Message Passing Interface (*MPI*) [172] can be used within a socket, a node, or the whole cluster. This is further discussed in Section 5.4.3.

Today's nodes are cache-coherent non-uniform memory access (ccNUMA) nodes. The terminology implies that the cache updates are synchronized and the memory bandwidth and latency depends on which memory unit is accessed by which core. Non-uniform memory access is caused by different path lengths for a core to access memory units located at different physical positions, see Figure 5.1. Accessing data in the various caches can likewise incur vast extra work through the cache coherence mechanism. Copying data up and down the memory hierarchies, potentially over the coherent link, in order to ensure synchronization can become expensive operations.

On the cluster-level, where memory is anyways distributed, it is crucial to minimize access on remote memory units and resource contention. This is because remote memory access bandwidth is clearly the lowest and can form a severe bottleneck. Every core can access any memory location, but the bandwidth and latency clearly depend on the physical position. The effective memory bandwidth can be increased by distributing working cores among a number of sockets. The effective cache sizes are thereby increased, since the highest cache level is commonly shared on a socket.

Hardware affinity

Topology optimization is one way to optimize bandwidth usage by reducing the communication over the slower inter-node/socket connections. On ccNUMA node systems, it is also instructive to avoid resource contention on shared memory paths. Contention is ubiquitous in the first place on the node interconnects but also on the coherent links.

Code can be bound to certain hardware threads by *pinning* the processes and threads explicitly to logical cores of the system. In fact, *OpenMP* and *MPI* programs should never be run without explicitly binding processes and threads to logical processors. Circumventing slow data paths is key. Techniques for the positioning of data and processes/threads are discussed in Section 5.3.1, relevant for Section 5.4.4.

On top of that, performance can vary significantly without pinning instructions, since some physical process/thread localizations are up to chance. The system does not simply guess the best assignments. The default behavior of ccNUMA systems is the *first-touch* rule: Data get mapped onto the local memory of the processor that first writes them. This is often problematic in the case of multithreading if data initialization is not parallelized.

Accordingly, *parallel first-touch* has to be explicitly implemented when using multithreading along with multiprocessing over several ccNUMA domains. This means parallelization of the initialization processes such that data are written to the local memory where it will be required for

further computations. Parallel first touch is detailed in Section 5.4.4. If NUMA balancing – automatic page migration – is activated on the system, as is the case on *KCS*, the system monitors the memory access during the process runtime and tries to move memory pages to the threads that are accessing them. One should definitely not rely on that, the default behavior is rather tentative and slow [191].

At extreme scales, pure *MPI* codes require a too large number of *MPI* processes and nodes. This causes a waste of memory since every process requires its own copy of data. To conclude, a well-designed hybrid approach with multiprocessing and multithreading is indispensable for extreme scaling.

5.3 Performance measurement

Measuring is better than guessing and performance is implementation-, problem-, and machine-dependent. So instead of a hope-for-the-best trial-and-error process, deficiencies ought to be systematically analyzed with state-of-the-art tools and techniques in order to gain a clear perspective for the relevant and suiting optimization tweaks.

In order not to waste resources, performance measurements are mostly performed on short "debug" simulations of only a few time steps. Moreover, without explicit pinning, runtime variations are quite high due to the unspecified distribution of the processes over the cluster. Benchmarks should furthermore always be run on exclusive systems, to this end the used nodes should not be otherwise occupied. This could however be to the detriment of other users and long queuing times, see Appendix A.

A combination of different methods, tools and techniques is employed. These involve

- retrieving system information and controlling hardware affinity with *LIKWID* [193]; and
- performance analysis on the cluster-level with the trace-based *Scalasca* toolset that is based on the *Score-P* instrumenter and measurement system and the graphical analysis report exploration format *Cube* [194, 195].

Furthermore, *Intel*® tools from the *Intel*® *oneAPI* toolkits [196] are used, namely

- the *Intel*® *Advisor* as a vectorization memory access analysis toolset on the core- and node-level;
- the *Intel*® *VTune*™ *Profiler* for profiling and evaluation of performance characteristics on the node-level; and
- the *Intel*® *Trace Analyzer and Collector* GUI to profile, analyze, and visualize *MPI* applications on the cluster-level.

The crucial metric here shall be time to success, leaving aside an economic occupation of resources. There are generally two ways to make things faster: Do the same amount of work faster or do less work. Making processing faster thereby often means more parallelization in some form. In that case, anything that takes time is only relevant for optimization if it lies on the *critical path* of execution. That means, overall performance in a parallelized application is determined by the weakest link. The critical path is responsible for the wall clock time the execution of the application takes.

Because of inevitable operating system noise, the execution time is often not deterministic. Averaging runtimes and taking care of hardware affinity are ways to mitigate runtime variations.

The performance is generally limited by [190]

- b_s: the maximum saturated bandwidth to move data from their memory location to the CPU register [Byte/s];
- I: the intensity of work for each byte moved [FLOP/Byte], i.e., the number of times the same float is reused in some set of floating-point operations; and
- P_{peak} : the theoretical maximum performance of the CPU [FLOP/s].

Besides the hardware, b_s also depends on locality of data because the bandwidth is highest for the caches. I depends on the structure of the algorithm and data layout. P_{peak} is affected by the clock frequency and the number of cores, but also by vectorization. The effective performance for a function/loop is thus given by $P = \min\left(P_{\text{peak}}, I \times b_s\right)$. These metrics are captured graphically in the *roofline model* discussed in Section 5.3.4.

5.3.1 Scaling

Parallelization of the algorithm

To the end of faster computation and scalability the code is parallelized in multiprocessing form at the outer level. It is expedient at this stage to give an overview of the techniques. The details have been omitted until this point but are important for this chapter.

A Cartesian domain decomposition of the lattice is performed, allowing individual compute cores to process patches of the lattice. The number of these sub-lattices is determined by the user who may subdivide each dimension of the lattice into a number of equally sized parts, see also the README file in Section 3.4.

The finite differences scheme used to discretize the spatial derivatives requires values from neighboring sub-lattices when it is applied at the boundaries of a patch. For this purpose, ghost cells, or halos, are placed at the boundaries and updated via message passing, making use of *MPI* [172], see Figure 5.3. The required depths of the boundary layers depend on the stencil order as described in the README in Section 3.4.

As a result of the sub-lattices forming a Cartesian grid, the communication scheme is conveniently implemented in an *MPI* virtual Cartesian topology, detailed in Section 5.4.3.

Output to *CSV* format is written to one file per process (patch), whereas output in binary form is written to one single file per step, making use of *MPI-IO*. This is discussed in more detail in Section 5.4.5.

Strong and weak scaling

Since efficient parallelization is paramount for expensive 3D simulations, strong and weak scaling tests proving the parallelization capability are conducted. For the tests no output data are generated, yet the *MPI-IO* variant is highly efficient anyways, as demonstrated in Section 5.4.5.

It is important to recall elements of the architecture of the employed high-performance computing system. *KCS* contains sixteen cores per memory locality domain (here a socket) and two

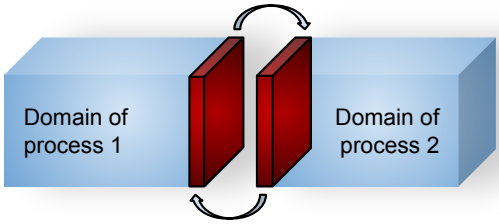


Figure 5.3: Ghost cell exchange. The blue boxes represent sub-lattices (patches), batches of the interdependent grid data, each being processed on single cores. The processes (compute cores) exchange their boundary layers' values with the neighboring processes. The boundary regions, whose required depths depend on the order of the finite differences scheme, are indicated in red. This exchange is performed sequentially for each dimension. Here it is shown for one boundary region between two processes.

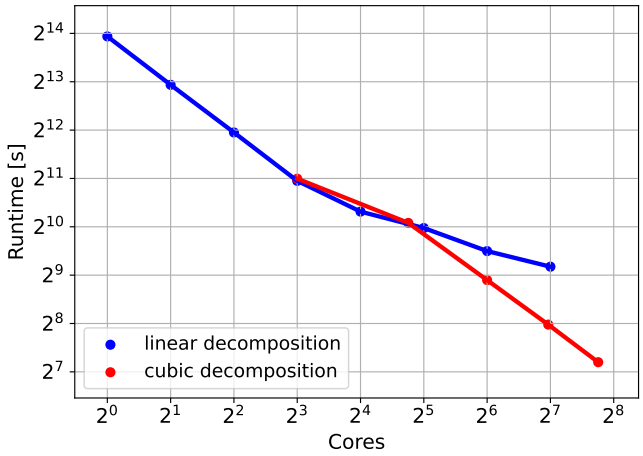


Figure 5.4: Strong scaling test for a relatively cheap simulation in 3D. The overall lattice keeps its size and is split into smaller and smaller patches to be processed by single cores. The runtimes nearly halve for a doubling of cores for up to eight cores. The lattice is sliced into sub-lattices in one dimension (*blue line and dots*). The runtime speedup decreases where one memory domain (socket) is fully occupied with sixteen cores. The code for this benchmark configuration is memory-bound as this typical behavior at the socket saturation shows. The intra-socket scaling is not linear since the memory as a shared resource on the socket does not scale along with additionally used cores. Slicing only in one dimension leads to a communication overhead when the sub-lattices become too narrow. This is remedied in this scenario by an equal slicing in every dimension such that the patches are cubic (*red line and dots*). The scaling across nodes is then again optimal. Each setting ran twice and is averaged in order to take into account minor runtime variations.

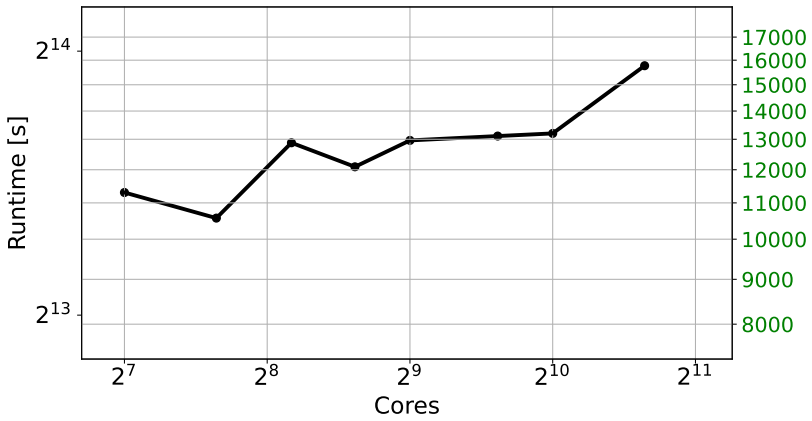


Figure 5.5: Weak scaling test for a simulation such as to obtain 3D results shown in Section 4.5. While the strong scaling test uses a fixed overall lattice, a weak scaling test is obtained by keeping the patch size constant, while increasing the size of the overall lattice with the number of patches and compute cores. Hence, the problem size is increased along with the number of parallel workers. For such 3D simulations the code becomes more and more communication-bound. Runtime variations and scaling issues are mainly as a result of data transfer via the node interconnect and resource contention thereon. These expensive simulations ran only once each, in order to save resources for the community.

sockets per node, see Figure 5.1 and Appendix A. The results of the scaling tests are visualized in Figures 5.4 and 5.5.

The former case of the strong scaling test, which was performed on an older code version, demonstrates that an efficient decomposition of the lattice is important. At the scales explored in this test, an optimal scaling performance can then be achieved. More tests prove that cubic patches form the most efficient decomposition. This is discussed in Section 5.4.3.

In the latter case of the weak scaling test, which was performed at larger scales with an already optimized code version, runtime differences are mainly due to communication among nodes that are farther apart on the cluster and resource contention on the interconnects. No runtime averaging was performed in this case. The memory required per node was about 30 GB for each simulation. The weak scaling can be considered satisfying between 400 and 1000 cores. A steep increase in runtime is observed at the most expensive run shown, fully occupying 50 of a total of 153 nodes on *KCS*.

The decisive effect of the distribution of participating nodes on the cluster was impressively demonstrated with another, similar simulation. On a compact set of nodes the runtime was three and a half hours and was thereby nearly one hour shorter than a repeated run that was allocated on farther apart nodes.

Hardware affinity control

Almost all performance features depend on the system topology and when using multiprocessing and multithreading further on the process/thread placement, also called *pinning* of processes/threads to certain compute cores. In the optimal case, this ensures that the data are equally distributed among the involved processors and their local memory.

Pinning is performed as a first step before any further modifications in order to reduce the

runtime variations during testings. This is necessary for reliable benchmarks that should always be accurate, deterministic, and reproducible. The *LIKWID* performance tools provide the means to control the pinning of *MPI* and *OpenMP* programs. For a hybrid *MPI* + *OpenMP* model, affinity is conveniently controlled with the compiler wrapper `likwid-mpirun` and its options. With *OpenMP* there are also environment variables available to control hardware affinity as well as some *Intel*® specific variables, see Appendix A with Listing A.5 and the documentations.

An important point to make is that the most relevant pinning, namely that of the nodes, is most of the time not performed by the user. On HPC systems, usually batch job schedule software is responsible for the resource allocation. The user's influence with respect to the node distribution is often very limited, which explains the issue with large runtime variations mentioned in Section 5.3.1. The scheduler employed on the *KCS* is outlined in Appendix A.

Outlook on bottlenecks

The question is what else limits performance and scalability. Runtimes and bottlenecks vary strongly with the particular setting for the problem under consideration. Consequentially, it is hard to give universal benchmarks and provide general scaling properties.

Bottlenecks for various configurations are outlined over the following sections. Analyses show that the code is – for the presented standard simulations used to produce the results of Chapter 4 – compute/memory-bound in 1D, memory-bound in 2D, and communication-bound in 3D.

In order to improve the performance of simulations in lower dimensions (1D and 2D), more vectorization and memory locality are enforced. At large scales, which are most important and where *MPI* communications play the biggest role, speedups are obtained through a reduction of the communication overhead.

The latter can be achieved by lowering the order of the numerical scheme, because the required depth of the ghost layers decreases, e.g., from seven to three from order thirteen to order four, as can be seen from the stencils in Section 2.3. As mentioned in the REAMDE file in Section 3.4, the stencil order is on the other hand not decisive for computation speed alone. To further ease the messaging pressure and memory requirements at large scales, where the pure *MPI* scaling seems to deteriorate, additional parallelism by multithreading is used on top of the multiprocessing with the help of *OpenMP* [173]. Section 5.4.4 is devoted to the latter approach.

5.3.2 Code hotspots

Profiling and performance snapshots

The *Intel*® *VTune*™ *Profiler* (former *Intel*® *VTune*™ *Amplifier*) for node-level performance evaluation performs hardware counter sampling, i.e., numbers of specific hardware-related actions are counted in order to determine, e.g., memory and CPU activity. A *profile* is a summary of events over an execution interval. The tool is applied here to give an idea about the bottlenecks of different kinds of simulations and time-intensive hotspots in the code.

Minimally invasive first analyses can be performed with *Intel*® *VTune*™ *Profiler* Application Performance Snapshots. The *Intel*® *VTune*™ *Profiler* identifies the process-communication to be the limiting factor for a test simulation in 3D. It is accordingly classified as *MPI*-bound. Figure 5.6 shows an *Intel*® *VTune*™ *Profiler* Application Performance Snapshot.

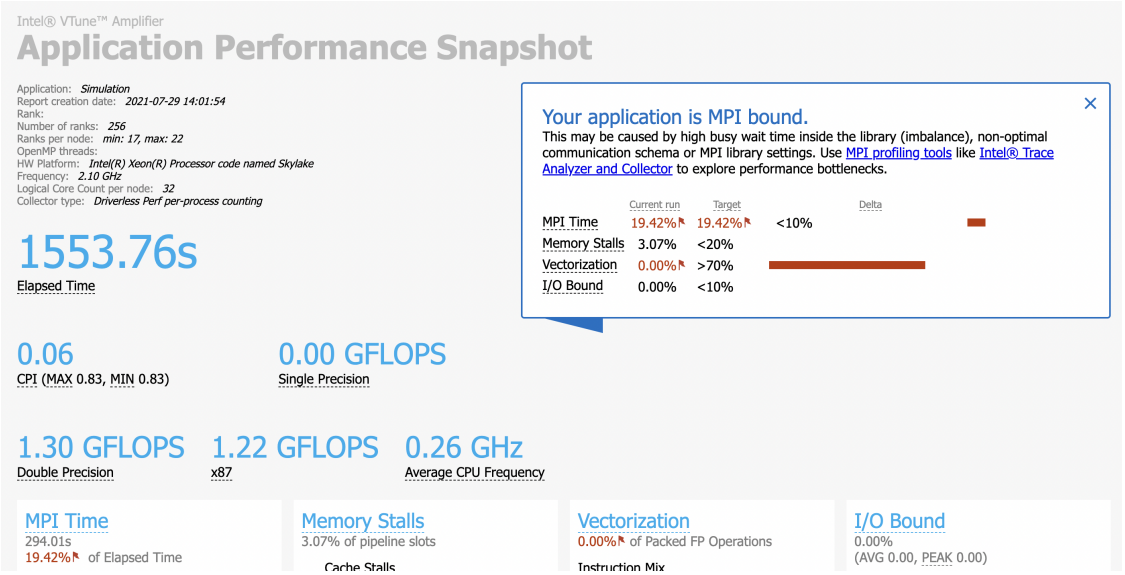


Figure 5.6: *APS* report for an *MPI*-bound 3D simulation on 256 cores.

Smaller simulations in 1D and 2D require less processes, occupy fewer cores, and thus the overall communication load is smaller. Principally, these simulations find enough memory space on a single node and can in principle be run even without *MPI* at all. These simulations are thus rather memory-bound or even compute-bound, see Figure 5.7 for an *Intel® VTune™ Profiler* Application Performance Snapshot of a 2D simulation on four cores.

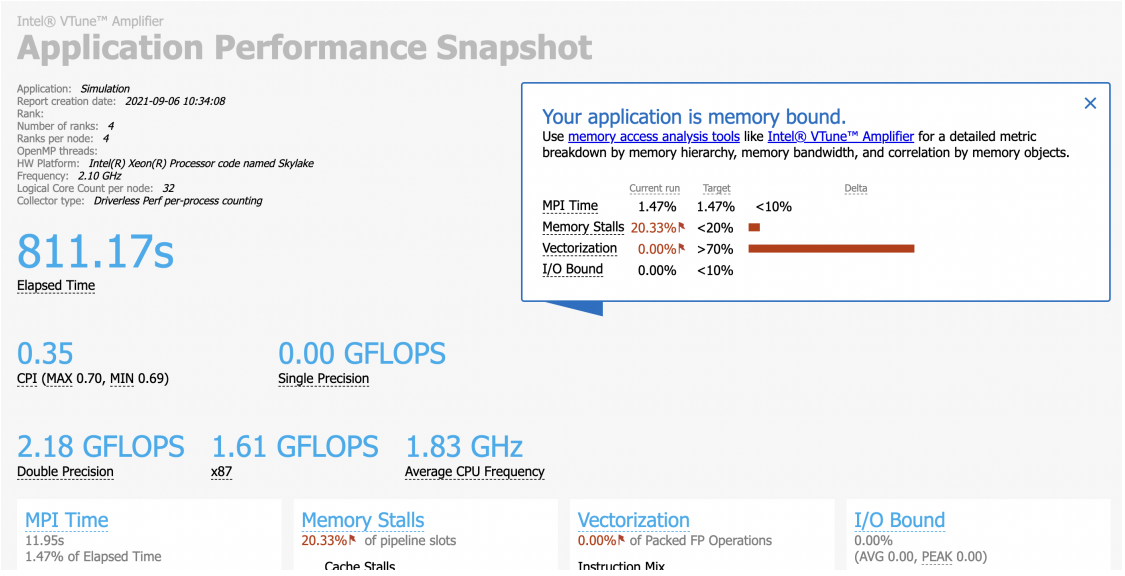


Figure 5.7: *APS* report for a memory-bound 2D simulation on four cores.

Only in such cases, where the overwhelming factor is not the communication, does the actual computational effort, e.g., whether one or two pulses are simulated, play a relevant role for the overall simulation time. On the other hand, in large 3D simulations the connection speed of the occupied nodes on the cluster forms the decisive factor, besides the optimizations on the software side detailed in Sections 5.4.3 and 5.4.4. For a non-admin user this is most of the time uncontrollable.

Note that the maximum clock frequency is in both cases of memory and communication-bound simulations by far not exhausted. This is the case for most HPC applications since data transfer, be it memory access or inter-node communication, usually forms the bottleneck. That fact has been discussed in Section 5.2.1 and explains why the clock frequency of cores in an HPC system are generally not that high.

The performance snapshots in Figures 5.6 and 5.7 were performed on an older code implementation, where still every floating point arithmetic instruction is scalar. This deficiency is taken care of in Section 5.4.1.

Compute-intensive functions

An *Intel® VTune™ Profiler* hotspot analysis identifies the time evolution operation of the electromagnetic waves to be most compute-intensive, along with the derivative as well as rotation operations, see also Chapter 2 with Section 2.4. These use the most CPU time besides some *SUNDIALS* functions, see Listing 5.1.

```
Function                                CPU Time  Module
--------------------------------------  --------  ----------------------------
Vaxpy_Parallel                          703.880s  libsundials_nvecparallel.so.3
NLin4u6_2D_Propagation                  323.384s  Simulation
LatticePatch::derive                    304.750s  Simulation
LatticePatch::derotate                  176.332s  Simulation
N_VWrmsNorm_Parallel                    174.280s  libsundials_nvecparallel.so.3
LatticePatch::rotateToY                 166.996s  Simulation
s13b                                    143.318s  Simulation
s13b                                    135.283s  Simulation
s13f                                    128.579s  Simulation
s13f                                     99.892s  Simulation
s13f                                     96.580s  Simulation
VSum_Parallel                            92.757s  libsundials_nvecparallel.so.3
s13f                                     89.295s  Simulation
VDiff_Parallel                           88.141s  libsundials_nvecparallel.so.3
VCopy_Parallel                           82.853s  libsundials_nvecparallel.so.3
VScaleBy_Parallel                        77.380s  libsundials_nvecparallel.so.3
VLin2_Parallel                           67.520s  libsundials_nvecparallel.so.3
LatticePatch::rotateToX                  66.771s  Simulation
N_VInv_Parallel                          34.423s  libsundials_nvecparallel.so.3
N_VMin_Parallel                          28.003s  libsundials_nvecparallel.so.3
N_VAbs_Parallel                          21.610s  libsundials_nvecparallel.so.3
PMPI_Allreduce                           21.296s  libmpi.so.12
```

Listing 5.1: Excerpt of an *Intel® VTune* hotspots analysis for a 2D simulation. Only the most time consuming functions are included down to the first of the *MPI* library. Clearly, this is not a communication-bound simulation, where the *MPI* functions are be among the top ranking.

Most functions from the `Simulation` module can be understood by their name. `s13f` and `s13b` are the backward- and forward-biased stencils of order thirteen that are applied on forward- and backward-propagating modes.

Notably, in many of the expensive functions there are loops where compilers assume vector dependencies in their reports. An investigation with the *Intel® Advisor* proves that there are no dependencies present in some of them. Vectorization can there safely be enforced with the help of *OpenMP* compiler directives that are made use of in Section 5.4.1.

5.3.3 Communication load

Tracing

Profiling as well as *tracing* are possible with the *Scalasca* toolset. While a profile gathers the essential information for most serial applications, parallel codes primarily require relative timing information. An event trace is a chronologically ordered sequence of event records. Information about entering and leaving code regions and sending/receiving data, are recorded and saved in a full trace [197].

Using tracing, information about the individual contexts of events and their dynamics is kept, since temporal and spatial relationships are preserved in the traces. In general, the profile data can be reconstructed from the traces. The problem is that a full trace analysis consumes a large amount of storage and incurs large overheads but is most of the time not necessary. The measurement overhead for trace collection is eased by filtering only relevant parts.

Scoring can identify where the overhead is likely to occur and optimizes profiling by weighting these regions higher. *Scalasca* does an automatic trace analysis and thereby distills the measurement to the essential events, first modeling and scoring the traces, then returning a high-level summary of relevant traces. The *Score-P* instrumenter automatically inserts measurement code to capture only the events of interest. *Scalasca* acts as a convenience function package over those provided by *Score-P* alone and creates a summary within a comprehensive and augmented report [197].

Environment variables are used to configure the measurement. Typically, compute routines, memory allocation, communication and input/output are instrumented. *MPI* and *OpenMP* routines are thereby always taken into account but frequently visited computation functions, which are not relevant for the analysis of threading/communication, get a lower score.

Further filtering seems not be necessary in the present case and manual instrumentation is possible. The code instrumentation causes an intrusion overhead but yields more detailed information than a periodic interruption of the code execution would in order to take measurements. The latter approach is called *sampling*.

Scalasca analysis

Scalasca offers an automated diagnosis of many parallelization peculiarities, including a search for patterns of inefficient behavior and their identification, classification and quantification. It thereby guarantees to cover the entire relevant event trace. It can be seen from Listing 5.2 that the buffer sizes for the event traces are quite small.

```
1  Estimated aggregate size of event trace:                   94MB
2  Estimated requirements for largest trace buffer (max_buf): 374kB
3  Estimated memory requirements (SCOREP_TOTAL_MEMORY):       4097kB
4  (hint: When tracing set SCOREP_TOTAL_MEMORY=4097kB to avoid intermediate flushes
5   or reduce requirements using USR regions filters.)
6
7  type max_buf[B]    visits  time[s] time[%]   time/visit[us]  region
8  ALL    381,966 1,545,984 17454.96   100.0         11290.52  ALL
9  MPI    381,966 1,545,984 17454.96   100.0         11290.52  MPI
10
11 MPI    193,536   516,096  6159.55    35.3         11934.88  MPI_Sendrecv
12 MPI     68,544   258,048  4902.22    28.1         18997.31  MPI_Barrier
```

```
MPI     67,184   252,928  6232.40    35.7         24640.99  MPI_Allreduce
MPI     26,338   259,328     0.29     0.0             1.11  MPI_Comm_rank
MPI     26,312   259,072     1.48     0.0             5.69  MPI_Comm_size
MPI         26       256     2.52     0.0          9824.59  MPI_Finalize
MPI         26       256   156.52     0.9        611404.14  MPI_Init
```

Listing 5.2: Profile of the *Scalasca* analysis of an old code version. The communication is performed with a combined sending/receiving routine. A lot of time is spent at the barrier.

`MPI_Allreduce` is used a lot within *CVODE* algorithms, see Section 6.3.3. Otherwise, much time in this call is a hint to load imbalance [198], since it performs re-synchronizations of processes. The barrier, which is completely unnecessary, aggregates too many resources. This is taken care of in Section 5.4.3.

An automated postprocessing of the analysis infers important additional metrics such as the time consumed by computation, *MPI* and *OpenMP* calls, critical path, imbalance, performance impact, etc. The idea of *Scalasca* is to identify *MPI* wait states (idle time) and the critical execution path. Inefficient behaviors are then classified and quantified in order to help identify the root causes. The corresponding *Cube* visualization is threefold. It shows

- the metric to investigate;
- the location in the code; and
- the distribution over processes/threads.

A colorization scheme lets problem areas be identified and located, see Figure 5.8 for a 3D simulation with that code version. With the focus on *MPI*, the excerpt shows a good load balancing of the communication. For a more thorough load imbalance analysis, profiles of varying problem sizes can be compared function by function in order to detect scalability problems.

5.3.4 Memory efficiency

An *Intel*® *Amplifier* Application Performance Snapshot (APS) report for a small simulation can be used to get an overview of the efficiency of the different memory access levels, see Listing 5.3 for a perfomance snapshot of a memory-bound 2D simulation.

```
| Your application is memory bound.
| Use memory access analysis tools like Intel(R) VTune(TM) Profiler for a  detailed
    metric breakdown by memory hierarchy, memory bandwidth, and correlation by memory
    objects.
|
  Elapsed Time:                            2771.30 s
  SP GFLOPS:                                  0.00
  DP GFLOPS:                                  2.57
  Average CPU Frequency:                      7.37 GHz
  CPI Rate:                                   1.53
| The CPI value may be too high.
| This could be caused by such issues as memory stalls, instruction starvation,
| branch misprediction, or long latency instructions.
| Use Intel(R) VTune(TM) Profiler General Exploration analysis to specify
| particular reasons of high CPI.
  MPI Time:                                  45.47 s            3.28% of Elapsed Time
    MPI Imbalance:                            N/A*
```

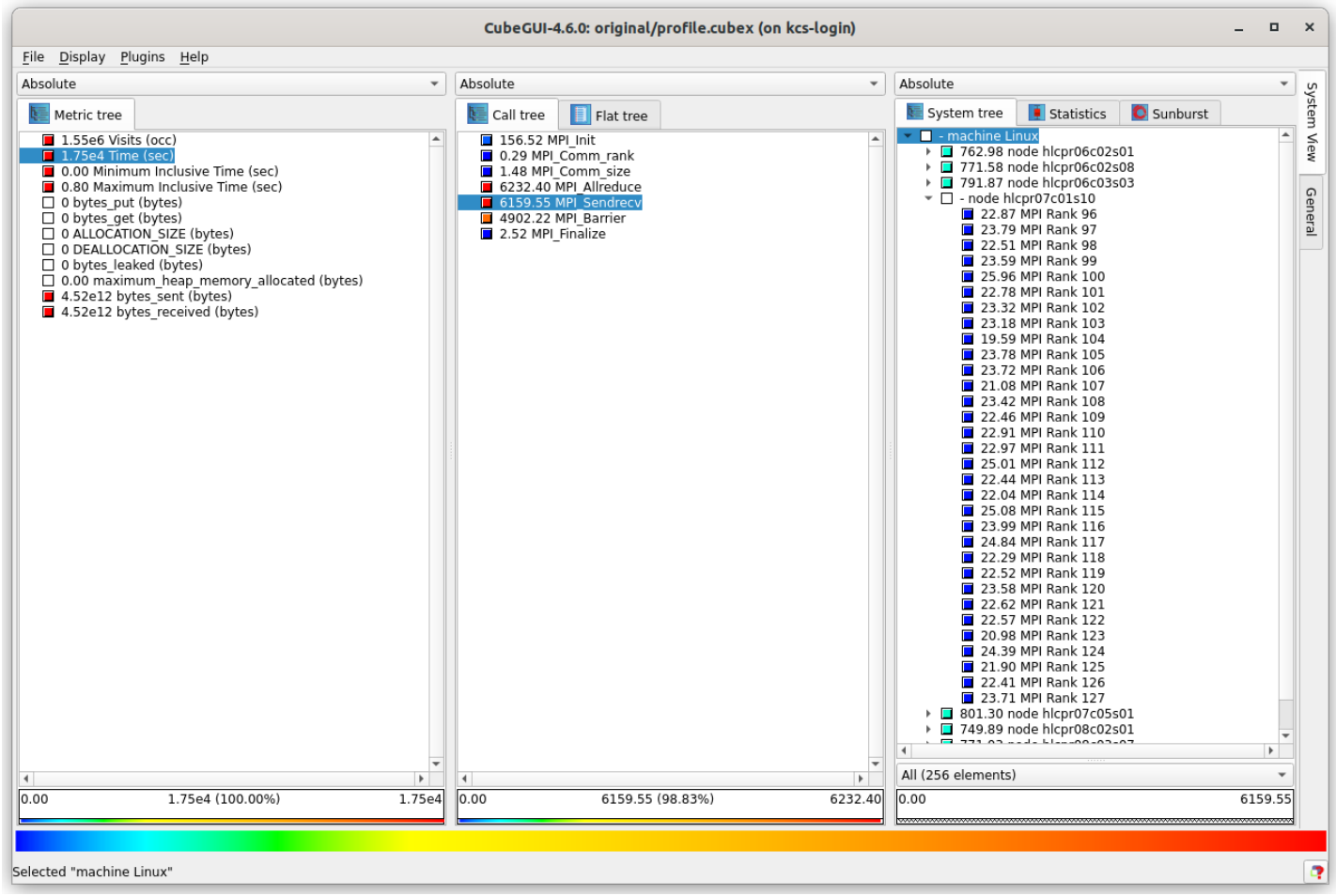


Figure 5.8: *Cube* visualization of a *Scalasca*-generated profile. Deep insights can be gained with a number of available metrics (*left pane*) the instrumented function (*center pane*) and individual listings for all cores processing the various instrumented functions (*right pane*). Here, the time spent by the processes in the communication routines is shown. It can be seen that there is a good load balancing in the communication. This is common for pure *MPI* applications.

```
| * No information about MPI Imbalance time is available. Set APS_IMBALANCE_TYPE
| to 1 or 2 to collect it.
    Top 5 MPI functions (avg time):
        MPI_Allreduce:                        30.13 s             2.17% of Elapsed Time
        MPI_Sendrecv:                         14.74 s             1.06% of Elapsed Time
        MPI_Init:                              0.57 s             0.04% of Elapsed Time
        MPI_Finalize:                          0.02 s             0.00% of Elapsed Time
        MPI_Comm_size:                         0.01 s             0.00% of Elapsed Time
  Memory Stalls:                             113.40% of Pipeline Slots
| The metric value can indicate that a significant fraction of execution
| pipeline slots could be stalled due to demand memory load and stores. See the
| second level metrics to define if the application is cache- or DRAM-bound and
| the NUMA efficiency. Use Intel(R) VTune(TM) Profiler Memory Access analysis to
| review a detailed metric breakdown by memory hierarchy, memory bandwidth
| information, and correlation by memory objects.
    Cache Stalls:                            51.70% of Cycles
| A significant proportion of cycles are spent on data fetches from cache. Use
| Intel(R) VTune(TM) Profiler Memory Access analysis to see if accesses to L2 or
| L3 cache are problematic and consider applying the same performance tuning as
| you would for a cache-missing workload. This may include reducing the data
| working set size, improving data access locality, blocking or partitioning the
| working set to fit in the lower cache levels, or exploiting hardware
| prefetchers.
    DRAM Stalls:                             34.60% of Cycles
| The metric value indicates that a significant fraction of cycles could be
| stalled on the main memory (DRAM) because of demand loads or stores. Use
| Intel(R) VTune(TM) Profiler Memory Access Analysis to get more details if the
| code is latency- or bandwidth-bound and what can be done to increase memory
| access efficiency.
```

Listing 5.3: Excerpt of an *Intel® Amplifier* Application Performance Snapshot (APS) for a rather small 2D simulation. The simulation is identified to be memory-bound. The various levels of memory access are shown. The simulation was running on only two cores, so there is no remote memory access.

Memory access patterns

Some inefficient memory access patterns are identified with the help of the *Intel® Advisor*, especially within some functions pointed out by the *Intel® VTune™ Profiler* to be particularly time-consuming in Listing 5.1.

The rotation functions mix up components of a six-dimensional vector. Memory access instructions with mixed-, unit-, and mostly constant-stride in both read and write operations are inherent there and unavoidable, but at least no irregular access is detected. An *Intel® Advisor* memory access pattern analysis points this out.

SIMD vectorization instructions for these functions ease that pressure and the problem becomes less severe when higher instruction set architectures for vectorization are used. See Appendix A for the highest available one on the *KCS* system. On the other hand, with more aggressive vectorization a vector register spilling shows up in the *Intel® Advisor* report along with the high vector register pressure in the propagation loops. This can sometimes be improved by splitting large loops into smaller ones, which can be done in several more loops to ease memory bandwidth pressure. Vectorization is discussed in Section 5.4.1.

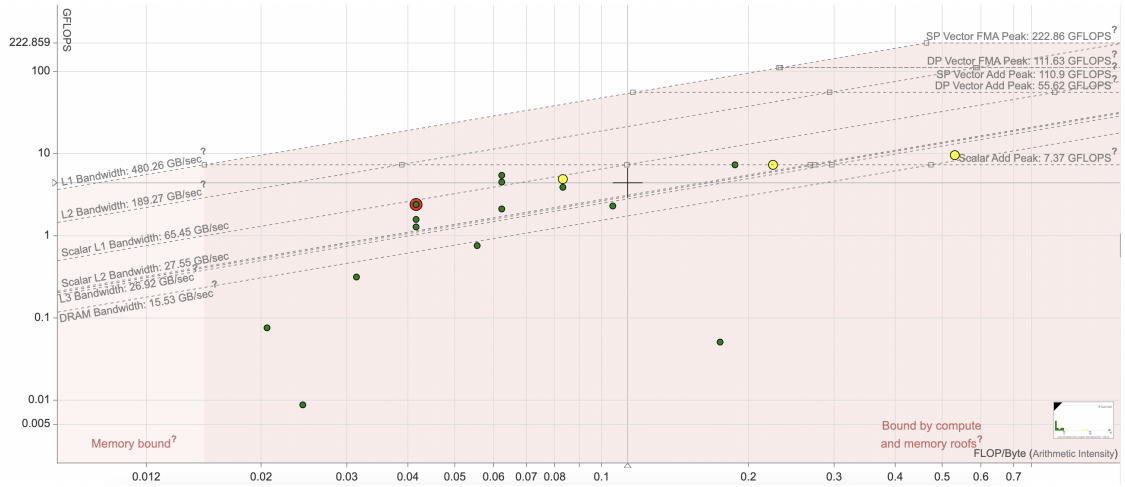


Figure 5.9: An *Intel® Advisor* roofline analysis of a 1D simulation on one core to investigate the performance and bottlenecks of program parts. Dots denote functions/loops and indicate the performance bounds for each loop, whether it is the memory-/cache-level bandwidth or the CPU performance. The more to the right and to the top, the better. The abscissa is the arithmetic intensity; the ordinate, the performance.

Roofline model

A roofline model enables to investigate which bandwidth type limits the memory performance of the specific function/loop. A complete roofline analysis with the *Intel® Advisor* capturing a large number of metrics and details about the software-hardware interaction is extremely resource-intensive and thus only conducted for 1D simulations. The roofline model also helps to investigate whether the CPU is reaching its theoretical peak performance and if there is a bottleneck in memory access or processing. A screenshot is provided in Figure 5.9.

Moving a function to the right side of a roofline model plot is often hard to achieve. This requires a reorganization of the computation style but different data layouts can help as well. Moving it upwards can be achieved by improved data locality in the memory-bound region and by more vectorization in the compute-bound region [190].

The remaining memory issues in the rotation functions are not disastrous, because the cache misses are only fractional compared to the number of memory loads and stores. Moreover, the maximum memory address range turns out to be 47 kB and is therefore in an acceptable range, yet beyond the capacity of the L1 cache with 32 kB on the *KCS*, see Appendix A. Some CPUs nowadays come with L1 cache sizes of 64 kB.

Attempts to achieve a reduction of cache pressure using cache-blocking, i.e., splitting the computations in such a way that fitting subsets of the data are processed sequentially, and unrolling loops with the help of pragmas have not been successful.

5.4 Parallelization

5.4.1 Vectorization

SIMD vectorization tests are performed with the *Intel® Advisor*. It mainly works as a vectorization checking tool but is not designed for HPC applications. In-depth analyses incur large overheads

and 3D simulations would require too large amounts of extra memory and are clearly unsuited for such an analysis. In view of these memory issues and because the focus is not on communication, vectorization analyses are done solely at the hand of 1D and sometimes 2D simulations.

Auto-vectorization

The least complicated way to increase the vectorization share is via compiler options such that the compiler automatically applies auto-vectorization instructions tailored to the CPU. Compiler optimization reports from some of the employed compilers (*GCC, Clang, Intel®*) are valuable tools to understand the compiler's judgments and behavior. There are many, often subtle, differences between compilers and compiler versions. Some compilers are more conservative than others with respect to auto-vectorization. Runtime differences due to more or less vectorization are observable. Compilers get more sophisticated and efficient with time. It is also remarkable that there is an increase of auto-vectorization with more threading of outer loops, see Section 5.4.4.

Compilers need to be cautions when in doubt in order not to falsify the computation. As mentioned in Section 5.3.2, the compilers sometimes assume vector dependencies where an *Intel® Advisor* dependency analysis proved there are none. This often happens when pointers are used. Different pointers might point to the same data and thereby a SIMD (single instruction multiple data) operation could possibly process overlapping data. This is called pointer aliasing. In the present case, there exist a variety of pointers, some actually pointing to the same data or subsets, see also Section 2.4. If safety is proven, vectorization can be forced with an *OpenMP* SIMD compiler directive in front of these loops or via a compiler option as shown in Appendix A.

SIMD declarations

The advisor yields hints on further functions and loops that could possibly profit from vectorization. Many more loops are in fact vectorized with SIMD declarations. These directives can be tailored specifically to the code. Safety of the changes is guaranteed with crosschecks. It has to be paid attention on so-called write-after-write dependencies and write-after-read anti-dependencies, where concurrency of instructions can cause trouble [199]. These are identified by the advisor and vectorization problems can often be solved by specifying a maximum number of iterations that can be performed concurrently.

For a 1D simulation the time spent in vectorized loops is now up to over 90% where it was zero before, see Figure 5.10.

The code consists of the user code and the external *MPI* and *SUNDIALS* libraries. The amount of user code varies, taking up to about 50% of the runtime, depending on many factors like the number of dimensions and the number of processes. Similarly, while the time spent for communication grows for larger simulations, the total share of vectorized code decreases and therewith its benefits.

The most important, most time consuming loops, which have already been discussed in the context of profiling, are now vectorized. This includes the time evolution and the derivative operations, as can be seen in Figure 5.10. Note that for this 1D simulation the order of most time consuming loops changes in relation to a 2D test, c.f. Listing 5.1, because a 1D simulation is dominated by more compute-intensive functions like those from *SUNDIALS*.

More vectorization draws the bandwidth and latency issues to the attention, as the processors digest larger amounts of data in a given time. This causes register spillings in the time evolution

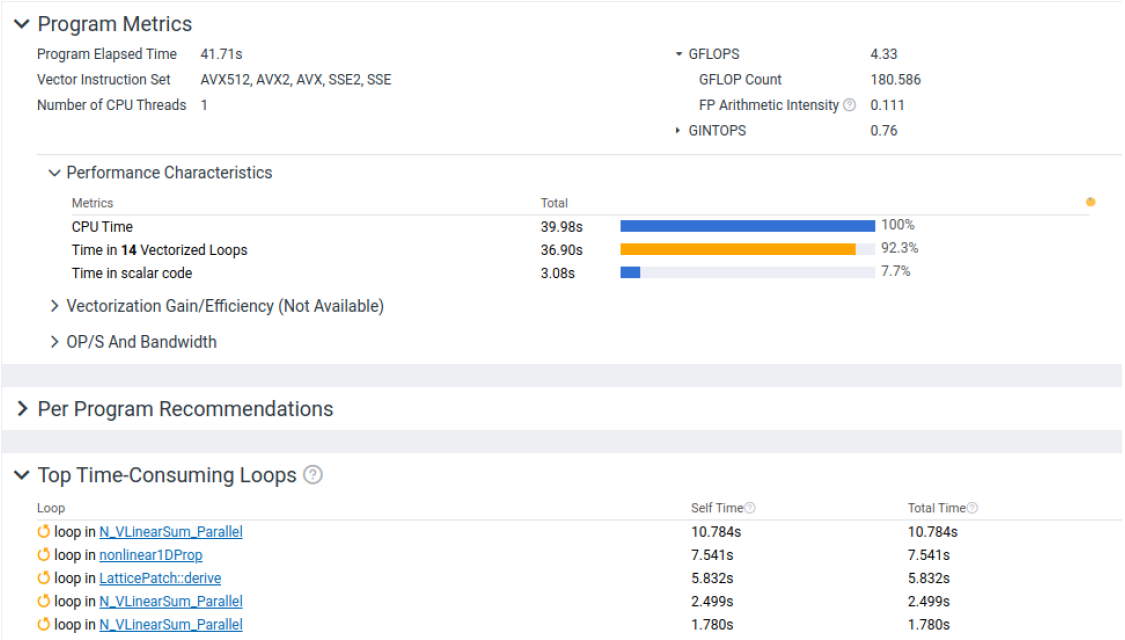


Figure 5.10: An *Intel® Advisor* analysis of a (nearly) fully vectorized 1D simulation.

function.

Benchmarks

The performance gain from vectorization cannot be stated in a single number, as it very much depends on the kind of simulation. Vectorization is rather a small-scale, core-level tuning. Keeping in mind the architecture of HPC systems, it can be concluded that a modern laptop is able to perform way better on small applications, where a large-scale distributed system cannot unfold its strengths.

Tests on the personal device in 1D and 2D showed large benefits from incremental vectorization. On an *M1 MacBook Pro 2021* 2D simulations such as to produce the results at the end of Chapter 4 on four processes sped up from 45 min with the old code version to only 12 min. The simulations for harmonic generation in 1D, c.f. Table 4.4, are now conducted within only a few seconds, while it was approximately half a minute before.

On *KCS*, a simulation in 2D on four cores utilizing a rather new code version built with and without vectorization took 776 s and 920 s, respectively. A short 3D simulation with a small enough grid to fit on one node ran on one core to avoid communication and provoke heavy computing. With and without vectorization the simulation took 420 s and 584 s.

A notable effect here is that the *Intel®* compiler performs way better on the *KCS* and simultaneously vectorizes more aggressively. This is attributable to the native support of *Intel®* compilers for their processors, incorporating more features.

With the worked out vectorization optimizations now in place, simulations in 2D can easily be performed on a notebook instead of having to submit a batch job to an HPC system already at that stage. To achieve that, also the *MPI* optimizations detailed in Section 5.4.3 play a role, while the influence of the latter really only unfolds when a larger number of processors is involved.

Optimized building

The *CVODE* solver from the *SUNDIALS* family of solvers has been used in various versions and build types. The computations rely heavily on the *SUNDIALS* library, where only a small fraction of functions are auto-vectorized by the compiler per default.

To enable higher compiler optimization levels also for the *SUNDIALS* library, it has to be built in "Release" mode. The optimized build type has a significant impact on the performance of compute-intensive simulations, since a large number of the computations is performed within *CVODE* functions. The library code is thereby vectorized to a large extent.

Employing multi-file or whole-program interprocedural optimization (IPO), the *Intel*® compiler analyzes all code files together and may apply various tweaks in order to optimize the overall program. IPO yields a performance gain and combats many problems identified by the *Intel*® *Advisor*. Additionally, it enforces more vectorization.

5.4.2 Multithreading

MPI was originally developed for distributed single core processors with message passing only being necessary for communication between different memory regions. With modern multi-core CPUs *MPI* is also used on shared memory resources, e.g., a node or socket, c.f. Figure 5.1. Reducing the number of *MPI* processes is the simplest way to reduce the communication overhead. In order not to lose parallelization speedup, multithreading can be implemented on top.

The industry standard to achieve this is the *OpenMP* API [173]. Threading with *OpenMP* works by insertion of compiler directives into existing code, as in the case with vectorization using *OpenMP*, discussed in Section 5.4.1. While *MPI* is an API for distributed memory computers, where each processing core refers to its own data, *OpenMP* is an API for programming shared memory parallel computers. The latter is based on the notion of threads and tasks, with the understanding that threads work together on a computation while sharing their memory, i.e., using the same address space. Tasks are computation snippets independent of each other.

An *MPI* process can spin up several threads that are processed by cores on the same socket or node. See for example Figure 5.11, where four processes on a socket span four threads each. The different ways to perform this kind of hybrid parallelization are outlined in Section 5.4.4. As mentioned in Section 5.2.2, cores can often be divided into hardware threads that are each capable of processing a thread. There is no notion of message passing in multithreading. "Communication" is achieved by enforcing the cache coherence of the cores discussed in Sections 5.2.1 and 5.2.3.

Threading pitfalls

Cache coherence implies some non-negligible, and often even costly, memory accesses throughout the memory hierarchy. Threads can contend for memory bandwidth and cache capacity, and even for functional units if SMT is used, c.f. Section 5.2.2. When a thread writes data, the copies of that data in the other threads' caches are invalidated to ensure that subsequent accesses use only the updated data. An issue to be kept in mind here is that the cache coherency mechanism operates on units of typically 64 or 128 bytes, the cache lines introduces in Section 5.2.1.

Differing threads accessing neighboring bytes on the same cache line cause severe memory access overheads, called cache line *false-sharing*. Concurrent access is prohibited because cache coherence is maintained on a cache line basis, not for individual elements. The mechanism thus

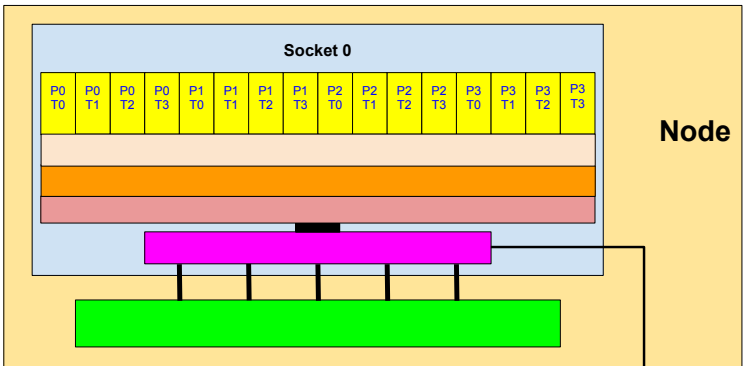


Figure 5.11: Hybrid multiprocessing and multithreading model with one process spanning four threads. One socket is shown, c.f. Figure 5.1. Here, each core (*yellow*) processes one thread.

causes the cache line to bounce between the caches of the processors and is therefore extremely time consuming. Threaded applications can suffer significantly from that [200]. A *race condition* occurs when multiple threads access the exact same data that is shared by the threads and at least one access is a write operation. The result of the computation then depends on which thread wins the race.

Sequential code regions in a threaded application can become extra time consuming. With threads running through the sequential region, long data transfer times are risked due to cache misses. It is thus often sensible to expand the parallel regions. Furthermore, some shared resources do not scale and a single core may have better memory access.

Schedules

`for` loops are the main source of parallelism with threads. Within a parallel region there is the notion of *shared* and *private* data. Shared data can be accessed by all threads, private data only by the owning thread. The execution of a loop by parallel threads is performed according to a *schedule* and a *chunksize*, the part of iteration space that each single thread operates on. Possible schedules are

- `static`: Equally large chunks are assigned to the threads in order. This causes the least overhead and best suited for balanced loops;
- `dynamic`: Iterations are split into chunks and assigned to threads on first-come first-served basis. This is useful for varying loads but compromises data locality;
- `guided`: Like `dynamic` but the chunksize decreases exponentially down to an indicated one. This is unsuited for loops where the first iterations are most expensive; and
- `auto`: The schedule is determined by the runtime. This has the lowest overhead and best load balancing and addresses loops that are often accessed.

To decide for a proper schedule a trade-off has to be found between thread idleness and scheduling overheads.

NUMA considerations

Synchronization overhead is most of the time caused implicitly in order to ensure that threads do not interfere with each other and process the up-to-date values [201]. There is an implicit synchronization barrier at the end of every parallel region, if not a contradicting clause is explicitly added to the directive. These so-called *flushes* guarantee a consistent view of memory for each thread.

Scaling issues with threads are present as a result of contention for bandwidth and cache capacity. It is thus mandatory to ensure data affinity, e.g., with the methods mentioned in Section 5.3.1 and bind logical threads to physical cores in order not to have them roam over the cluster.

There are undesired NUMA effects concerning memory allocation with *OpenMP*. The risk is that all the data are first written into memory by one *master thread*. This would cause all data to reside on that thread's local memory unit and can pose a severe bottleneck, as can be imagined considering diverse data paths from different memory units in Figure 5.1. Accordingly, a socket is a memory locality domain in the case of the *KCS* architecture. Performance-degrading NUMA effects can be combated by explicitly implementing parallel first-touch, mentioned in Section 5.2.3 and further discussed in Section 5.4.4.

Remarkable speedups can in principal be obtained through threading simulations, but in consequence of the problems discussed, most of the time the gain is small compared to multiprocessing. The pure *MPI* application is thus usually scaling better in a long range of core numbers [200]. Only when multiprocessing reaches its limits, multithreading as an additional layer of parallelism is useful to maintain scalability. This is the subject of Section 5.4.4.

5.4.3 Multiprocessing

As stated in Section 3.2 and detailed in Section 5.3.1, *MPI* is employed to perform the communication between processes. This is achieved by having single cores process patches of the lattice, thereby transferring boundary data to neighboring cores/patches, as shown in Figure 5.3. In short, *MPI* send and receive calls are posted in the code in such a way that this communication is correctly taking place. To combat parallel performance deterioration at larger scales, a revamped *MPI* communication style paves the way for the code to run distributed over hundreds of nodes. For purposeful simulations a sufficiently high grid resolution is required, c.f. relation (2.58).

Communication overhead

There are four main categories of parallelization overheads [198],

- lack of parallelism: The workload cannot be split into enough pieces;
- load imbalance: The pieces for each processor are not identical amounts of work;
- synchronization: Processors need to wait for each other; and
- communication: Inefficient patterns of communication.

Communication is overhead by construction and moreover forms the number one bottleneck for large simulations, c.f. Section 5.3.2 with Figure 5.6.

The three levels of communication in an HPC system, listed with increasing bandwidth, are

- between the nodes;
- between the sockets of the nodes; and
- between the cores of a socket.

Hence, in the first place, node-to-node communication ought to be reduced, especially when a large number of nodes is targeted. It has to be noted that generally latency forms the bottleneck, not the bandwidth. Bandwidth could easily be added by further interconnects. For the total transfer time it is obtained

$$\text{transfer time} = \text{synchronization time} + \text{latency} + \frac{\text{message length}}{\text{bandwidth}} \,. \tag{5.1}$$

It is generally faster to send fewer large messages than many small ones, since the latency of all individual sending operations accumulates. The synchronization time is also called idle time, the amount of time it takes for sender and receiver to perform a "handshake".

From the software perspective latency and bandwidth are dependent on the internal *MPI* protocol. The latter defines the mechanism for sending data to remote processes and contains an envelope (header) with metadata. For standard `MPI_Send` calls the protocol type is selected based on the message size and is implementation dependent, but can also be configured by the user. The three protocol types are

- the *short protocol*, which can send short messages within the metadata envelope. It has a short latency but small bandwidth;
- the *eager protocol*, which temporarily buffers the data. It has higher latency and bandwidth comparable to the short protocol; and
- the *rendezvous protocol*, which does not buffer. When the envelope is accepted at the receiver end, i.e., the receiving routine is called, a ready-to-send message is returned to the sender and delivery begins. Latency is worst in this case but the bandwidth highest.

Using the synchronous `MPI_Ssend` routine, the rendezvous synchronization can be enforced, `MPI_Bsend` on the other hand enforces buffering. The latter avoids deadlocks and may prevent idle time (synchronization) and returns directly, but the creation, filling and emptying of very large message buffers can be disadvantageous. Another, and most of the time the best option, are non-blocking routines. Such have been implemented in *HEWES* and are discussed below.

First steps

A simple but unwelcome way of speeding up the runtime is by reducing the precision of computations. Reducing, e.g., the order of the stencil matrices from thirteen to four, the width of the halos can be reduced from seven to three, see Chapter 2. This results in a tremendous reduction of the communication load and accordingly a significant speedup for distributed simulations. The runtime is reduced by more than 20% at a test on 256 cores.

It has to be kept in mind that a lower stencil order in turn causes the dispersion relation to deviate from the true vacuum speed of light at lower grid resolutions. Hence, the grid resolution has to be increased, requiring a larger computational effort in turn. There is thus a trade-off worth of further investigation. Such trade-offs are analyzed in Chapter 6 for the case of 1D simulations.

First thing to optimize is getting rid of the *MPI* barriers that can safely be removed but are often present in older *MPI* applications. The performance-degrading effect of the barriers is shown in the *Scalasca* analyses excerpts in Listing 5.2 and Figure 5.8. Removing the barrier is a simple but efficient optimization.

MPI virtual topology

OS noise is ubiquitous, caused throughout the time by other running processes. It can lead to synchronization issues in sensitive implementations. Non-uniformity in memory access and communication demands an optimization of the *MPI* topology. Adapting the virtual topology of the communicating processes to the simulated physical system is however beneficial in any case, since it enables the *MPI* implementation to perform optimizations under the hood.

Since the ghost cell exchange is a local operation, the communicating processes are best located nearby. The topology should generally be as cubic as possible in order to obtain a minimal "surface" for node-to-node communication. Accordingly, it is instructive to implement a Cartesian communicator. The new communicator is a member of the `Lattice` class, c.f Section 2.4, and is at startup configured at the hand of the user-defined lattice decomposition, c.f. Chapter 3.

Unfortunately, the current *MPI* implementations map multi-dimensional grids unevenly onto the nodes when using the convenience function `MPI_Dims_create` to create the process topology from the given number of processes and dimensions. More flexibility to optimize to the hardware will be provided with *MPI* 4.1 [200, 202].

One way to speed up code is to remove unnecessary tasks. The use of the virtual Cartesian topology dramatically reduced the number of instructions for ghost cell exchange. A virtual Cartesian topology assigns coordinates to the processes on the lattice. This allowed to eliminate hard-coded lookups for the correct neighboring processes. Associating IDs to the patches in the lattice became redundant, too. Large code fragments, including two whole classes thereby became superfluous, which enabled the shortening of call paths. The initialization of the communicator that was previously handled by a dedicated class is now performed by a member function of the lattice. The important functions taking care of the halo exchanges themselves could also be simplified and now work more efficiently. The initialization of the ghost cells, i.e., the communication buffers, is merged into the exchange function.

Nonblocking communication

The use of nonblocking *MPI* routines is in almost any case favorable [198]. Nonblocking means that the communication call returns immediately, i.e., it does not block further code execution until the operation is completed. This implies that other, independent operations can be performed by the processor while communication is ongoing or pending, as opposed to standard blocking routines. It is in this way basically possible to overlap communication and computation. Deadlocks and idle time can be prevented.

Just like the synchronous and buffered routines mentioned above start with `MPI_S` and `MPI_B`, the nonblocking routines start with `MPI_I`, where `I` is for "immediate". On the downside, nonblocking communication is also more error prone and fragile. The nonblocking operation might still be ongoing while the program continues and it has to be ensured that there is no interference. The message buffer must remain untouched until the message passing operation has completed, i.e., the buffer is safely transferred.

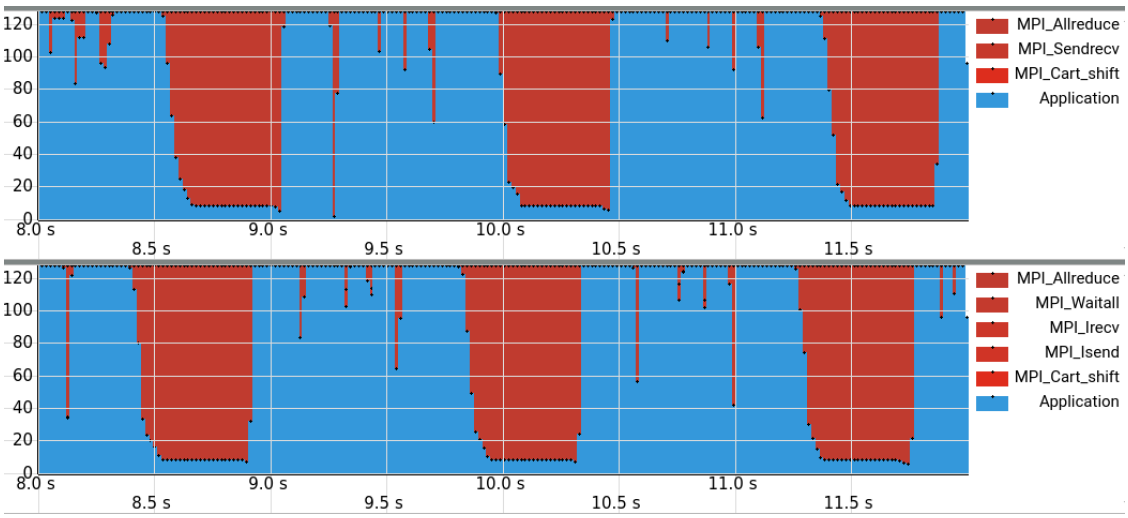


Figure 5.12: Excerpt of an *Intel® Trace Analyzer and Collector* analysis comparing blocking and nonblocking communication styles. For better visibility a short cutout of the runtime of a sample simulation in 3D is shown. The quantitative timelines show the overall load of *MPI* calls and the rest application accumulated over the 128 processes for the blocking communication (*top*) and nonblocking communication (*bottom*)

Note the difference between the return of a routine and the completion of a communication operation. Blocking routines only return when the data transfer is completed. *MPI* buffered send operations thereby return as soon as the send buffer is filled, independent of whether a matching receive call has been posted. Merely enough buffer space has to be provided. *MPI* synchronous sends only return when a matching receive call has been posted to pick up the data. As a consequence, synchronous sends might deadlock when the target is not getting ready to receive, e.g., when the latter is itself still trying to send. The standard `MPI_Send` operation is buffered for a sufficiently small amount of data to be transferred that fit into the system buffer. However, it changes to synchronous mode for larger amounts of data. Hence, the standard send call is at risk of deadlocking for enlarged data or even just another *MPI* implementation and should thus be used with care and foresight.

The blocking `MPI_Sendrecv` call, a convenience function combining send and receive calls under the hood, has been employed in an old code version. As of the *MPI* 4 standard [202] the nonblocking variant `MPI_Isendrecv` will be available and could bear benefits. Nonblocking routines return independently of whether the communication process has completed. Variants of `MPI_Wait` routines are required after nonblocking routines to wait for the final completion, after possible independent communications or computations could have been performed in the meantime. Idle time can occur at the waiting routine, of course.

In fact, an *Intel® Trace Analyzer and Collector* analysis shows that most of the *MPI* communication time is spent waiting, while the actual immediate sending and receiving operations are extremely fast. In contrast, the `MPI_Sendrecv` calls take more time in total. The nonblocking variant is quicker getting started. In both cases, a large amount of time is spent in the collective reduction operation. The latter is in order to re-synchronize the processes, c.f. Section 5.3.3, and is performed at all norm-like calculations of the *CVODE* solver, see Section 6.3.3. An excerpt of the analysis is shown in Figure 5.12.

It is in order to provide a benchmark for the overall performance at moderate scales in three

spatial dimensions. Full-time simulations in 3D took quite a long time before the optimizations. On the *KCS*, simulating colliding 3D Gaussian waves, such as in Figure 4.23, that propagate a distance of 40 µm on a lattice with $800 \times 800 \times 200$ points, divided into $8 \times 8 \times 2$ patches (128 cores), previously took 13 h 20 min. Now, the same simulation takes only 2 h 47 min.

Collective communication routines and cubic patches

There is an important distinction between point-to-point and collective communication routines that is worth considering, particularly when using virtual topologies. While the former describes data transfer between two specific processes, the latter is a predefined communication pattern between many processes. For the use of the latter, processes need to be ordered in some form, as is the case in virtual topologies.

Instead of a point-to-point halo transfer, requiring looping over the patches in every dimension of the lattice, one call to a neighborhood collective routine could suffice. Such routines exist in blocking and nonblocking versions. Using neighborhood collectives it is up to the *MPI* implementation to optimize the operation. Neighborhood collectives require a slicing of the lattice patchwork into patches of equal side length, i.e., into cubic patches in 3D, or squares in 2D. This is to ensure halo layers of equal size in any direction and has to be taken care of at the setup of the patchwork.

Results for the different ways of communication for a standard ghost cell exchange program have been tested [198]. Remarkable runtime and bandwidth differences between the standard `MPI_Sendrecv` calls and nonblocking routines could be observed. Using one neighborhood collective call has proven to be even way faster in that particular case.

Yet, the quantum vacuum algorithm as introduced in Chapter 2 at present does not allow this communication style. The algorithm processes the dimensions of the lattice sequentially in the order: I) ghost exchange, II) rotation to prepare for derivation, III) derivation, IV) derotation, V) time step update, c.f. Section 2.4. One collective neighborhood communication would distort the rotation and derivation operations.

Conceptually equal on a Cartesian grid and in compliance with the algorithm are nonblocking `Isend/Irecv/Waitall` combinations. `MPI_Cart_shift` can thereby be used on the virtual topology to directly infer the neighboring patches. It can prove beneficial to post the nonblocking `Irecv` calls before the `Isend` calls in order to evade late progress [198].

With regard to load balancing of the communication, it is in any case beneficial to use halos of equal size in every dimension. In Section 5.3.1 it was demonstrated how unequal slicing of the simulation box affects the scalability in an extreme case. The effect of using cubic instead of cuboidal patches in 3D simulations has proven to be tremendous. A short simulation on a lattice of the size $800 \times 400 \times 200$ with cubic slicing into $16 \times 8 \times 4$ processes took 104 s, while the unfortunate slicing into $8 \times 8 \times 8$ processes took 209 s. An analysis with the *Intel® Trace Analyzer and Collector* showed that in less extreme situations, non-cubic patches do not degrade the performance that badly. Nevertheless, there is an explicit advise given in the README file in Section 3 to use cubic patches. At present, even a runtime warning is issued if a 3D simulation is running with non-cubic patches.

Cubic patches can be further leveraged by initializing the ghost cells once at the generation of the patchwork, since their size stays the same for the transfer in any direction. This way, it is circumvented to create six ghost cells (two for every direction, as is effectively done for a collective routine) at every step, but the same two buffers can be reused without resizing for every direction.

Also for non-cubic patches the allocation of buffer space has been tailored to fit exactly the size of the exchanged data. Before that, there was an overhead generated on purpose, in order to create enough buffer space. The result of the modification is less memory consumption and communication overhead. Furthermore, the width of the ghost cells is configured to automatically adapt to the chosen order of the finite differences scheme according to the rule

$$\text{ghost layer width} = \text{stencil order} \div 2 + 1 \,, \tag{5.2}$$

where $\div$ denotes integer division, ensuring a reduced communication load for lower orders and a sufficient width for the calculation of the derivatives of Section 2.2.

5.4.4 Hybrid parallelization

With a large number of *MPI* processes per node, it becomes advantageous to additionally employ multithreading. The basic idea is to combine *OpenMP* threading on the node/socket level with processes spinning up the threads that communicate with the help of *MPI*. Such a configuration is shown in Figure 5.11 in Section 5.4.2. This can also be viewed as *MPI* on the outer level and an additional layer of parallelism for shared memory using *OpenMP* on the inner level [200].

Advantages

The advantage of a single address space accessible to parallel threads is that there is no need to buffer messages for communication. Additionally, with pure *MPI* it is not possible to further subdivide the distributed processes. A smaller number of *MPI* processes reduces the size of the internal *MPI* buffer space, the space for replicated data, as well as the communication overhead. The potential advantages are thus [198, 201]

- an additional level of parallelism, enabling increased scalability;
- less communication in virtue of simpler intra-node data transfer between threads;
- less memory requirements, since *MPI* processes often replicate data and the communication buffers consume a lot of space;
- no communication overhead in a strict sense, since only required data are accessed by the threads;
- better manageable load balancing, since in pure *MPI* this would require the transfer of tasks and data to underloaded processors, while with threading only tasks have to be moved; and
- the possibility of incremental parallelization, as opposed to pure *MPI*.

Adding another level of parallelization enables the application to scale up to more cores. The benefit of a hybrid parallelization, however, often becomes noticeable only at a very large number of cores. For extreme scaling, fewer numbers of processes per node presumably perform better because of the substantially increasing communication and memory costs. On the other hand, a hybrid parallelization does generally not improve performance in the region where the pure *MPI* application is still scaling well [201]. Figure 5.13 shows a sketch of typical performance curves related to the development efforts of the parallelization methods.

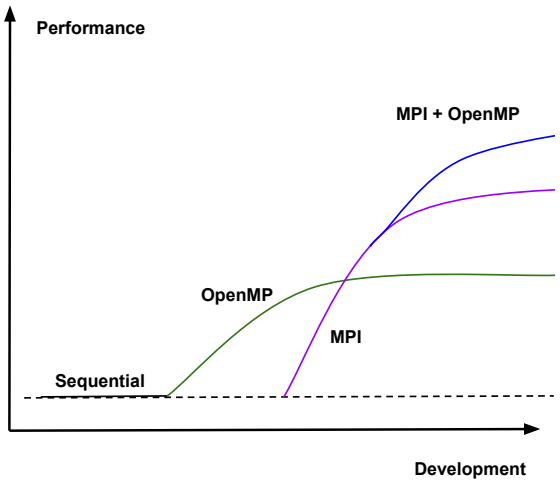


Figure 5.13: Sketch of typical performance curves for *OpenMP* and *MPI*. The use of *OpenMP* can very early lead to application speedups in multi-core environments. The implementation of *MPI*-capability takes some programming effort but pays out on distributed systems. When the performance of a pure *MPI* application no longer increases it can help to additionally make use of *OpenMP* threading. The figure is composed of ideas in [203] and [204].

The hybrid procedure is considered to serve as the workhorse on large systems. In this regime, it is quite common today. A hybrid parallelization is in fact even demanded from modern HPC code and rumored to be the key to exascale [200].

NUMA considerations and pitfalls

Nevertheless, the implementation of the hybrid parallelization is a very subtle procedure with a lot of pitfalls to avoid, e.g., race conditions (also on buffers) and non-deterministic bugs [198]. Additional overheads might be introduced through the sequential code parts, synchronization, false-sharing, and NUMA effects. At the same time, it has to be paid attention to other bandwidth bottlenecks created by shared caches. An oftentimes neglected factor is that as a result of more cache invalidations the sequential code parts take more time when operated on by multiple threads, which has been pointed out in Section 5.4.2.

It has to be stressed again that data affinity becomes of utmost importance, such that each thread can access its data as fast as possible. Guided first-touch programming is essential to evade NUMA problems when using *OpenMP* on top. At this point the discussion on the first-touch rule of Sections 5.2.3 and 5.4.2 is resumed.

Memory is structured in pages of typically 4096 kB that can be written according to first-touch by different processors to their local memory [187]. Hence, it has to be ensured that those processors first write the data that need it again afterwards. With *OpenMP* it is often the case that one master thread is responsible for writing, e.g., in a sequential region, c.f. Figure 5.14, with the consequence that other processors have non-optimal access to the data. As discussed in Section 5.2.3, the path lengths for data access in terms of bandwidth and latency can vary strongly on ccNUMA nodes. This is an unfortunate situation and can be prevented by threading the data writing, at best with `static` schedules, c.f. Section 5.4.2. It is further important to place threads onto the same NUMA domain as their parent processes.

With pure *MPI* every process automatically has its local copy of data. As repeatedly stated, this is problematic with regard to overall memory requirements. A severe memory overhead of pure *MPI* can be bypassed by sharing data between multiple threads within one process instead of having to allocate individual copies of data for each single core.

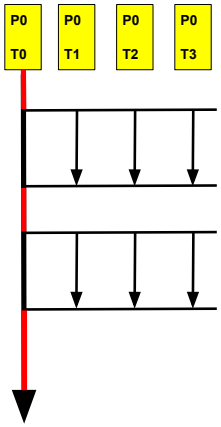


Figure 5.14: Idle threads in serial code regions. One process (P0) spans up four threads. The master thread (T0) works alone in the serial code regions (*red*) while the other cores are idle. Many processes can do this in parallel to form a hybrid parallelization model, c.f. Figure 5.11.

Additional overheads

OpenMP comes with its own issues that have been discussed in Section 5.4.2 and hence introduces additional overheads, e.g., with every implicit barrier at the end of parallel regions. In essence, parallel regions must be costly enough to justify multithreading, otherwise the implicit synchronizations alone slow down the execution.

Spinning up a parallel threaded region thus induces an overhead that has to be compensated by the gain, i.e., the region must be quite time consuming in sequential style. Multithreading on top of the multiprocessing will therefore unleash its potential commonly only at very large scales, where communication is not only the dominating bottleneck but also reaches its scalability limit. For small simulations the threading gain disappears into overheads.

To summarize, the performance gain may disappear into overheads formed by

- sequential code;
- idle (waiting) threads;
- synchronization; and
- hardware resource contention.

Possibly the greatest hurdle is that single-threaded regions in a hybrid model must be averted by all means in order not to waste resources. Parallel speedup is clearly limited by the serial code parts. In the context of strong and weak scaling this is manifested in the scaling laws conceived by Amdahl and Gustafson [205, 206]. Single-threaded regions put all other cores idle, see Figure 5.14. The effect will be a large number of spinning threads. In principle, in a hybrid model the entire code should be threaded, if possible.

It is thus a hassle to incrementally parallelize the existing code loop by loop. If this is not manageable and many instructions cannot be threaded, single-threaded regions enlarge the critical path and this approach might have to be discarded after all. Threading has to be at least implemented for selected, expensive routines along the main code path.

Hybrid styles

A neat way to circumvent NUMA effects and superfluous work is to place one *MPI* process on one NUMA locality domain. In the case of the *KCS* system this amounts to two MPI processes per

node, as there are two sockets with attached memory units per node, c.f. Figure 5.1. In order to exploit the full capacity of the machine, there should be at least one thread-spanning process per memory region.

Fewer processes imply more memory to be available per process, diminished intra-node messages, and fewer inter-node messages, since the latter get bundled. This is particularly beneficial for memory- and communication-intensive software. A *fully hybrid* parallelization means to put only one *MPI* process on each used node, the largest shared memory domain of the cluster. Having more MPI processes per node implies a *partly hybrid* parallelization as shown in Figure 5.11. There is no other way to find the ideal number of threads than to test it out with experiments for the application and system under prospect.

There are generally four styles of hybrid *MPI* + *OpenMP* programming [198], given by

- master-only: Communication takes place only in the sequential regions by the master thread. This is the simplest solution with a clear separation between the parallelization levels. Obviously, the disadvantages are idleness of all other threads and communication through the master thread caches only;
- funneled: All MPI communication takes place through the same thread and may also be inside parallel regions. Overlap of computation with communication is possible but communication still flows through the cache of the one thread;
- serialized: One thread at a time is allowed to make *MPI* calls. Sending and receiving threads have to be distinguished with *MPI* tags or different communicators. Overlap of computation with communication is possible, but additional latency overheads occur through the larger number of smaller messages being sent. With this style it can be arranged that threads communicate the data they own; and
- multiple: Several threads can do *MPI* communication simultaneously. This is clearly the hardest way, remarkably with no overall support (portability) and often bad performance, but often required to reduce overheads. The advantage is that threads naturally communicate their own data only. Also, inter-process and inter-thread communication can overlap.

Implementation of the hybrid parallelization

The relevant functions and loops that all need to be threaded are inferred from the *Intel® VTune™ Profiler* analyses, see Section 5.3.2. The most expensive routines in the time-evolution process are concerned with the computation of derivatives and the rotation operations. The flow of execution, including communication and communication preparation, according to the numerical scheme goes: I) fill the ghost cells, II) communicate (exchange) them, III) rotate, IV) derive, V) derotate, VI) update time, c.f. Section 2.4. With exception of the communication, all these are now optionally threaded. Hence, communication takes place only in a sequential region and the master-only style is chosen.

The patchwork is split into process domains and the communication calls are performed per process exchange instructions, not subdivided into threads as of yet. The latter would require quite some programming effort with no predictable outcome, as the last bullet point concerning hybrid programming styles above describes. Overlapping communication with computation is not an option since only when the exchange is complete the next operation on the data can begin. Trying to change this behavior would likewise require an expensive redesign.

Threading is implemented with explicit parallel loops, without resorting to more advanced constructs that would allow extended parallel regions, because synchronization after each step in the scheme is required anyways. With the exception of the filling of the ghost cells and their transfer, all of these are additionally vectorized, see Section 5.4.1. In the case of nested loops, the standard procedure is to thread the outer loop and apply vectorization at the innermost level.

Since the concerned loops are balanced, the `static` schedule is chosen. Notably, this schedule is best for parallel first touch [201]. In order to implement parallel first-touch along the discussion in Section 5.2, the initialization instructions of the electromagnetic waves and the helper vectors on the lattice mentioned in Section 2.4 are threaded with a `static` schedule. Since the communication is still serial in terms of threads, the receive buffers are written only by the master thread.

The remaining parts of the code are concerned with the creation of the setup configuration and not suited for parallelization. Both the use of the *OpenMP* library as well as *MPI* are optional in *HEWES*, while the use of *MPI* is strictly recommended, as detailed in the README file in Section 3.4.

Optimized *SUNDIALS* containers & benchmarks

The *SUNDIALS* library provides various `NVECTOR` implementations optimized for different data types and partitioning styles. These vectors form the containers for the electromagnetic field values used in the computations. The `NVECTOR_MPIPLUSX` module is available to support the *MPI*+X paradigm for hybrid parallelization with *MPI* and another local parallelization level like *OpenMP*. This is a rather new feature of *SUNDIALS*, see Chapter 6.

If a simulation is executed using processes and threads, this vector implementation is used globally on the lattice, wrapped around the local `NVECTOR_OPENMP` vectors on the single lattice patches. If only *MPI* or *OpenMP* are employed, the program resorts to the `NVECTOR_PARALLEL` or `NVECTOR_OPENMP` implementations. If neither multiprocessing nor multithreading are made use of, the fundamental `NVECTOR_SERIAL` is used.

The optimized containers give the hybrid *MPI+OpenMP* parallelization a tremendous performance boost compared to the old `NVECTOR_PARALLEL` module, designed solely for *MPI*. For a simulation on 512 cores, when running on 64 processes with eight threads each, the runtime is reduced by 50%. Globally, a performance gain with the hybrid model over the pure *MPI* model is achieved only for more than a thousand cores. As an example, the simulation running on 512 cores from the weak scaling test (Figure 5.5) can be equipped with additional four threads. The runtime thereby decreases from nearly 13×10^3 s to 8.7×10^3 s and the memory required per node from 31 GB to 8 GB. Of course, this is not directly comparable since the resources are extended, while the load is not.

It is noteworthy, however, that the same simulation on $512 \times 4 = 2048$ processes is problematic. Nominally, it occupies the same resources, but the processes do not operate on cubic patches in this case. Accordingly, eight threads would have to be used, which would exceed the allowed resources. Unfortunately, the tests cannot be carried much further because there is a limit of 121 nodes per job on *KCS*, see Appendix A. The memory requirements are at 11 GB per node and *CVODE* reported struggles in this specific test. There is a number of subtle convergence issues with the *CVODE* solver. A short list of experiences is given in Chapter 6.

In consequence, the threading can be useful to scale resources, while keeping patches cubic. As

in the case above, this has proven to be beneficial in some situations.

MPI-3 shared memory

The *MPI-3 shared memory* model deserves being mentioned in the context of hybrid parallelization. It works with shared memory windows replacing communicators, where every process of a node can directly read from and write to. It is based on the *MPI* single-sided remote memory access, where processes independently access memory windows with `MPI_Put` and `MPI_Get` calls.

In the case of *MPI-3 shared memory*, windows can be created on each shared memory island, e.g., a node or socket. Direct read and write operations are possible within an island, without *MPI* communication and without *OpenMP*-specific problems. Separate communicators for each shared memory island are created and joined to an overall communicator. Only inter-node/socket communication still relies on send and receive routines.

Utilizing *MPI-3 shared memory* may result in a powerful reduction of memory requirements, with data being stored only once per memory domain. The likely waste of resources of a hybrid model can be efficiently combated with this model, where *MPI* emulates the shared memory concept of *OpenMP*.

It might yield a significantly better bandwidth and the opportunity to reduce the halo buffers, since halo exchange is facilitated by direct writes or receives from other halos. The downside is that all the halo data have to be stored in the windows and the remote memory access calls need to be synchronized. Moreover, the communicator handling is intricate. The implementation of this concept might be an idea for future work.

5.4.5 Parallel I/O

Input/output (I/O) overheads can in some cases grow to dominate the overall runtime. With larger simulations and datasets becoming ever larger, efficient I/O evidently gains attention. Serialized I/O is no longer an option with potentially thousands of cores at work.

Reading from and writing to disk from a large number of parallel processes poses a challenge, since standard I/O routines are designed for serialized processing. Unix generally is not designed for parallel read/write operations. With newly put focus, nowadays I/O infrastructures are getting more complex with designs specifically for parallel I/O.

Such file systems can be configured for optimal performance, with different strategies for differing systems and I/O patterns, see Appendix A for the case of the *KCS*. Parallel file systems are constructed from many standard disks, where each single one is not particularly fast. Performance comes from operating on many disks at the same time. This is in full analogy to the computing power of HPC systems.

The standard parallel one-file-per-process approach generates an abundance of files on large scales, resulting in a saturation of the file system and metadata servers. These problems can occur when the maximum number of supported files is reached, by contention through simultaneous disk accesses, and because of exceedingly wasted space [207]. On a personal device the one-file-per-process approach might work well but it is not scalable on HPC systems.

Hence, efficient parallel I/O aims at a single file, which is operated on by multiple cores. In this way, the end result of parallel I/O is the same as for serial I/O and it is possible to reduce the metadata operations and the concurrent accesses to shared files. Postprocessing, too, is simpler on a single file.

As mentioned in the README file in Section 3.4, the output size of 3D simulations can become extremely large. The amount of output varies substantially, depending on the requirements. The user might want to construct a step-by-step dynamic visualization or be only interested in the final asymptotic state alone. Particularly a time-resolution of processes in 3D can demand very large disk space at the order of several terabytes.

The output format deserves consideration, too. While for small to medium size outputs a *CSV* format comes in handy, keeping the *CSV* format at large scales would be impractical. To achieve highly efficient parallel I/O, the file output operation is optionally parallelized with *MPI-IO* in binary data representation.

MPI-IO

MPI-IO is part of the *MPI* standard and thus highly portable. It operates very low-level and is the fundamental building block of other common parallel I/O libraries like *pHDF5* [207]. In simple terms, in order to coordinate the writing/reading position for each process, explicit file offsets can be defined. A process-specific "view" assigns the individual process a portion of the file.

A decision has to be made on the data representation used within *MPI-IO*. The options are

- "native": raw bytes as they are present in memory. There is no loss of precision and no conversion overhead. Yet, postprocessing for conversion is required;
- "internal": implementation dependent and not necessarily compatible with other implementations. Conversions are performed if necessary, which incurs overhead; and
- "external32": the standardized data representation big-endian IEEE. The best portable choice, which is always readable by *MPI-IO* on any platform and also other programs. Precision and performance may be lost in consequence of conversions.

Most *MPI* Implementations require the native format and extra conversion afterwards. The system architecture could thereby use little- or big-endian byte order. Hence, "native" is the format of choice in the present project.

The output size can then be exactly computed to be the number of lattice points times six (the number of field components) times eight bytes (double precision floating point numbers). This can be much less than in *CSV* format, particularly for large 3D simulations, where the data sizes increase drastically.

Parallel file systems are optimized for bandwidth, not latency. I/O performance suffers considerably when many small I/O requests are issued. Utilizing collective I/O is more efficient than independent I/O, since therewith a smaller number of large write/read operations is invoked. *MPI* thereby merges or funnels reads and writes and thus prevents file system contention. Hence, in *HEWES*, collective and nonblocking *MPI-IO* routines are utilized that are generally considered most efficient [207].

The speedup is substantial in comparison to the previous style of writing one *CSV* file per process. There is an HPC characterization tool that transparently captures I/O access pattern information, called *Darshan* [208]. With its help, the I/O operations of a rather small 3D simulation on 128 cores are compared in the two I/O styles. The grid for the corresponding simulation has a size of $800 \times 800 \times 200$ points and the data is written to disk at each of the 40 time steps. Results are shown in Figure 5.15. Note that in this case the one-file-per-process variant already produces $128 \times 40 = 5120$ output files.

Average I/O per process (POSIX and STDIO)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	0	0
Independent writes	0.000106171875	1.16452574729919e-05
Independent metadata	0	N/A
Shared reads	0	0
Shared writes	102.569920304688	1831.0546875
Shared metadata	26.306454921875	N/A

Average I/O per process (POSIX and STDIO)

	Cumulative time spent in I/O functions (seconds)	Amount of I/O (MB)
Independent reads	0	0
Independent writes	638.349089437499	4894.43458784372
Independent metadata	66.7813298515629	N/A
Shared reads	0	0
Shared writes	0.000106	1.15856528282166e-05
Shared metadata	0	N/A

Figure 5.15: Comparison of I/O variants by time consumption and output size. The quantities shown are averaged to per process, so the total amount is obtained through multiplying by 128. *Top: MPI-IO*. *Bottom:* one-file-per-process style in *CSV* format. In the former case of *MPI-IO* the independent writes are negligible and the bulk are shared write operations. The latter case of one-file-per-process forms the contrasting pattern. The overall amount of data is larger in the latter case, the time consumption is much smaller in the former case.

There is a rule of thumb: More than 10% of the runtime spent in I/O operations is considered an I/O bottleneck [190]. Measurements show that I/O does not form a bottleneck according to this rule, not even with one output per step in the one-file-per-process style. In 3D, where one would guess that I/O could become a bottleneck, computation and communication consume by far the largest amount of the total runtime. The share of I/O operations of the total runtime amount to about 5% with the one-file-per-process style and only about 1% using *MPI-IO*. This is visualized in Figure 5.16 with the help of *Darshan*.

Nevertheless, adequate and competitive I/O is an absolute requirement in view of ever larger simulations, as the discussion above makes clear. 5% of the total runtime can still become a significant amount of time, and large output volumes should better be stored as compact as possible.

Processing high-volume data

Compared to previous code versions, the disk output is reduced to only the electromagnetic field components. Earlier versions also saved lattice information that could be useful for postprocessing. To nevertheless ensure ease of use, *Python* modules to read out the field data in the correct order and shape for any dimensionality and format are provided in the code repository, as described in the README file in Section 3.4.

Simulations in three spatial dimensions plus time are even so likely to produce several terabytes of output data. It is then instructive to perform some form of downsampling and preprocessing or even the postprocessing directly on the cluster computer. Visualization of high-volume data can be carried out with the help of *MPI*-capable tools like *ParaView* [177].

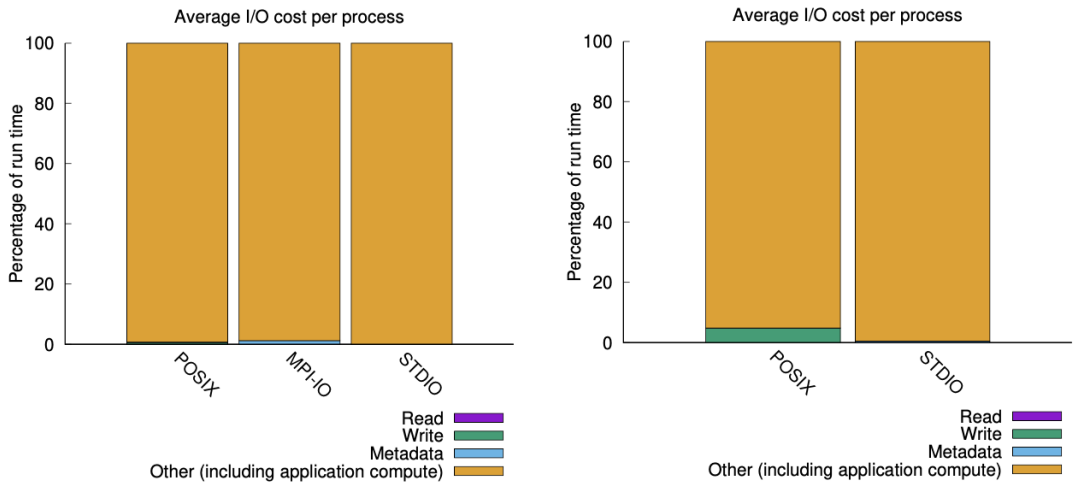
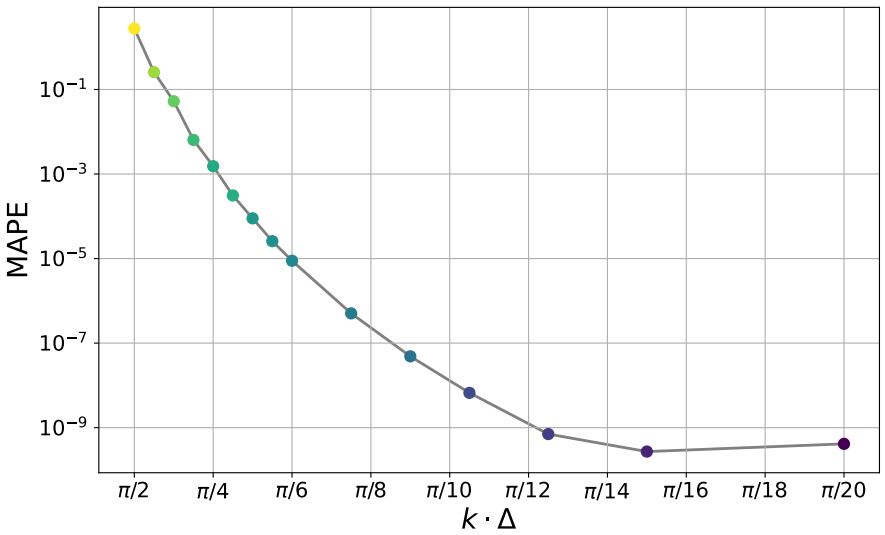


Figure 5.16: Comparison of I/O variants by their share of the overall runtime. The simulation configuration is the same as for Figure 5.15, however these shares seem to be quite universal. Different tests resulted in about the same outcome. STDIO marks the shell output, discussed in the README file in Section 3.4. *Left:* using *MPI-IO* that consumes about 1% of the runtime. *Right:* using the one-file-per-process style in *CSV* format that requires about 5% of the runtime.

Chapter 6

Accuracy Considerations



Errors at varying grid resolutions

"In order to solve this differential equation you look at it until a solution occurs to you."
– George Polya

6.1 Introduction

Differential equations are pervasive in physics and everywhere in science. Most of the time, they appear in a form of partial differential equations (PDEs), i.e., equations containing derivatives with respect to more than one independent variable. The solution of ordinary differential equations (ODEs), which contain only derivatives with respect to one variable, is a much better understood subject.

In Maxwell's theory the superposition principle for the set of differential equations holds, i.e., the equations are linear. As explained in Chapter 1, the Heisenberg–Euler theory of the quantum vacuum supplements Maxwell's theory of electromagnetism with nonlinear light–light interactions that are forbidden at tree-level in quantum electrodynamics. Taking those into account in the low-energy limit below the Compton scale of the electron, nonlinear terms in the electromagnetic field extend Maxwell's theory. This results in nonlinear differential equations of motion, invalidating the classical superposition principle in the vacuum.

The extended Maxwell equations as given in Equation (1.25) are thus classified as a system of nonlinear partial differential equations (PDEs). A merged PDE is obtained in matrix form in Equation (2.4). As outlined in Chapter 2, semi-discretization of the PDEs with the help of the finite differences scheme for the spatial derivatives results in the ODE system of Equation (2.22).

Solutions of differential equations can rarely be derived analytically in a closed form. Most of the time numerical approximations are required. The simplest way is not to reinvent the wheel but to rely on centrally managed and optimized libraries for code routines. As mentioned in Chapter 3, the *CVODE* solver [167–169] is the numerical software employed for the time integration of the propagation equation (2.22). *CVODE* is capable of solving nonlinear differential equations.

Previous chapters left open questions with respect to sensible choices of parameters and the impact on accuracy and performance of the numerical scheme for spatial derivatives and the time integration by *CVODE*. With the finite differences scheme being thoroughly discussed in Chapter 2, this chapter mainly goes into detail about the time integration. Further, the general accuracy of *HEWES* is analyzed. To this end, various configurations are scrutinized, studying different settings and the impacts on accuracy and performance.

Outline

This chapter first provides a rough overview of ordinary differential equations and a selection of widely used solution methods in Section 6.2. In Section 6.3, the *CVODE* solver, which is employed for the time integration of the equations of motion, is introduced. An accuracy analysis with varying configurations is conducted in Section 6.4, including trade-off considerations, and closed by a note on convergence issues.

6.2 Numerical solution of ODEs

Classification

An initial value problem (IVP) for a time-dependent function $y(t)$ with initial value y_0 at time t_0 can be formulated as

$$\dot{y} = f(t, y) \,, \quad y(t_0) = y_0 \,, \tag{6.1}$$

with

$$\dot{y} = \frac{dy}{dt} \,, \tag{6.2}$$

i.e., the time-derivative of $y(t)$ generally depends on a function of $y(t)$ itself.

Equation (6.1) is a first-order differential equation, since it contains only derivatives up to order one. Note that if higher-order derivatives were contained the equation could be rendered a system of first-order differential equations [209]. If f contains no explicit dependence on t, i.e., $f(t,y) = f(y(t))$, the differential equation is homogeneous. If f contains nonlinear terms in y, the differential equation is nonlinear.

Roughly speaking, the IVP (6.1) has a unique solution (in some interval of t and y) if f is continuous in t and Lipschitz-continuous in y,

$$|f(t,y_1) - f(t,y_2)| \leq L\,|y_1 - y_2| \,, \tag{6.3}$$

with a Lipschitz constant $L \geq 0$ [210].

6.2.1 Explicit and implicit methods

The explicit Euler method is the ancestor of all numerical methods to approximate solutions of differential equations. Truncating the Taylor expansion of $y(t_0 + h)$ at the first order in the step size h, one obtains

$$y(t_0 + h) \approx y_0 + h\,\dot{y} = y_0 + h\,f(t,y) = y_1 \,. \tag{6.4}$$

The true solution after one time step of size h, $y(t_0 + h)$, is approximated by y_1. An approximation y_n for $y(t + nh)$ can then be obtained in an iterative manner. The step size $h_n = t_n - t_{n-1}$ can principally be varied from step to step.

In other terms, approximating the derivative by the first-order forward finite difference approximation, c.f. Section 2.2.2,

$$\dot{y} = f(t,y) \approx \frac{y(t+h) - y(t)}{h} \,, \tag{6.5}$$

it is obtained that

$$y(t+h) \approx y(t) + h\,f(t,y) \,. \tag{6.6}$$

An explicit algorithm infers the state of the next time step from the information given at the current state of the system. An implicit method on the other hand makes use of nonlinear approaches – coupled equations at each time step – to find a prediction for the iterate of the next step. The prescription for the implicit Euler method is thus

$$y_{n+1} = y_n + h\,f(t_{n+1}, y_{n+1}) \,, \tag{6.7}$$

whereby y_{n+1} is defined implicitly. This scheme is obtained using instead of the forward difference approximation, the backward difference approximation [209]. Iterative approaches are required to compute the updated approximations y_{n+1} with the help of the available values t_n, y_n, t_{n+1}. Commonly root-finding or fixed point iterations are employed. The initial guess for the iteration is a predicted value computed explicitly from the available history. In the case of the implicit Euler method, a first prediction is usually obtained with the help of explicit Euler method [209]. The explicit Euler method thereby functions as a predictor formula for the implicit Euler method.

While the explicit Euler method in many applications tends to introduce large errors if the

step size is not sufficiently small, the implicit approach is more stable. The principally higher cost induced by the additional nonlinear equation system can thus be compensated. Stability means that a small change in the initial value leads to a small change of the solution. In other words, the solution is well-behaved under a small perturbation of the system [209]. For sufficiently small step sizes this can always be achieved with Euler methods, but small step sizes can become very costly and thus render the technique less efficient than one that is stable for larger values of h. Hence, for stability quantification it is instructive to assume that h is not too small and define regions of stability [209, 210].

Where it is beneficial to employ the implicit Euler method, the system can be called *stiff*. Put differently, parameters in a system of differential equations that cause a variation of the solution by orders of magnitude make it stiff [169]. A rigorous mathematical definition for stiffness does not exist, but the general concept is important for stability considerations [210, 211]. In stiff differential equations f has a large Lipschitz constant and such systems more rapidly break out of a stability region. The Lipschitz constant plays an important role in the error bound estimations of the approximate solutions. The more stable a scheme, the less restrictive the conditions on the step size [209].

6.2.2 Higher-order methods

Stability is closely related to convergence. The latter means that for $h \to 0$, the error of the approximations also approaches zero. The local errors of the approximate solutions introduced at every time step accumulate to a total or global error in the final solution approximation. The global error of a method of order p is bound by h^p, i.e.,

$$|y(t_n) - y_n| \leq Ch^p \,, \tag{6.8}$$

with an error constant C. Comparing the Taylor expansions of the exact and the numerical solution, they should agree up to and including the h^p term, when a method of order p is used. A method of order p is said to be convergent with order p. Higher-order methods naturally converge faster [209].

The Euler methods are of order one. The so-called Runge–Kutta methods are among the most widely used higher-order methods and rely on integrating (6.1),

$$\int_{t_n}^{t_{n+1}} \dot{y}(t)\, dt = \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt \,, \tag{6.9}$$

to obtain as approximation

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt \tag{6.10}$$

and approximate the integral with the help of quadrature formulas. The higher orders are in this case not attained through the evaluation of higher-order derivatives but via more intermediate evaluations of f [210]. There are also implicit Runge–Kutta methods.

SUNDIALS includes the *ARKODE* package that provides explicit, implicit, and explicit-implicit Runge–Kutta methods for stiff, nonstiff and mixed stiff/nonstiff problems [168]. Runge–Kutta methods belong to the class of so-called one-step methods because the new solution is obtained solely form that of the previous step. Other important types of higher-order are Adams and BDF (Backward Differentiation Formula) multi-step methods.

6.2.3 Multi-step methods

Using approximations of previous values for f, the integral in (6.10) can also be approximated with the help of interpolations. The explicit Adams method (Adams–Bashforth formula [212]) of order p uses an interpolation polynomial of degree p through t_{n-i} and f_{n-i} for $i = 0,...,p$. This polynomial is then integrated. Higher-order Adams methods are obtained using higher-degree interpolation polynomials. Taking into account available solution approximations of previous steps renders Adams a multi-step method. Multi-step methods are superior to Runge–Kutta methods particularly when high accuracy is essential [209].

Especially at higher orders it is favorable to employ the implicit Adams–Moulton formula [170]. The latter commonly corrects the approximation obtained with the explicit Adams–Bashforth formula, taking it as starting iterate to be inserted into the Adams–Moulton formula. This is a famous example of a predictor-corrector method [211]. Fixed-point iterations are most prominently utilized to solve the nonlinear equations [210].

The implicit p-step Adams–Moulton method has a larger interval of absolute stability and a smaller error constant than the explicit p-step Adams–Bashforth method [210]. Yet, while Adams methods have great success for nonstiff systems, they often perform poorly on stiff problems.

Backward differentiation methods, like the backward Euler method, are better suited for the latter case [210]. These make use of numerical differentiation rather than integration. Instead of integrating the interpolating polynomial in the integral expression (6.10), an interpolating polynomial of the differential equation (6.1) itself is differentiated backwards [211]. At order p, p previous solutions that have been approximated are used for an interpolation polynomial $u(t)$ passing through the preceding solutions and the upcoming step approximation, while requiring

$$\dot{u}(t_{n+1}) = f(t_{n+1}, u(t_{n+1})) \,. \tag{6.11}$$

BDF is accordingly an implicit method and thus also executes a predictor-corrector scheme. Instead of fixed-point iteration, Newton's method for root-finding is commonly used [210].

Both Adams methods (with constant step size) and BDF are linear multi-step methods, using a linear combination of previous time steps in the inference process and in the implicit forms an additional corrector included for a future step. The general formula for linear multi-step methods is given in Section 6.3.2.

As mentioned above, higher order integrators can be more efficient. As opposed to one-step methods, high orders do unfortunately not ensure convergence of multi-step methods for $h \to 0$. Further conditions have to be satisfied [210, 211]. Adams and BDF multi-step methods are those with the most widespread use for many applications.

SUNDIALS includes the *CVODE* package targeting IVPs in the form of (6.1), with implementations of Adams–Moulton and BDF methods for nonstiff and stiff problems, respectively [168]. The highest available orders of the methods in *CVODE* are twelve for Adams–Moulton and five for BDF. The latter suffers from instabilities above order six [211]. *CVODE* is detailed in Section 6.3.

The nonlinear algebraic equations appearing at implicit methods, which can be viewed as generic root-finding or fixed-point problems, can be solved with Newton methods or fixed-point iterations. *SUNDIALS* includes the *KINSOL* package for nonlinear algebraic systems that provides modified and inexact Newton methods for the root-finding case, and fixed-point and Picard iterations for fixed-point systems [167, 168].

6.3 The *CVODE* solver

6.3.1 Structure

The time integration of the propagation equation (2.22) is performed by *CVODE* (C-language Variable-coefficients ODE solver), a state-of-the art open source ODE solver from the *SUNDIALS* (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers) family of solvers which are explicitly designed to solve nonlinear equations on large-scale, HPC systems [167–169, 213, 214]. The *SUNDIALS* packages have a long history of success and are widely used in the scientific community.

SUNDIALS solvers are written in a data-independent manner. *SUNDIALS* operates on vector structures of different types, called `NVectors`, with specific type-dependent functionality. Mathematical operations on the `NVectors` are generic, agnostic to the particular data structure, parallelization method, and solver type [168].

A few implementations of `NVectors` have been discussed in the context of hybrid parallelization in Section 5.4.4. Notably, all parallelism within *CVODE* is encapsulated in vector and solver modules and user-supplied functions. For *MPI*-based parallel vectors each dot product or norm-like operation requires one `MPI_Allreduce` call [168].

Since the first versions of the Heisenberg–Euler simulation codes were written, the development of *SUNDIALS* has made significant progress. To keep up with the growing complexity of computer architecture, new programming models, and the popularity of external libraries, a redesign of the *SUNDIALS* packages has been undertaken over the recent years.

To meet the above points, efforts have been made to increase the flexibility of the solver classes and the `NVECTOR` implementations. The new, highly customizable solver classes are equipped with many optional settings. Interfaces ensure seamless interoperability with other popular scientific libraries. Furthermore, the object-oriented design principles ensure extensibility for the user, easing the addition of new features and code reuse [168].

Of utmost importance for the present work are the enhancements regarding performance at extreme scales. As outlined in Section 5.4.4, the hybrid *MPI* + *OpenMP* programming model is taken into account by *SUNDIALS* with new many-vector implementations. These form a software layer ensuring well-defined partitioning of data between computing resources and computation operations on that data. The data can be partitioned among distinct vector instances placed on separate hardware resources, whereon the actual computations take place.

Devoted to the hybrid style of *MPI* communication over a cluster in conjunction with some further form of parallelism on the individual nodes is the *MPI+X* `NVECTOR` implementation. It can be comprised of various subvectors for different kinds of parallelism on the nodes, e.g., multithreading with *OpenMP*. A dedicated *OpenMP* `NVECTOR` is available as one subvector type. The performance boost described in Section 5.4.4 is impressive. The general many-vector approach can also be used, e.g., for GPU offloading.

Other big changes to the interface include the new SUNContext object with the purpose to enable mechanisms for logging, profiling, and monitoring that are described in Section 6.3.4 and employed for analyses in Section 6.4. Evidently, later versions of *SUNDIALS* incorporated many bug-fixes and provide more overall stability. The code has been refactored to work with the latest versions as of writing.

6.3.2 Methods

CVODE targets stiff and nonstiff IVPs for ODEs in the explicit form of (6.1) using variable order and variable step size implicit linear multi-step methods with formulas of the form [168, 169]

$$\sum_{i=0}^{K_1} \alpha_{n,i}\, y_{n-i} + h_n \sum_{i=0}^{K_2} \beta_{n,i}\, f(t_{n-i}, y_{n-i}) = 0 \; . \tag{6.12}$$

As mentioned above, implementations of Backward Differentiation Formulas and Adams–Moulton formulas of orders p up to 5 and 12, respectively, are available. The methods are characterized in Equation (6.12) by values of $K_1 = 1$, $K_2 = p - 1$ and $K_1 = p$, $K_2 = 0$, respectively. Orders and steps size are automatically adapted according to the local error tests described in Section 6.3.3. Additionally to step size adjustments for error control, the order of the method is dynamically changed with the purpose of maximizing the step size. For details of the implementation, see [169].

As described in Section 6.2.3, nonlinear solver modules are available within *SUNDIALS* to solve the nonlinear systems that are formulated by the implicit methods [167, 168]. An implementation of a Newton iteration for root-finding is used by default. The Newton iteration requires the solution of linear systems and the type of the Newton iteration depends on the extra choice of linear solver. This results in either a modified or inexact Newton method for a matrix-based or matrix-free linear solver [169]. *CVODE* provides a bunch of different linear solvers and is highly versatile in the tuning of each.

The fixed-point iteration does not require the additional solution of a linear system. Anderson acceleration can be used to speed up convergence of the fixed-point iteration [215]. However, Anderson acceleration showed to have only detrimental effects in the present case. Testing it in various configurations resulted in a – sometimes severe – slowdown for 1D simulations.

In line with the discussion above, the Adams method (Adams–Moulton formula) with a fixed-point iteration is recommended for nonstiff ODEs in the *CVODE* user documentation [169]. For stiff ODEs mostly BDF in conjunction with Newton iteration would be used.

Roughly, the *CVODE* solver performs the following operations to integrate the equations of motion stepwise [169]: I) predict the solution, II) solve the nonlinear system and thereby correct the prediction, III) perform an error test, IV) choose the order and step size for the subsequent step. In case the error test fails or if there are issues with the convergence, the solver is capable to perform recovery attempts by adapting step sizes and method orders.

Choice of the *CVODE* method

In order to choose the suitable method within *CVODE*, the stiffness of the system has been investigated in [145]. The parameter under investigation is the lattice spacing, or resolution, Δ, defined as the number of points per physical length. Its importance is stressed in Section 2.3.

Adams–Moulton is found to be superior until the equations become, at a very high resolution of $\Delta = \lambda/500$ or $k \cdot \Delta = \pi/250$, very stiff. Only at that point BDF should be employed for enhanced stability and faster convergence. Such high grid resolutions are definitely not targeted, first because that significantly impairs the efficiency and second because the available high orders of the scheme make them avoidable. Moreover, when shrinking the lattice point distance the accuracy is at some point limited by a dominating round-off error.

To conclude, implicit Adams–Moulton is the method of choice. As explained above, the order and step size are adapted automatically. The maximum order is by default set to twelve. A fixed-point iteration is chosen for the nonlinear algebraic system.

6.3.3 Error controlling

CVODE uses a weighted root-mean-square (WRMS) norm to control error-like quantities. The used weights are based on the current solution and on the relative and absolute tolerances mentioned in Chapter 3 and supplied by the user [169],

$$W_i = \frac{1}{\mathrm{rtol} \cdot |y_i| + \mathrm{atol}_i} \,. \tag{6.13}$$

As mentioned above, every WRMS norm operation on parallel *MPI* vectors requires an `MPI_Allreduce` call. Error control is a critical feature of *CVODE*.

At every step the local error is estimated and required to satisfy the tolerance conditions. Relative and absolute errors are defined by

$$e_{\mathrm{rel}} = \frac{|y_n - y(t_n)|}{|y(t_n)|} \,, \quad e_{\mathrm{abs}} = |y_n - y(t_n)| \,. \tag{6.14}$$

The relative error tolerance should generally not be smaller than the unit round-off [169].[1]

Just as in Chapter 4, the quantification of errors with a relative error metric is limited to regions where the true solution is not too near to zero. The absolute error tolerance is thus important to control small values of the solution, where the relative error tolerance becomes inapplicable. The absolute error tolerance should therefore be set to some conservative threshold value, while the relative error tolerance controls the significant digits above that threshold.

The relative and absolute error tolerances for the time integration with the *CVODE* solver are set to 10^{-12} or lower for the simulations presented in Chapter 4. This yields time integration errors that are hardly detectable due to the errors introduced by the finite differences scheme. With sufficiently low tolerances, the time integration can be considered exact in light of the errors introduced by the spatial derivative approximations. This matter is discussed in Section 6.4.

At every step, the local error is estimated and required to satisfy the tolerance conditions. Whenever the error test fails, the step is redone with a reduced step size. New step sizes are computed based on the asymptotic behavior of the local error [169]. Tolerances hence guide the time step adaptations. Depending on the dynamics of the system, the solver performs larger time steps in quiet regions and shorter steps in highly dynamic regions. Clearly, the locally controlled errors accumulate with the number of integrator steps to a global error.

Additionally, the order of the method is adjusted with the goal of maximizing the step size, selecting the order for which a polynomial of that order best fits the data [169]. Finding a more efficient configuration, *CVODE* is therewith able to maintain the desired accuracy while taking larger steps. The sizes of the steps can be further controlled with the help of tuning parameters, but the default values in those configurations have a long history of success [169], see also [213, 214].

Besides the error of the time integration, which is restricted by *CVODE*, there is the spatial derivative error of the scheme.

[1] The unit round-off precision machine epsilon for the *ISO C double* type is $2.220\,446 \times 10^{-16}$.

6.3.4 *SUNDIALS* evaluation functionalities

SUNDIALS provides capabilities for profiling of code regions, monitoring of the solver states, and detailed logging. These modules constitute useful tools to inspect the solver load and performance, and are made use of in Section 6.4 to collect data on every test run. It has to be kept in mind that these mechanisms can act seriously performance-degrading, even more so the more detailed the evaluation.

Conveniently, there is a function to directly print a bunch of solver statistics, see Listing 6.1.

```
Current time                = 0.0001000034355709279
Steps                       = 26031
Error test fails            = 3
NLS step fails              = 0
Initial step size           = 4.115745890359425e-21
Last step size              = 3.845318480764656e-09
Current step size           = 3.845318480764656e-09
Last method order           = 7
Current method order        = 7
Stab. lim. order reductions = 0
RHS fn evals                = 44830
NLS iters                   = 44829
NLS fails                   = 0
NLS iters per step          = 1.722138988129538
LS setups                   = 0
Root fn evals               = 0
```

Listing 6.1: *CVODE* statistics. In particular the number of internal steps and iterations of the nonlinear solver (NLS) serve as quick reference to estimate the load and performance of *CVODE*. No linear solver (LS) is used for the employed fixed-point iteration.

More details can be extracted with the advanced functionality. Parameters of the solver can be accessed during runtime. This is most conveniently done with the help of the monitoring capability that extracts solver states in defined intervals of internal solver steps. For example, supplementary metrics like estimated errors and error weights are collected. To make this possible, additional `NVectors` need to be allocated. Hence, used in that way, monitoring also noticeably affects the memory resources.

Profiling lists the *SUNDIALS* code regions and their time consumption, alike the profiles shown in Section 5.3.2. Listing 6.2 lists the *CVODE* functions in a sample 1D simulation of Section 6.4. Instrumentation is possible and currently used around the integration loop, called *Propagation*, which obviously encompasses nearly the full time spent in the *CVODE* routines.

```
SUNDIALS PROFILER: SUNContext Default
Results:                              % time     max/rank    average/rank    count
================================================================================
From profiler epoch                   100.00%    29.072401s    21.804267s    2
Propagation                           99.94%     29.055443s    21.791331s    100
CVode                                 99.94%     29.055359s    21.791267s    100
N_VLinearSum                          88.18%     25.635841s    19.187485s    1803942
SUNNonlinSolSolve                     38.10%     11.075888s     8.293876s    26034
N_VScaleAddMulti                      21.52%      6.256312s     4.688735s    52062
N_VWrmsNorm                           20.00%      5.815803s     4.348774s    160492
```

```
11 N_VWSqrSumLocal                    17.97%      5.222930s     3.911580s    160492
```

Listing 6.2: Excerpt of *SUNDIALS* profiling information. The time consumption and the count of calls of the various functions is listed. Four processes (here called ranks) are used for the simulation. The propagation is split into 100 integrator steps. This kind of report can be used to quantify the varying loads of *CVODE*.

The logging capabilities provide a *SUNDIALS*-wide mechanism for logging of errors, warnings, and information. If everything runs fine, there are neither errors nor warnings. The informational output encompasses a summary of the nonlinear solver iterations. Further debug output to, e.g., examine the `NVectors` is possible.

6.4 Accuracy and performance evaluation

The stability of the numerical scheme with respect to a well-behaved propagation of electromagnetic modes of any frequency has been detailed in Section 2.3 and is further discussed in Section 7.2. In this section the accuracy and performance of both the numerical scheme of biased finite spatial differences and the time integration with *CVODE* are analyzed. Some trade-offs have been marked in the preceding chapters, which are brought up for discussion here. While utilizing the *CVODE* solver at high accuracy, it is imperative to guarantee a real vacuum-like dispersion relation with suitable parameters of the spatial grid.

Note that the focus is not on parameters within the ODE system, but in the first place on the numerical scheme and the configuration of the simulation space (lattice) as well as the *CVODE* solver settings. The investigation first expands on the analysis of the dispersion relation in Section 2.3. Likewise, one plane wave is propagated on a one-dimensional grid. The wave has a wavelength of 1 µm and propagates a fixed distance of 100 µm. This is a comparably long distance in relation to the simulations presented in Chapter 4. The basic settings are listed in Table 6.1.

Table 6.1: Parameters for accuracy and performance tests. A plane wave with a fixed wavelength and polarized in E_z-direction propagates in x-direction on a 1D lattice with fixed physical size. The wave further propagates for 100 periods, i.e., 100 µm. Recall that the lattice has periodic boundaries. Other parameters, the grid resolution, the stencil order, and the tolerances of the *CVODE* solver are varied.

Grid	Length	100 µm
	Lattice Points	400 – 4000
Probe	A	$0.1E_{\mathrm{cr}}$
	λ	1 µm
Stencils	Order	1–13
CVODE	Relative Tolerance	$10^{-16} - 10^{-6}$
	Absolute Tolerance	$10^{-16} - 10^{-6}$

The settings are used for testing the accuracy of the scheme and for trade-off estimations. Recall that there are no nonlinear interactions with a single plane wave. After having propagated for 100 full periods, the final wave position should be identical to the original position, such that the deviations are easily identifiable. In order to get an averaged, relative expression, the error

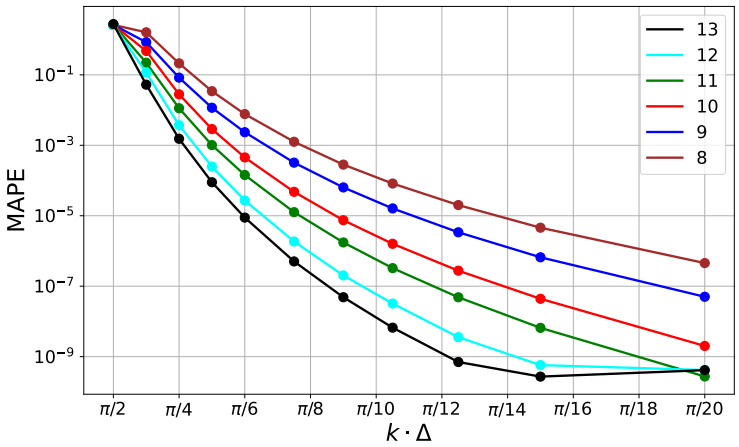


Figure 6.1: Errors at varying grid resolutions and stencil orders. A MAPE of about unity is not meaningful, hence an even coarser grid is not taken into account. Lower stencil orders need accordingly higher resolutions to converge to a minimal error. There is a pattern in the differences observable. The accuracy loss going from an even order of accuracy to the next lower uneven one is larger than from an uneven order to the next lower even order.

metric is chosen to be the mean absolute percentage error. The same metric has been chosen to assess simulations in Chapter 4. For completeness, it is defined as

$$\text{MAPE (in \%)} = \frac{100}{n} \sum_{i=1}^{n} \frac{y_{\text{pred,i}} - y_{\text{true,i}}}{y_{\text{true,i}}} \,. \tag{6.15}$$

The MAPE shown in the following diagrams is not given in percent. As opposed to the problems in Chapter 4, a critical region for the metric, where the *true* distribution lies near to zero, can be avoided. This is achieved by shifting the plane waves upwards with a just sufficient z-offset of 0.11.

6.4.1 Testing the scheme

For tests of the scheme only, the *CVODE* tolerances are kept very low at 10^{-16} each. First, varying grid resolutions are compared at varying order of the scheme. The results, visualized in Figure 6.1, can be compared to the analytical plots of the dispersion relations at all orders in Figures 2.6.

According to Equation 2.58, the lattice resolutions used for quantum vacuum simulations should adhere to the rule $\Delta \lesssim 1/12\,\lambda$ or $k \cdot \Delta \gtrsim \pi/6$ at order thirteen of the numerical scheme. As in the simulation of higher harmonics, c.f. Table 4.4, even after a propagation of 100 µm the accuracy is then still appropriate to extract the vacuum nonlinearities with high accuracy.

On the other hand, decreasing the resolution from there on, the critical regime is quickly reached and the modeling of waves deteriorates. At 400 lattice points, i.e., $\Delta = 1/4\,\lambda$ or $k \cdot \Delta = \pi/2$, the wave gets damped and shifted so heavily after 100 µm that a comparison to the initial state is nonsensical. This situation can be directly compared to Figure 2.5 (*left*). The order of accuracy of the scheme evidently plays an outstanding role.

The average times in the propagation loop for the simulations of Figure 6.1 are depicted in Figure 6.2. The times are obtained through *SUNDIALS* profiling information, c.f. Listing 6.2.

As mentioned in Section 3.2, the computation time is approximately independent of the stencil

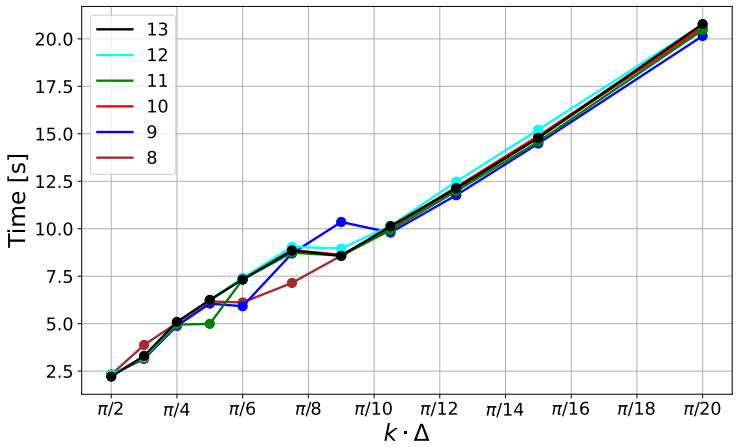


Figure 6.2: Time consumption at varying grid resolutions and stencil orders. This proves that the stencil order has a negligible impact on the runtime. Generally, the load scales linearly with the grid size.

order, but the correct physical solution is rather approached with a higher stencil order. This lucidly demonstrates the benefit of the high-order scheme, where a small lattice can be employed.

Only for distributed simulations the order of the scheme unfolds an impact on the performance through the communication load, c.f. Equation 5.2. This impact becomes dominant in 3D, as detailed in Section 5.3. Apart from that, even higher order stencils could be implemented. Obviously, the grid resolution influences the *CVODE* profile of Listing 6.2. This is because in the present case the resolution is increased with the overall grid size, i.e., the problem size.

6.4.2 Testing *CVODE*

Concerning the solver statistics of Listing 6.1, both the stencil order and the grid resolution seem to be rather irrelevant. On the other hand, the tolerances significantly impact the number of internal solver steps and iterations.

Tolerances

The time integration should be kept exact in order to have only the scheme limiting the accuracy. It is not a simple task to guess the right tolerances and it is clearly problem-dependent. The relative error tolerance controls the relative error and, keeping in mind its insufficiency at small absolute values, is complemented by the absolute tolerance. As described in Section 6.3.3, the latter is crucial to control small overall values that are naturally being dealt with in the investigation of quantum vacuum nonlinearities.

Errors are tested at varying tolerances with the baseline simulation given by the one with 1200 lattice points, i.e., $\Delta = 1/12\,\lambda$ or $k \cdot \Delta = \pi/6$, and stencil order twelve. With respect to the lattice resolution and propagation time, this is a suitable setting for the detection of quantum vacuum effects. The errors are summarized in Figure 6.3.

In this particular case, an absolute tolerance of 10^{-8} is sufficient. The relative tolerance then only needs to be adequately lower than the error introduced by the scheme, c.f. Figure 6.1. Finding

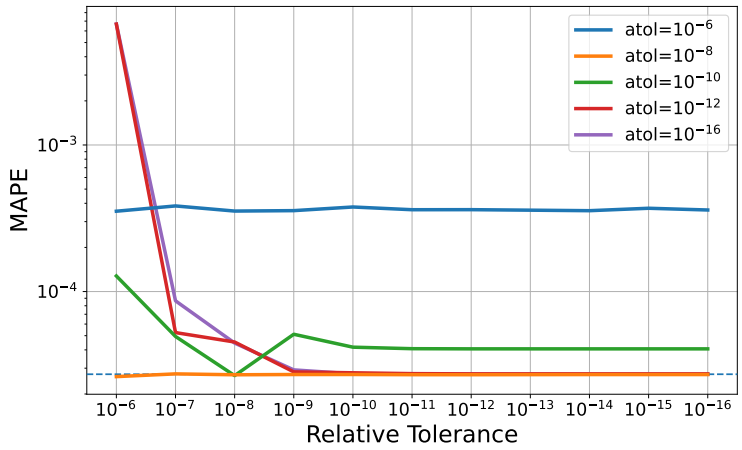


Figure 6.3: Integrator tolerances and accuracy. Tolerances are varied for the simulation with stencil order 12 and $k \cdot \Delta = \pi/6$. There is a long range in the conservative region, where the relative tolerances do not affect the overall accuracy of the integration in this case. The errors introduced by the dispersion on the grid predominate in that range, as one can see from Figure 6.1. The dashed horizontal line shows where the total errors are at the threshold given by the scheme and where the time integration is exact in that sense. Clearly, it is not sensible to set a strict absolute tolerance but at the same time a loose relative tolerance.

an apt pair of tolerances most of the time requires experiments for the particular class of simulation. Vacuum nonlinearities, which are tiny compared to the strong pulses, are not present in this test. More stringent error bounds are for this reason considered as required for the simulations of Chapter 4.

For example, the configuration for harmonic generation in 1D of Table 4.4 with the results in Figure 4.13 can be considered. For the initial wavelength of the probe pulse, the resolution is $k \cdot \Delta \lesssim \pi/12$. According to Figure 6.1 this would yield a MAPE of the scheme of less than 10^{-9}. Note that the mangitudes of the pulse amplitudes are smaller than in the tests above. Moreover, the propagation distance and hence also the propagation of the error is less. On the other hand, after about 40 µm, higher harmonics arise with accordingly smaller wavelengths, upon which the scheme introduces larger errors.

Considering all factors, it is challenging to make a decided prediction. In view of the above discussion, a relative tolerance of about three orders of magnitude smaller than the error of the scheme should be sufficient, considering error propagation. The absolute tolerance would be chosen small enough to take into account the nonlinearities. It is obtained $\text{atol} \simeq \text{rtol} \simeq 10^{-12}$. The actual tolerances to obtain the accurate results of Figure 4.13 are conservatively chosen to be $\text{atol} = \text{rtol} = 10^{-16}$, since the cost of such a simulation hardly matters.

Notwithstanding, the tolerances do matter in more expensive simulations. The simulations in 2D and 3D of Section 4.5 are indeed conducted with $\text{atol} = \text{rtol} = 10^{-12}$. Clearly, these simulations do not yet aim to be of ultra high precision and the highest harmonics quickly fade, such that they need not be propagated for a long distance. Nonetheless, the elemental results were unchanged even when testing one of the 2D simulations of Chapter 4 with doubled grid resolution in every dimension or by lowering the tolerances for the 3D simulation generating Figure 4.25 to 10^{-15}. In the latter case, the runtime nearly doubled.

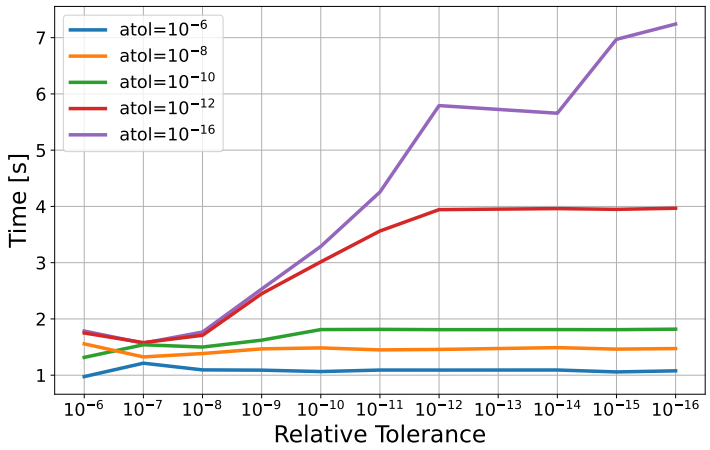


Figure 6.4: Time consumption at varying integrator tolerances. Too strict tolerances can incur very large overheads that need not necessarily be mandatory, as discussed in the text.

This demonstrates that it is worth trying out higher tolerances, keeping in mind the imperfection of the numerical scheme for a given stencil order and grid resolution. Figure 6.3 shows a long range up to very low relative tolerances. Relaxing the tolerances can reduce the time consumption significantly, at least in compute-intensive simulations, while still keeping time integration exact. Runtimes are shown for the simulations of Figure 6.3 in Figure 6.4.

Number of steps

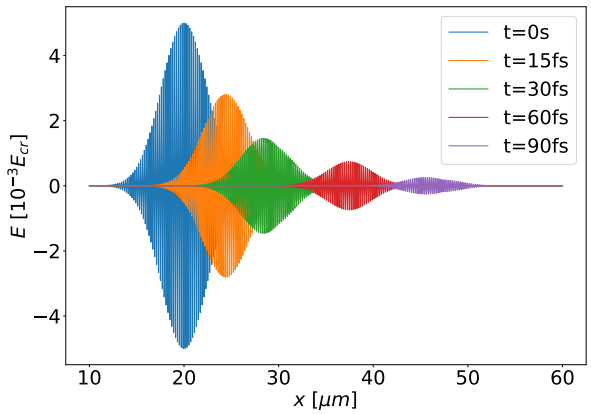
Another concern is the number of steps taken per propagation distance. *CVODE* takes a self-adjusted number of internal steps to reach each user-given step. In the README file in Section 3.4 the advise is given not to set too few steps in order to elude large interpolation errors. In the `CV_NORMAL` mode, in order to reach an external step exactly, *CVODE* performs internal steps until the external step is just reached or surpassed and then interpolates back [169]. Large steps can incur *CVODE* to complain about too much work in order to control the error. Good results are achieved with a step size of about 1 µm and there is no noticeable accuracy improvement in the case of the above simulations with shorter user-defined steps.

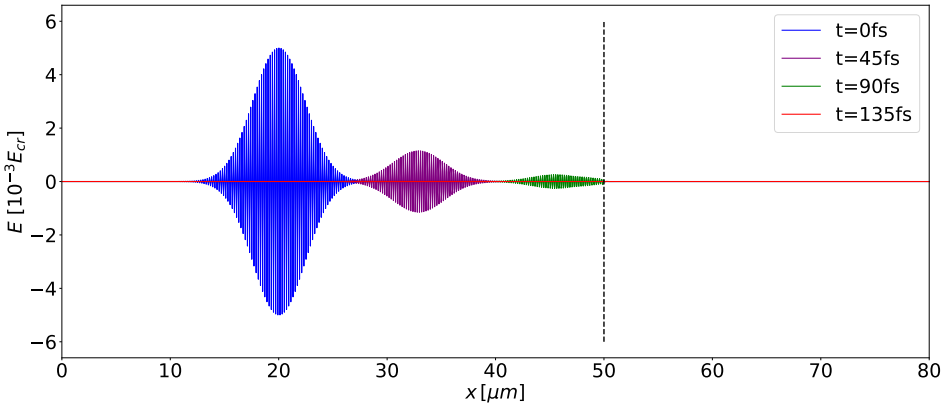
CVODE convergence issues

There are unstable scenarios where *CVODE* takes increasingly long to converge or fails completely when a too small step size is reached. In this sense, the solver unfortunately proves not to be stable with respect to the initial configuration. Moreover, this behavior seems to be even system dependent. For example, some later compiler versions are demonstrably responsible for causing convergence errors more often. The README file in Section 3.4 includes a note that for the initial wave parameters decimal numbers should be chosen. Using, e.g., a `sqrt` expression instead can already cause problems. Code instrumentation for different purposes can also cause convergence corrector errors. Issues of this kind have to be investigated in more detail in the future, e.g., with the help of the *CVODE* debugging capability.

Chapter 7

Outlook





Dispersion with and without a resolution barrier

"The only way you can predict the future is to build it." – Alan Kay

7.1 Introduction

Instead of discussing ongoing research employing the Heisenberg–Euler solver, which has been mentioned throughout the preceding chapters, this chapter gives an outlook on future descendant simulation codes. Ongoing investigations are outlined in preparation to conduct simulations with farther scope.

As discussed in Section 2.5, the code is subject to the curse of dimensionality. Practical feasibility limits are set upon the solver for high-frequency waves requiring high-resolution grids, particularly when it comes to full three spatial dimensions. The all-optical QED vacuum simulator outlined in Chapter 3 uses a numerical scheme of high order and consequently requires a comparatively small number of lattice points for the accurate propagation of an electromagnetic wave of a given wavelength, c.f. Chapter 2. The number of lattice points nevertheless needs to be further reduced without diminishing the accuracy in order to simulate effects like polarization rotation in three-dimensional setups.

To address this problem the optimizations outlined in Chapter 5 have been put into place. The scalability on distributed computing systems can be leveraged to conduct high-load simulations. Nonetheless, in order to reach the x-ray regime in 3D, a drastic reduction of the computational load in the first place has to be achieved. The hope is that this can be done with novel techniques including machine learning and multi-scale simulations.

Matter is totally absent in the description in the present work, c.f. Section 1.6. To take it into account, a conceptually very different formalism seems adequate. The Dirac–Heisenberg–Wigner formalism takes into account the full QED picture, including pair creation. There is progress on that numerical front, too.

Outline

A sophisticated future approach to render extremely expensive simulations feasible via a multi-scale ansatz is discussed in Section 7.2. Taking into account pair creation numerically might be possible with the Dirac–Heisenberg–Wigner formalism outlined in Section 7.3.

7.2 Dynamic multi-scale simulations

Harmonic generation has already been simulated in 3D, see Section 4.5, but polarization flipping, even when using extrapolation techniques, could incur a numerical load that is easily million-fold. To make this clear, see the lattice requirements for the simulations of birefringence and harmonic generation listed in Chapter 4.

Specifically birefringence scenarios exhibit a scale separation between the probe and background wavelengths by many orders of magnitude. At the same time the spatial extension of the probe pulse is commonly quite small compared to the background field. Hence, the seemingly best option is to resolve only a fractional region of the whole simulation space with the high precision that is required to propagate the high-frequency probe pulse in order to simulate vacuum birefringence effects in three spatial dimensions and time.

The implementation of a dynamically resolution-adapting grid can render the simulation of high-frequency pulses more feasible. The computational load for a simulation of the prominent

low-frequency pump and high-frequency probe pulses setup to detect vacuum nonlinearities can in principle be dramatically reduced by locally varying grid resolutions.

Simulating all waves from their initial to final positions on a uniform grid, whose resolution must be adequate for the highest involved frequency, incurs a very large and unnecessary overhead. Instead, dynamical grids are capable of reducing the overall numerical load significantly by ensuring a lower resolution in regions where at a given time only the low-frequency pump pulse or no pulse at all is present. This strategy does not diminish the overall accuracy and is the way to achieve the maximum reduction of computational load from a conceptual point of view.

Ultimately, a fully automatically adapting grid is envisioned, which changes resolutions in certain regions on demand and on the fly with a machine-learning based inference technique, employing an AI spectral FFT analyzer.

The challenge with an explicit solution of the numerical multi-scale problem using dynamical grids is that topologically complex communication interfaces at grid resolution boundaries, an inline diagnostic sensing the evolution of dynamic scales, and advanced load balancing are required. This can be achieved with the use of adaptive data structures and integrators combined with novel machine learning concepts.

In order to render the methods dynamic, it appears that interpolations are required for the transition to finer grids, while for the transition to coarser grid points might be omitted. The way these interpolations are performed, presumably relying heavily on machine learning methods, is part of active research. Focus has to be put on maintaining the stability of the algorithm on dynamic multi-scale grids. This could become problematic using interpolations [216].

7.2.1 A static resolution barrier

In order to demonstrate that the solver presented in this work is capable of multi-scale simulations, the problem of transmission and mode reflection at grid resolution boundaries is investigated at the hand of the specific dispersion properties of the numerical scheme on a static grid. The idea of an adaptive grid is thereby mimicked with a static resolution barrier inserted into the lattice.

This can be implemented using different grid resolutions on neighboring patches. To the end of load balancing, the number of points is kept constant among the patches, while the physical length is changed.

1D Simulations are performed in this basic multi-scale form and the validity of employing a grid with regions of varying resolution is investigated. The dispersion of the waves, especially in the vicinity of the barrier, is studied. Reflection and error behaviors of the waves are investigated with the purpose of quantifying the validity of more sophisticated future implementations.

The finite difference method works well for regular grids, but it is a nontrivial task to connect two grids with different resolutions. Depending on the underlying algorithms, changing the resolution can lead to numerical artifacts at the transition regions. Another option is to adapt the finite differences at the transition region, taking into account varying step sizes on the fine and coarse grid parts [216]. This is a more sophisticated and promising approach that can, per contra, introduce instabilities.

Recapitulation of the dispersion relations

A short summary of the dispersion relation is in order. The dispersion relation is symmetric in k for all stencils and hence the phase velocity equal in both directions. All used stencils are listed

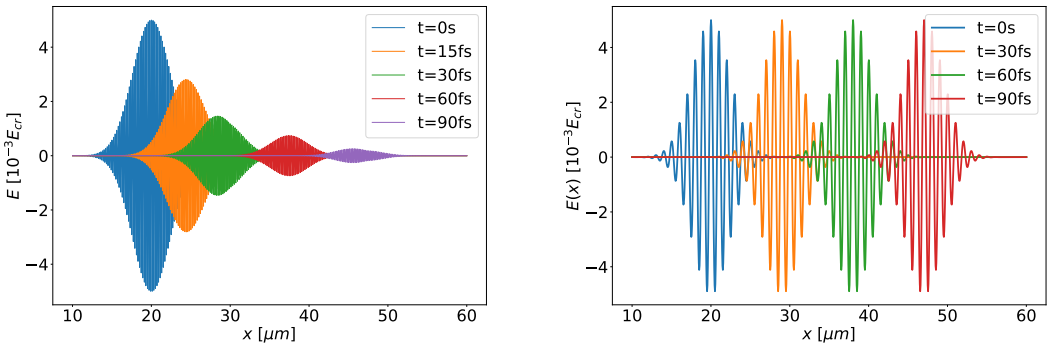


Figure 7.1: Amplitude damping with stencils of order thirteen. *Left*: Time evolution for a Gaussian wave packet with a wavelength corresponding to half the Nyquist frequency. There is a strong damping due to the dispersion relation. *Right*: Time evolution for a Gaussian wave packet with a wavelength corresponding to a tenth of the Nyquist frequency. No damping is observed with the sole eye.

in Chapter 2. Higher orders defer and decrease the rise of the imaginary parts. A stronger bias causes more absorption on the grid and the phase velocity to become superluminal as can be seen in Figure 2.6 at the even orders of accuracy, which are more imbalanced.

The wavelength/frequency regions where the dispersion relation deviates heavily from the linear vacuum form a *critical regime*. General dispersion effects for Gaussian wave packets in the critical and non-critical regime are visualized in Figure 7.1.

Note that the simulation of a high-frequency wave does not overshoot the Nyquist limit, but will be sampled as a wave with lower frequency, as explained in Chapter 2 and demonstrated in Figure 2.3. The frequency axes of the dispersion relation plots in Figure 2.6 thus range only up to the Nyquist limit, but the algorithm is stable for any frequency.

Stability at transition regions

The presumption is thus that there is no danger for a wave with a given wavelength to get a blown up amplitude even after crossing some resolution barrier. The resolutions to the right of a resolution barrier will be fourfold lower in the forthcoming discussion. The simulations are tested with the highest available stencil order thirteen.

By crossing the resolution barrier from a fine to a coarse grid the wave might enter the strongly nonlinear regime of the dispersion relation, the critical regime. This is visualized in Figures 7.2 for the same wave packets as in Figure 7.1.

It can easily be understood that a stronger damping can be observed when the wave transitions into a region of lower resolution. This is shown for the extreme case in the zoomed-in Figure 7.3.

Reflection at the barrier

As a consequence of energy non-conservation there is no defined relation between transmitted and reflected wave fractions at resolution barriers. An investigation of reflection effects is performed via an analysis of the evolution of wave amplitudes, see Figure 7.4, and by a time-step analysis of the reflected fractions, see Figure 7.5.

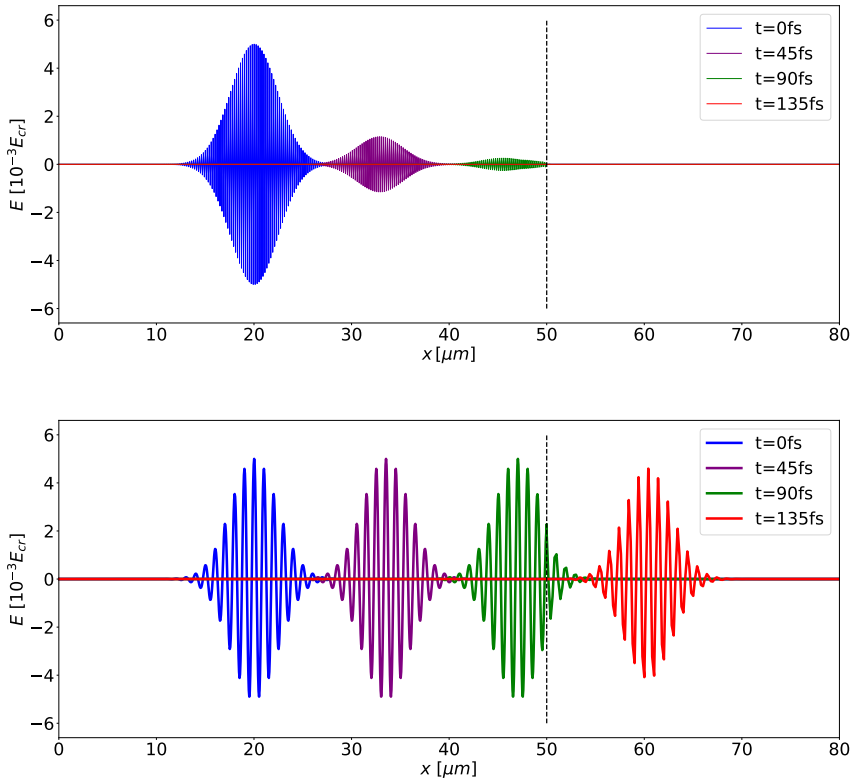


Figure 7.2: Evolution of the waves from figure 7.1 crossing a grid resolution barrier. The above wave is very strongly damped and to the right of the barrier is extincted quickly. What looks like a hard cut is indeed a strong damping as can be seen in figure 7.3. The wave below when passing the barrier experiences a noticeable damping. See figure 7.4 for the evolution of the amplitudes over time.

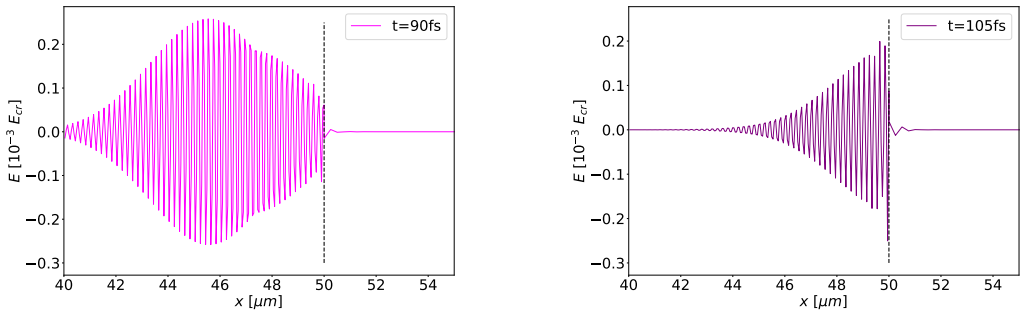


Figure 7.3: Strong damping after crossing the resolution barrier of an incoming wave with a wavelength corresponding to a half of the Nyquist frequency.

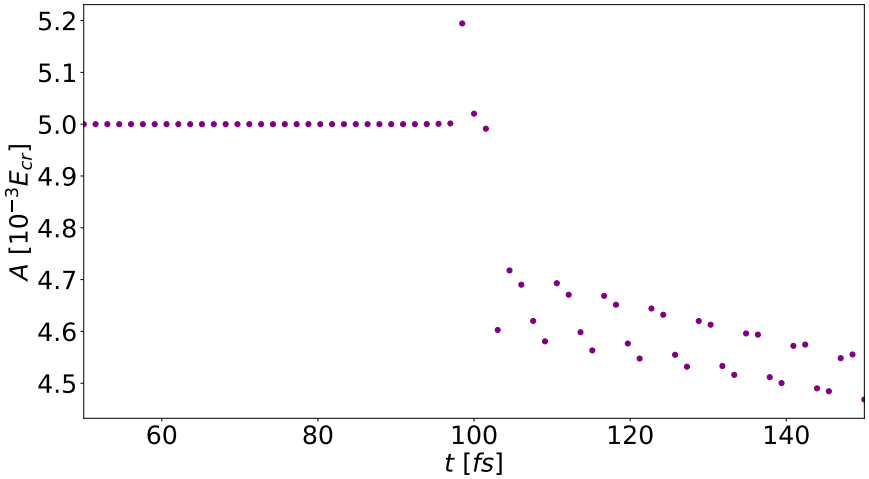


Figure 7.4: Peak amplitude evolution of a Gaussian wave packet with a tenth of the Nyquist frequency, hitting the resolution barrier with the packet center at 100 fs. There is a pile-up at the barrier which relaxes quickly, see Figure 7.5. Right from the barrier a bouncing of the amplitude can be observed. This comes down to the inaccurate modeling of a Gaussian wave packet when the frequency is large compared to the lattice resolution. In less severe form the latter effect can also be observed in Figure 4.13.

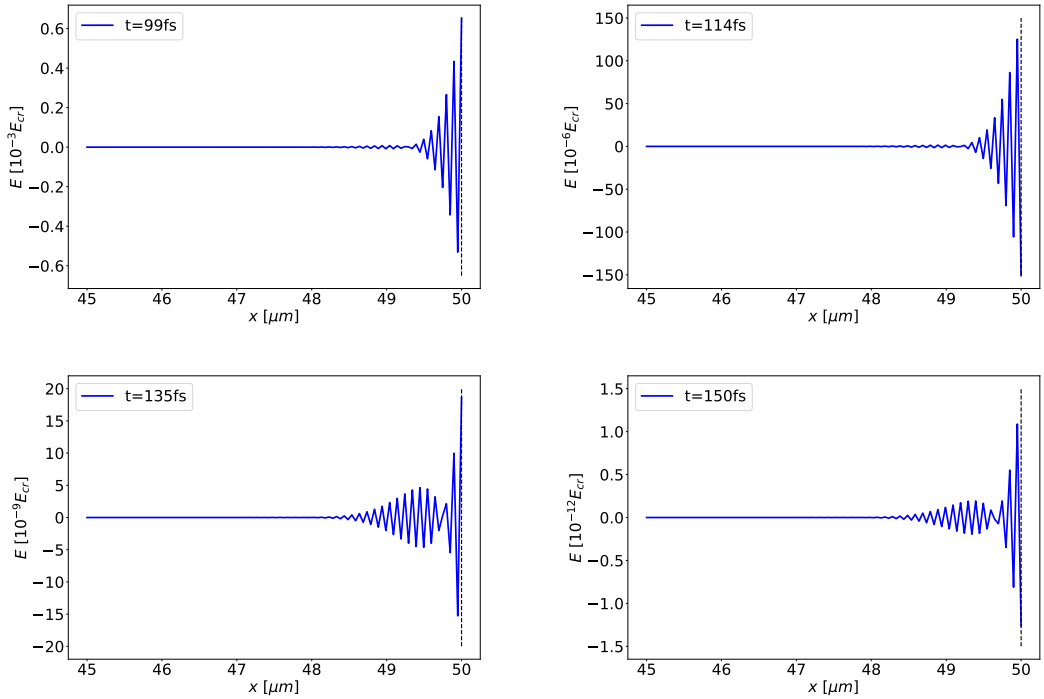


Figure 7.5: Reflected wave parts for a Gaussian packet modeled with a tenth the Nyquist frequency after hitting the resolution barrier. The reflections are obtained by subtracting the same simulation on a uniform grid. The reflection is diminished step by step.

The fact that the reflection pile up is damped away quickly is constructive. There is thus no noticeable disturbance to be expected through reflected wave fractions. Even harsher dampings of the reflections occur for incoming waves with smaller wavelengths. Waves with very low frequencies can pass a resolution barrier nearly undisturbed.

The current analyses prove the dispersion to be stable on both sides of a resolution barrier and reflections to be quickly absorbed, paving the way for dynamical solutions to be implemented. Further analysis material, such as videos of simulations not covered here can be found in the *Mendeley Data* repository [150].

Reflection-less static resolution barrier

Another approach of implementing a static resolution barrier is to adapt the finite differences at the transition regions. Using only the grid points from the regular grid that form the non-regular grid can annihilate any reflection.

In the case of a two-fold smaller resolution to the right of a barrier one would use at transition regions instead of the grid points $(-1,0,1)$ the points $(-1,0,2)$. This yields new stencils that have to be derived for every point in the transition region. With this, the discretization of the grid is dependent on the position and hence also the dispersion relation is grid point-dependent.

Taking this approach, no numerical defects could be detected [216]. It has been tested for resolution jumps of a factor of two or one half. A downside of this method is that it was found that it becomes unstable for stencil orders higher than six and lower than twelve, generating an amplifying imaginary part for relevant frequencies.

7.3 Phase-space approach to the quantum vacuum

The optical properties of the nonlinear quantum vacuum represent one aspect of a more complete, self-consistent formulation of a relativistic dynamical many-body quantum theory.

Within reach over the next years appears the formulation and numerical implementation of a consistent, relativistic quantum mean field theory. The latter is a prerequisite for novel applications and is sufficiently remote from the validity range of scattering theory, which is not capable of capturing collective effects and dynamical properties of the quantum vacuum [217].

It is clear that the optical properties of the quantum vacuum have to be coupled to dynamical matter fields beyond the scope of the effective theory approach at a subsequent stage. While probe pulses with ever shorter wavelengths and pump pulses with ever higher intensities enhance vacuum effects, they might destabilize the quantum vacuum and pair creation can be seeded, c.f. Section 1.3.

Advanced numerical methods are required to investigate the problem of vacuum stability. A numerical code capable of simulating at the onset of vacuum instability needs to combine the effects of the nonlinear quantum vacuum, seeded pair production, and seeded radiative emission in a strong electromagnetic background.

There are detriments of the perturbative approach that can be resolved with another, more fundamental method. The perturbative approach is deficient of a structural understanding of the vacuum [218]. Discovering the physics hidden in the vacuum itself requires methods exceeding the Schwinger limit.

7.3.1 The Dirac–Heisenberg–Wigner formalism

An elaboration based on the Dirac–Heisenberg–Wigner (DHW) formalism as a full-picture approach to vacuum dynamics seems promising [218–220]. The handy classical phase-space is no more available in quantum physics due to the incompatibility of position and momentum measurements manifested in the uncertainty relation. Nonetheless, there is a desire to set up quantum-valid expressions for average values of observables defined on phase-space, since relations to classical physics and measurable effects can be easier drawn and the physical intuition works better in a phase-space description [218].

The fundamental object providing a phase-space description of quantum theories is the Wigner operator $\hat{W}(x,p)$ that is defined somewhat ad-hoc as a Fourier transform of the density operator in coordinate space [221–224]. In scalar QED, with the "scalar matter" and radiation field operators given by ϕ and A, it is expressed as

$$\begin{aligned}\hat{W}(x,p) &= \int \frac{d^4y}{(2\pi)^4} e^{-ipy}\, \phi(x)\; e^{\frac{1}{2}yD^\dagger} e^{-\frac{1}{2}yD}\; \phi^\dagger(x) \\ &= \int \frac{d^4y}{(2\pi)^4} e^{-ipy}\, \phi(x+y/2)\; \exp\left[ie\int_{-1/2}^{1/2} ds\, A(x+sy)y\right] \phi^\dagger(x-y/2)\,,\end{aligned} \tag{7.1}$$

with the gauge-covariant derivatives D_μ in the exponentials forming the Wilson line factor

$$U(A;x-y/2,x+y/2) = \exp\left[ie\int_{-1/2}^{1/2} ds\, A(x+sy)y\right]\,, \tag{7.2}$$

ensuring gauge-invariance.

The equations of motion are directly governed by the Klein–Gordon equation, such that the Wigner operator incorporates the complete picture of virtual particles and hence the whole dynamics of the vacuum without loss of information. The correlation function, which the Wigner operator is the Fourier transform of, is called the Wigner kernel,

$$\Phi(x,y) = \phi(x+y/2)U(A;x-y/2,x+y/2)\phi^\dagger(x-y/2)\,. \tag{7.3}$$

In the original form this was the Heisenberg–Dirac density matrix of quantum mechanics, introduced by Dirac [225] and already used in the context of vacuum polarization by Heisenberg and Euler [20]. The whole construct is therefore also called Dirac–Heisenberg–Wigner formalism.

The Wigner function is the normal ordered ensemble average of the Wigner operator. In spinor QED, introducing the spinor field ψ_α, the Wigner operator and function become [219],

$$\begin{aligned}\hat{W}_{\alpha\beta}(x,p) &= \int \frac{d^4y}{(2\pi)^4} e^{-ipy}\, \bar{\psi}_\beta(x) e^{\frac{1}{2}yD^\dagger} e^{-\frac{1}{2}yD} \psi_\alpha(x)\,, \\ W_{\alpha\beta}(x,p) &= \langle : \bar{\psi}_\beta(x)\delta^4(p-\hat{q})\psi_\alpha(x) : \rangle\,,\end{aligned} \tag{7.4}$$

where in the second line the y-integral has been performed and

$$\hat{q} = i/2(D-D^\dagger)\,. \tag{7.5}$$

This expression directly indicates that the trace over spinor indices of the Wigner function measures the Lorentz scalar density of electrons at space-time point x with four momentum p. The

dynamics are governed by the Dirac equation.

Hartree approximation

In order to handle the dynamics, simplifications are necessary in this formalism.

Since the ensemble average is defined by the density operator, if the gauge field is a quantum operator, the ensemble averaging of equations for the Wigner operator contains two-body interactions. This is because the radiation field operator is a functional of the fermion current operator. The two-body correlations in turn depend on three-body terms and so on, generating the BBGKY hierarchy [226].

The most common simplification is to apply a Hartree-type approximation, were the radiation field is treated as a classical mean-field. This approximation can be considered valid for strong but slowly varying electromagnetic fields, where quantum fluctuations of the radiation field should be negligible. In consequence of the "averaging over degrees of freedom", a many-body problem is reduced to a one-body problem transport theory. This results in a truncation of the BBGKY hierarchy at the one-body level. In this case, the Wigner operator is formally equal to the Wigner function.

The Wigner function is a natural and intuitive quantum analog to the classical density function. A closed differential equation for the Wigner function, which acts as a one-particle weight function, consequently serves as a transport equation that expresses the evolution of all macroscopic quantities of interest. Expectation values of observables are calculated via phase-space integrals in full analogy to classical physics.

Impact of the DHW formalism

Historical interest in relativistic quantum transport theory arose through the discovery of a new matter-state, the QCD quark-gluon plasma in a series of seminal works [227, 228] with a companion paper devoted to QED [219].

The Wigner formalism gives us a systematic framework to describe the structure of the QED vacuum independent of perturbation theory, at least mostly [218]. It facilitates the unification of three hitherto disjoint theories [218]; Dirac's theory of the electron in external fields, Maxwell's theory of electromagnetism, and the relativistic kinetic theory of charged particles.

Advantages of the DHW formalism are that it

- combines QED with known features of statistical physics;
- leads to transport equations in more general form than of quantum Vlasov kind, important for many applications in many-body physics [219];
- yields a complete phase-space description, including created electron–positron pairs [50, 229];
- reveals many novel signatures for understanding the complicated physics in pair production but also helps to guide the possible future experimental realization [230];
- facilitates simple expressions for observables; and
- is, most importantly, fully general, encompassing QFT.

It is important to know the trajectories of created electrons and positrons to exclude subsequent recombinations and formations of photons.

The expectation value of the current induced by an external field represents nonlinearities, since it also depends on the vacuum feedback [22]. The particles and antiparticles are accelerated by the background field, thereby creating an opposed current which in turn attenuates the external field. The external field acts as a control parameter, the back-reaction responds accordingly. Hence, the current is usually separated into an external part induced purely by the applied field and an internal vacuum response [231, 232].

Studying the phenomenon of pair creation non-perturbatively in phase-space had gained popularity around the nineties [218, 220, 233–235] and again in the 2010-years [236–238]. A downside of the DHW approach is the absolute need for numerical techniques, since only a few analytical results are available [236]. Yet, this pressure might spur the advancement in times of higher computing power. Some significant progress in the field has been achieved with numerical aid over the recent years [3, 24, 50, 229, 239].

7.3.2 Numerical approach to a seminal work

There remain major difficulties deriving consistent transport equations and it is computationally expensive to obtain realistic numerical solutions [223, 239]. An expansion formalism in terms of a signal photon theory alike the Vacuum Emission Picture is a promising approach. Phase-space is large and high-dimensional, hence, simplifying assumptions and lower-dimensional or symmetric scenarios have to be selected to render computations feasible [3, 24]. Besides, slicing space and employing parallelization techniques, such as they are used in the effective action approach, supposedly will render the DHW-based calculations more realistic.

In one of the seminal works the simple configuration of a spatially homogeneous but time-dependent purely electric field in scalar QED was considered [220]. An oscillator equation describing the process of pair creation was obtained,

$$\partial_t^2 \zeta + E_p^2 \, \zeta = 0 \, , \tag{7.6}$$

with $E_p = \sqrt{p(t)^2 + m^2}$. Here, $|\xi|^2 = W_3$ is the *energy-averaged* Wigner function describing a three-dimensional phase-space formulation [220],

$$W_3 = \int dp_0 \, W \, . \tag{7.7}$$

The initial condition is given by $|\xi|^2(t = -\infty) = 1/E_p(t = -\infty)$. This type of equation has also been found in [235]. Enhancing graphical visualizations can be obtained for various scenarios, e.g., the ones described in Figures 7.6 and 7.7.

In ongoing investigations parallel to the Heisenberg–Euler solver, a solver for the case of scalar QED is devised, based on the fundamental equations of motion for the Wigner function, without making use of energy-averaging. As explained above, these fundamental equations derive directly from the Klein–Gordon equation and hence the approach amounts to solving the complete field theory. The solver is intended to include the magnetic field as well.

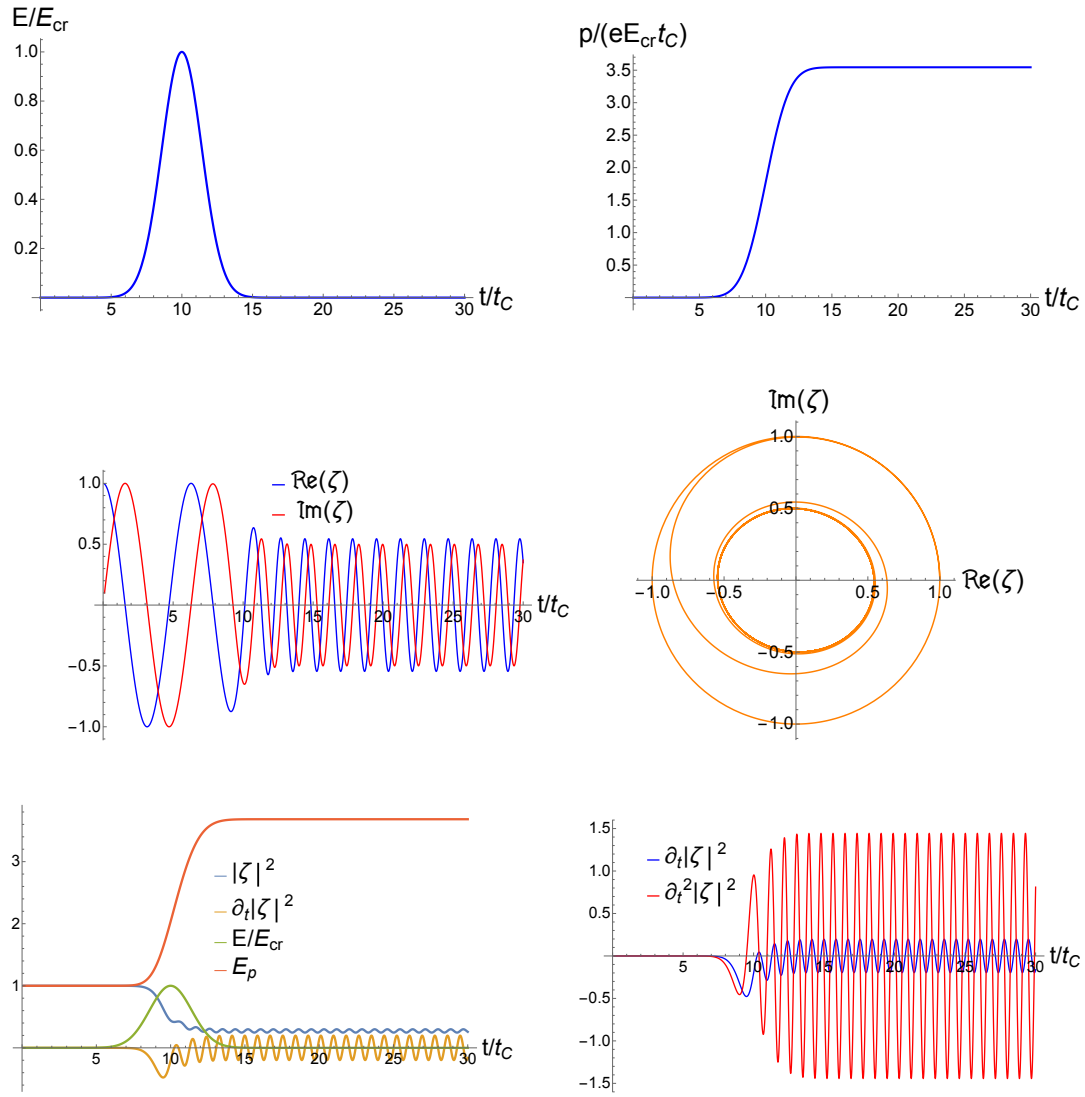


Figure 7.6: Polarized vacuum. *Top left*: amplitude of a switched-on electric field with a peak value of E_{cr}. *Top right*: total momentum (normalized) according to the Lorentz force with the initial momentum set to zero. *Center left*: oscillation of the real and imaginary parts of ζ. The amplitudes begin to decrease as the electric field is turned on and reach a new state after the field has vanished. As a result of the different initial conditions, the imaginary part decreases faster and farther. *Center right*: Corresponding Lissajou ellipsis of the coupled oscillator over time. At excitation (turning on of the electric field) one can see the intake of energy, hence the compression. *Bottom left*: combined time evolutions of $|\zeta(t)|^2$, its derivative, the electric field and the energy E_p. *Bottom right*: first and second derivatives of the energy-averaged Wigner function.

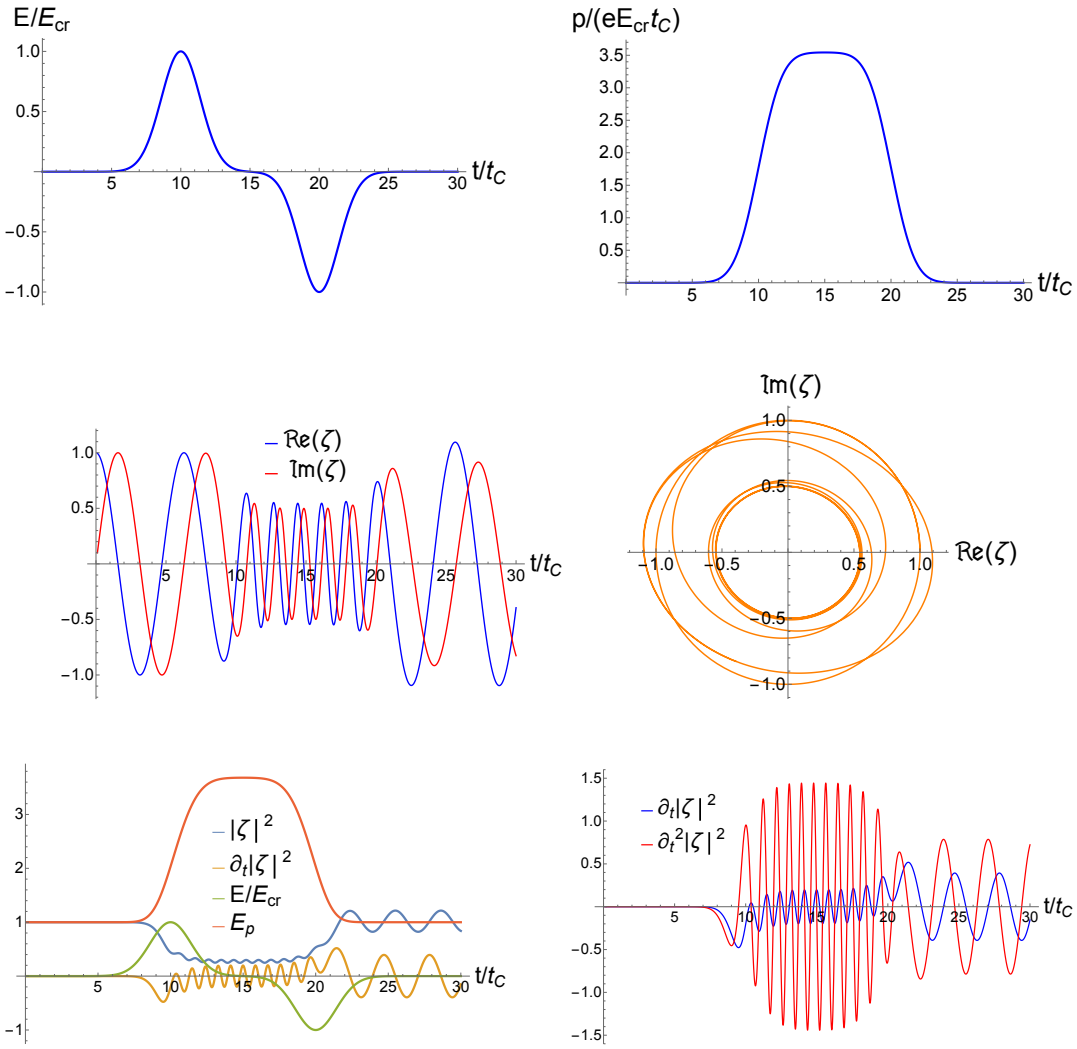


Figure 7.7: Restored vacuum. Shown are the same quantities as in Figure 7.6 for the scenario with a second pulse with opposite polarization. Strikingly, the latter pulse takes energy out of the system. Ideally, the oscillator would end up on the initial circle before the first pulse in the Lissajou ellipses. This, however, requires a careful tuning of the pulses. The concept resembles stimulated emission.

Chapter 8

Conclusion

This final chapter summarizes what has been achieved and what may follow in the context.

Simulations in the weak-field Heisenberg–Euler quantum vacuum look back at several years of tradition in the working group of Hartmut Ruhl. The first version of the simulation code was presented with 1D capability in the course of the PhD thesis of Patrick Böhl [113]. The algorithm was restructured and a new code version developed within the PhD project of Arnau Domenech [144] and the first results in 2D were produced. Full 3D capability is incorporated and demonstrated in this thesis.

New modules have been implemented and the code has been refactored with the purpose of modernization and optimization. Emphasis is put on high-performance computing to leverage the power of supercomputers for expensive simulations. To this end, a hybrid multiprocessing plus multithreading model, nowadays functioning as the workhorse on massively distributed systems, has been integrated into the code. This enables extreme scalability and hence to obtain more realistic simulation results. In the process, the performance of the solver has been analyzed.

The underlying algorithm is concisely outlined in [180], the first publication of the new solver. With science relying increasingly on computer simulations, software has become an important academic material. Having undergone a full refurbishment, the software project is now published open source under the name *HEWES* in [148]. The publication opens up the possibility for researchers in the community to use it as a tool and monitor further development. Within the limits of the Heisenberg–Euler weak-field approximation the solver represents an efficient numerical approach to the complete dynamical response of the nonlinear vacuum for complicated pulse setups in all three dimensions.

The validity of the presented numerical scheme for solving the modified Maxwell equations in the Heisenberg–Euler weak-field expansion relies on two basic assumptions: I) field strengths below the critical values E_{cr} and B_{cr}, and II) wavelengths larger than the Compton length of the electron.

An established solver of the Heisenberg–Euler dynamics can be very useful for strong-field QED research going on at present. There have been other approaches in the area of numerical solvers for the nonlinear quantum vacuum, put forward in [72, 143] and [165]. The one discussed in the present thesis stands out with a very high order of accuracy of the numerical scheme and the inclusion of six-photon processes, and is thus extremely precise. Of paramount significance is the dispersion relation, lying at the heart of the algorithm, which ensures stability throughout the

frequency spectrum and moreover creates an imaginary part that annihilates nonphysical modes. Furthermore, the linear vacuum-like behavior of the dispersion relation for a large frequency range is an essential ingredient.

Every simulation of the Heisenberg–Euler model incorporates the complete nonlinear physics in the weak-field approximation, whereas analytical calculations normally have to concentrate on single, isolated effects, leaving others aside, and hence often miss the complete picture of the interaction. By taking into account the whole dynamics of the nonlinear vacuum, the solver captures in particular back-reactions to the radiation fields. Withal, simulations permit to describe the temporal evolution of nonlinear vacuum processes, which is beyond the reach of many analytical approaches.

In order to set the stage for a soon-to-be-expected experimental verification of all-optical nonlinear quantum vacuum phenomena, theoretical predictions are required. Universality with respect to pulse configurations is the main raison d'être of the numerical approach as a complement to theoretical treatments. Analytical methods rely on simplifying assumptions and are in consequence most of the time limited to special scenarios. Practicable calculations are constrained to simple configurations and arrangements of the involved laser pulses, or neglect important properties of the quantum vacuum. Any such approximation in turn limits the accuracy of predictions and the precision with which theory can be tested [143].

The exploration of parameter regimes and the estimation of expected signals that should be detectable in experiments will be supported by numerical tools. Moreover, only computer-driven approaches are flexible enough to guide the development of experimental constructions and configurations in the research area of strong-field QED. A shift in perspective to accompany the numerous analytical treatments with versatile numerical solutions is apparent. Phenomena that are in prospect to profit from *HEWES* simulations are the investigated effects of vacuum birefringence and the generation of higher harmonics, and light-by-light scattering effects in general.

A good agreement with analytical results is achieved by making use of high orders of the numerical scheme and high expansion orders of the effective Heisenberg–Euler model. The computational cost scales strongly with the number of lattice points but weakly with the discretization order of the scheme and the expansion order of the Heisenberg–Euler model. The impact of the discretization order on the computational cost, however, does become relevant in 3D with increasing cluster computer communication for the transfer of spatial derivative data. Making use of high discretization orders and their favorable dispersion relations facilitates the use of comparably small lattices to accurately model the involved electromagnetic modes. Nonetheless, for high-frequency pulses the required grid resolutions can become unfeasible.

Ideas are being developed in order to overcome the obstacle of extremely large 3D grids. One promising, ongoing, and important project is on multi-scale simulation capability. This research is intended to pave the way for a dynamical grid, adapting its resolution regionally on the fly and on demand in order to reduce the computational load, while at the same time maximizing the accuracy in important spatial regions. An adaptive grid is particularly suited for the prominent low-frequency pump pulse and high-frequency probe pulse setup to detect vacuum nonlinearities, where the computational load can in principle be dramatically reduced.

It should be emphasized again that not only the high-energy, low-intensity regime discovered in particle accelerators, but also the low-energy, high-intensity regime might be a path towards new physics, since quantum fluctuations in the vacuum consist of all existing particles [14–16].

Appendix A

KCS System Information

The cluster computing system *KCS* of the Arnold Sommerfeld Center (ASC) for Theoretical Physics in Munich is hosted at the Leibniz Rechenzentrum (LRZ). It is a homogeneous system, i.e., the cluster consists of equal nodes and, in as opposed to heterogeneous systems, there are no accelerators like GPUs attached. The ASC hosts an own cluster consisting of the bureau workstation computers. The latter is an inhomogeneous cluster with different CPU types, which is clearly more difficult to handle. Moreover, distributed computations suffer from slower node interconnects.

Topology

KCS contains 153 nodes, however, as mentioned in Chapter 5, a maximum of 121 nodes can be occupied per simulation. This can be seen from the partition information in Listing A.1, obtained via the batch job scheduling system discussed further below.

```
PartitionName=kcs_batch
Nodes=hlcpr06c01s[01-10],hlcpr07c01s[01-10],hlcpr06c02s[01-10],hlcpr07c02s[01-10],
    hlcpr06c03s[01-10],hlcpr07c03s[01-10],hlcpr06c04s[01-10],hlcpr07c04s[01-10],
    hlcpr06c05s[01-10],hlcpr07c05s[01-10],hlcpr08c01s[01-10],hlcpr08c02s[01-10],
    hlcpr08c03s[01-10],hlcpr08c04s[01-10],hlcpr08c05s[01-10],hlcpr05c04s[07-09]
   TotalCPUs=4896 TotalNodes=153
   MaxNodes=121 MaxTime=3-00:00:00
   OverTimeLimit=NONE State=UP
```

Listing A.1: Configuration of the *KCS* batch partition (excerpt). The list of nodes is given. While the system has 153 nodes in total, only 121 of them can be used for a single job. The maximum runtime for a job is limited to three days.

Each node consists of 32 cores, 16 cores per socket, c.f. Figure 5.1, with each more than 180 GB memory. A selection of nodes with information is listed in Listing A.2.

```
NODELIST       PARTITION    CPUS   S:C  MEMORY
hlcpr05c04s07  kcs_batch*   32     2:16 184939
hlcpr05c04s08  kcs_batch*   32     2:16 765356
hlcpr05c04s09  kcs_batch*   32     2:16 765356
hlcpr06c01s01  kcs_batch*   32     2:16 378484
```

Listing A.2: Information for a selection of nodes. Each node consists of 32 cores (here referred to as CPUs), on two sockets with 16 cores each. The nodes on the *KCS* system, while being equipped with the same processors, have different sizes of memory attached.

In Listing A.3 and Figure A.1 the topology of a node on the cluster is detailed, i.e., the layout of cores and caches, and their specifications. The information is generated with the help of *LIKWID* [193].

```
CPU name: Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz
CPU type: Intel Skylake SP processor
CPU stepping: 4
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:    2
Cores per socket: 16
Threads per core: 2
--------------------------------------------------------------------------------
Socket 0:   ( 0 32 1 33 2 34 3 35 4 36 5 37 6 38 7 39 8 40 9 41 10 42 11 43 12 44 13 45
     14 46 15 47 )
Socket 1:   ( 16 48 17 49 18 50 19 51 20 52 21 53 22 54 23 55 24 56 25 57 26 58 27 59
    28 60 29 61 30 62 31 63 )
--------------------------------------------------------------------------------
********************************************************************************
Cache Topology
********************************************************************************
Level:      1
Size:     32 kB
Type:     Data cache
Associativity:    8
Number of sets:   64
Cache line size:  64
Cache type:   Non Inclusive
Shared by threads:  2
Cache groups:   ( 0 32 ) ( 1 33 ) ( 2 34 ) ( 3 35 ) ( 4 36 ) ( 5 37 ) ( 6 38 ) ( 7 39 )
     ( 8 40 ) ( 9 41 ) ( 10 42 ) ( 11 43 ) ( 12 44 ) ( 13 45 ) ( 14 46 ) ( 15 47 ) ( 16
     48 ) ( 17 49 ) ( 18 50 ) ( 19 51 ) ( 20 52 ) ( 21 53 ) ( 22 54 ) ( 23 55 ) ( 24 56
     ) ( 25 57 ) ( 26 58 ) ( 27 59 ) ( 28 60 ) ( 29 61 ) ( 30 62 ) ( 31 63 )
--------------------------------------------------------------------------------
Level:      2
Size:     1 MB
Type:     Unified cache
Associativity:    16
Number of sets:   1024
Cache line size:  64
Cache type:   Non Inclusive
Shared by threads:  2
Cache groups:   ( 0 32 ) ( 1 33 ) ( 2 34 ) ( 3 35 ) ( 4 36 ) ( 5 37 ) ( 6 38 ) ( 7 39 )
     ( 8 40 ) ( 9 41 ) ( 10 42 ) ( 11 43 ) ( 12 44 ) ( 13 45 ) ( 14 46 ) ( 15 47 ) ( 16
     48 ) ( 17 49 ) ( 18 50 ) ( 19 51 ) ( 20 52 ) ( 21 53 ) ( 22 54 ) ( 23 55 ) ( 24 56
     ) ( 25 57 ) ( 26 58 ) ( 27 59 ) ( 28 60 ) ( 29 61 ) ( 30 62 ) ( 31 63 )
--------------------------------------------------------------------------------
Level:      3
Size:     22 MB
Type:     Unified cache
Associativity:    11
Number of sets:   32768
Cache line size:  64
Cache type:   Non Inclusive
Shared by threads:  32
```

```
Cache groups:   ( 0 32 1 33 2 34 3 35 4 36 5 37 6 38 7 39 8 40 9 41 10 42 11 43 12 44
    13 45 14 46 15 47 ) ( 16 48 17 49 18 50 19 51 20 52 21 53 22 54 23 55 24 56 25 57
    26 58 27 59 28 60 29 61 30 62 31 63 )
--------------------------------------------------------------------------------
********************************************************************************
NUMA Topology
********************************************************************************
NUMA domains:   2
--------------------------------------------------------------------------------
Domain:     0
Processors:   ( 0 32 1 33 2 34 3 35 4 36 5 37 6 38 7 39 8 40 9 41 10 42 11 43 12 44 13
    45 14 46 15 47 )
Distances:    10 21
Free memory:    92412 MB
Total memory:   95351.3 MB
--------------------------------------------------------------------------------
Domain:     1
Processors:   ( 16 48 17 49 18 50 19 51 20 52 21 53 22 54 23 55 24 56 25 57 26 58 27 59
     28 60 29 61 30 62 31 63 )
Distances:    21 10
Free memory:    94784.3 MB
Total memory:   96760.9 MB
```

Listing A.3: *KCS* topology. The processor type and specifications are given at the top. The numbers of hardware threads and cores per socket are listed and a detailed mapping of the hardware threads to the cores is given, which is visualized in Figure A.1. The specifications of the caches list the cache line size of 64 bytes. The NUMA topology at the bottom indicates that there are two NUMA domains, the sockets. There is a memory unit with more than 90 GB attached to each of the two sockets, which indicates the non-uniform memory access (NUMA) for the cores on the node.

Compilers

Although the system is being upgraded regularly to be kept up to date, it is in order to give a summary of the system specifications and the employed software environment. The versions listed are those used at the time of writing.

The operating system the *KCS* is running on is *SUSE Linux Enterprise Server 15 SP1*. The compiler used for most performance analyses is *ICPC* version 2021.4.0 (*GCC* version 10.3.0 compatibility) from the *Intel® MPI* Library 2021.4 for *Linux*. *Intel®* compilers have been using the GNU tools on clusters; header files, libraries, and linker. This is called the *Intel®* and GNU compatibility and interoperability. The next generation *Intel® C++* compiler *ICX*, based on *Clang/LLVM* has recently been adopted in the *Intel® oneAPI* toolkit in 2022 [196]. Instead of the *GCC libstdc++* libraries it relies on *Clang libc++*. For the latest simulations on *KCS*, the latter compiler was employed to build the executables. Sometimes, *GCC* version 11.2 was used on the cluster.

Compiler options

Since *C++20* is required, the compiler option `-std=c++20` is mandatory. Further, the optimization level two `-O2` is strictly recommended, which contains function inlining and loop alignment to a large extent. The higher optimization level three can be advantageous for programs with loops that perform many floating-point calculations or process large datasets, but is not used here, as it did not yield noticeable speedup and is unsafe when threading is used.

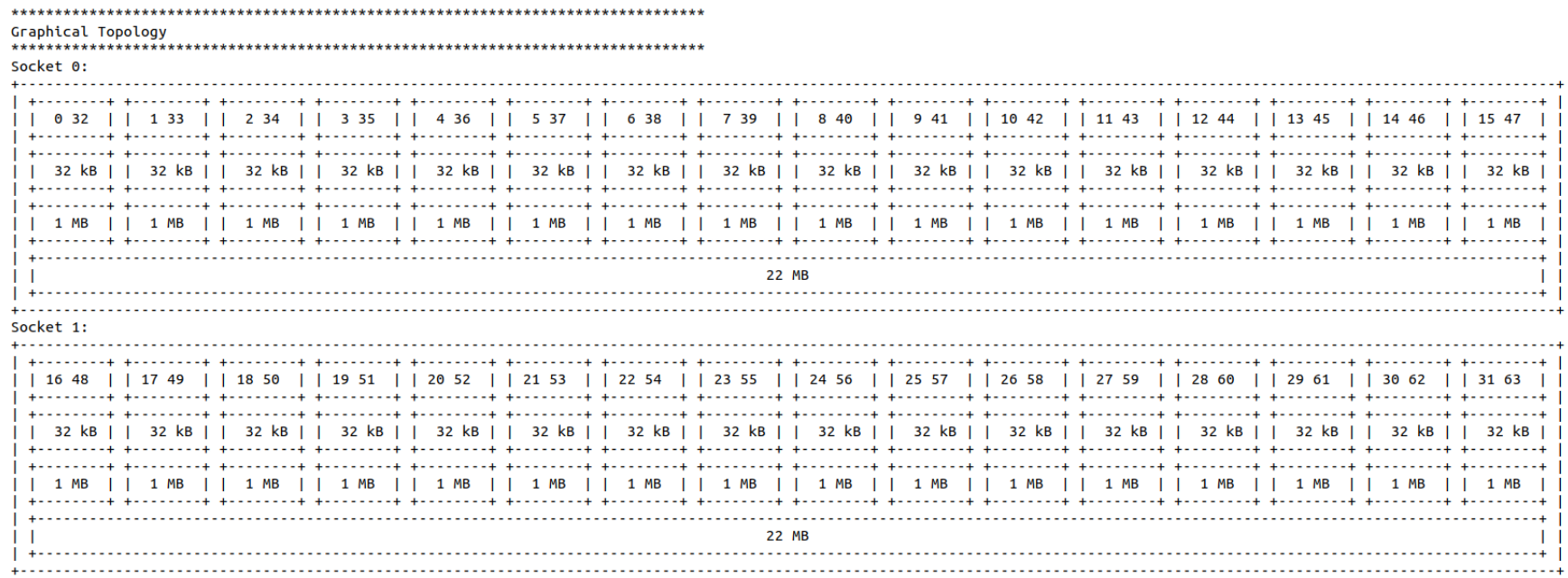


Figure A.1: *KCS* graphical topology. A node consists of two sockets, each with 16 cores. Each core can be divided into two hardware threads. The first row of a socket lists the two threads per core. Level one and level two cache are core-private. Their capacities are given in rows two and three. The level three cache in row four is shared upon a socket. More detail is provided in Listing A.3.

Depending on the performed tests, various additional compiler options can be beneficial. In order to enable the usage of the large 512-bit (zmm) registers for vectorization, the *Intel*® advanced vectorization extension instructions for 512-bit, `-xCORE-AVX512`, are used together with `-qopt-zmm-usage=high`. These are available for *Skylake*-type CPUs, c.f. Listing A.3. The code provably profits from 512-bit register usage, as described in Section 5.4.1. Enforcing SIMD vectorization with *OpenMP* compiler directives requires to include and link *OpenMP*.

Lower advanced vectorization extension SIMD features are automatically put into place where the compiler assumes that too aggressive vectorization is disadvantageous. Since there is no pointer aliasing, as found in Section 5.4.1, `-fno-alias` can be used as a hint to the compiler. Aggressive vectorization could be achieved by telling the compiler to wave its doubts with `-vec-threshold0`.

Module system

Users of HPC systems commonly have a large pool of software at their disposal. In order to make the required software ready for use on an HPC system, usually the module system is employed. All available software can be listed and searched and the desired tools can be loaded. The module system takes care of setting the corresponding system and environment variables and at the same time avoids clashes. It further ensures that all required dependencies are loaded. The loaded software stack can be checked at any time. An example of an environment is given in Listing A.4.

```
Currently Loaded Modulefiles:
 1) admin/1.0(default) <S>           13) intel-oneapi-vtune/2021.7.1
 2) tempdir/1.0(default) <S>         14) intel-oneapi-advisor/2021.4.0
 3) lrz/1.0(default) <S>             15) intel-oneapi-itac/2021.5.0
 4) spack/22.2.1(default)            16) numactl/2.0.14-intel21
 5) intel-mkl/2020(default)          17) likwid/5.2.0-intel21
 6) intel-oneapi-compilers/2022.2.0  18) perl/5.34.0 <aL>
 7) intel-mpi/2019-intel(default)    19) texlive/2019 <aL>
 8) llvm/13.0.0                      20) gnuplot/5.4.2-X11 <aL>
 9) scorep/7.0-gcc10-impi            21) darshan-runtime/3.3.1-intel21-impi
10) papi/6.0.0.1-intel21             22) darshan-util/3.3.1-gcc11
11) cube/4.6                         23) cmake/3.21.4
12) scalasca/2.6-gcc9-impi           24) valgrind/3.17.0-gcc11-impi

Key:
(symbolic-version)  <module-tag>  <aL>=auto-loaded  <S>=sticky
```

Listing A.4: A module environment. Besides the “sticky” modules that are always pre-loaded on the system, this enviroment uses the *Intel*® *MPI* compiler wrappers and the next generation compiler that works with *LLVM*. Furthermore, some of the tools used in Chapter 5 are loaded. Auto-loaded modules are those that are autmoatically loaded as prerequisites for another loaded module.

SLURM (Simple Linux Utility for Resource Management)

Simulations are submitted as jobs to the *KCS* cluster and scheduled with the help of the *SLURM* cluster workload management system [240]. Options are available to distribute the jobs on specific numbers of nodes and/or CPUs for parallel programs. Allocation of resources is automatically managed by *SLURM* at the hand of the chosen parameters. An excerpt of a *SLURM* file for *KSC* is given in Listing A.5.

```
#!/bin/bash

#############################################################
# Slurm File SBATCH Commands
#############################################################

#SBATCH -o /dss/dsshome1/lxc0A/ru68dab/slurm/slurm_out/%j.%N.out  # stdout file
#SBATCH -e /dss/dsshome1/lxc0A/ru68dab/slurm/slurm_out/%j.%N.err  # stderr file
#SBATCH -D /dss/dsshome1/lxc0A/ru68dab/repos/heisenberg_euler/src # work dir
#SBATCH -J HEWES  # Job name
#SBATCH --partition=kcs_batch  # Specifiy kcs for resource allocation
#SBATCH --mail-type=begin,end  # email at begin and end of job
#SBATCH --mail-user=and.lindner@physik.uni-muenchen.de  # email recipient
#SBATCH --time=15:00:00  # run time limit
#SBATCH --ntasks=512  # maximum number of tasks provided for resources
#SBATCH --ntasks-per-node=8  # maximum number of tasks per node
#SBATCH --ntasks-per-socket=4  # maximum number of tasks per socket
#SBATCH --nodes=64  # total number of nodes in resource allocation
#SBATCH --exclusive  # exclusive nodes
#SBATCH --cpus-per-task=4  # number of CPUs allocated per task (process)
#SBATCH --mem-per-cpu=1024  # minimum memory allocated per CPU in Megs


#############################################################
# Overall Environment Variables & Hardware Affinity Control
#############################################################
# Pass on variables for numbers of processes and threads
export MPI_NUM_PROCESSES=$SLURM_NTASKS
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Further OpenMP environment variables
export OMP_DYNAMIC=false  # not fewer threads than requested
export OMP_SCHEDULE=static,2  # thread runtime schedule
export OMP_WAIT_POLICY=active  # spinning instead of idle threads

# Variables to control hardware affinity
export OMP_PLACES=cores  # thread per core
export OMP_PROC_BIND=true  # bind thread to core or:
export KMP_AFFINITY=noverbose,granularity=thread,compact,1,0 (Intel runtime)
export I_MPI_PIN_DOMAIN=`expr 2 \* $OMP_NUM_THREADS` # hybrid pinning (Intel MPI)
export OMP_DISPLAY_AFFINITY=true  # display thread affinity

./run_3D_ex.sh
```

Listing A.5: Minimal *SLURM* file for *KCS*. Shown are basic configuration instructions for *SLURM* as well as some environment variables for hardware affinity control. At the end, a run script is executed.

Besides the pinning options with the help of *LIKWID* [193] mentioned in Section 5.3.1, there are, e.g., the *OpenMP* and *Intel*® specific environment variables used in Listing A.5.

However, since the inter-node bandwidth is lowest, the most crucial aspect of hardware affinity is formed by the distribution of allocated nodes. It is possible to tell *SLURM* which nodes should be used, but this can result in extremely long queuing times. Which specific nodes *SLURM* decides to allocate is therefore most of the time unpredictable.

Running jobs can be monitored and the resource consumption of finished jobs be analyzed.

SLURM further provides some useful metrics during and after the simulation. An example of *SLURM* accounting information is given in Listing A.6.

```
   JobName        JobID      Elapsed   NNodes      NCPUS   NTasks      MaxRSS
---------- ------------   ---------- ---------- -------- --------  ----------
Heisenber+ 325760           19:41:28        8        256
     batch 325760.batch     19:41:28        1         32        1      13684K
hydra_bst+ 325760.1         19:41:26        8          8        8  102585488K

   JobName        JobID      Elapsed   NNodes      NCPUS   NTasks      MaxRSS
---------- ------------   ---------- -------- ---------- --------  ----------
     HEWES 517918           02:31:01        8        256
     batch 517918.batch     02:31:01        1         32        1      20404K
hydra_bst+ 517918.0         02:30:56        8        256        8   15810988K
```

Listing A.6: SLURM accounting information for two different job that ran on 256 cores each. It can be chosen which fields of the many available job accounting fields should be displayed. There is, e.g., the rightmost metric that gives the maximium memory requirements per node. The computational and memory loads for the shown jobs differ considerably.

File systems

Cluster computers use parallel file systems optimized for large volumes of data, such as *GPFS* (general purpose file system) and *LUSTRE* (a compound of *Linux* and cluster). *GPFS* is employed for *KCS*, the storage of the cluster at the ASC relies on *LUSTRE*.

The `MPI_Info` object is a container of string-based key-value pairs that can be used to pass optimization hints to *MPI*, e.g., about file access patterns and file system specifications. The information can be set in an *MPI* call, via environment variables, or sometimes with a *ROMIO*-hints configuration file, which is at best configured by the file system vendor or maintainer. There is significant potential for performance benefits through file system-specific hints.

The hints may contain whether to allow collective buffering, i.e., the use of collective I/O and its tuning with respect to block size and buffer space, and more. Collective buffering reorganizes data across processes to match the data layout in the file. Data from sets of processes are to this end aggregated via *MPI* communication in order to produce larger chunks and fewer I/O operations [241].

Well proven hints according to the LRZ on the *SuperMUC* system are to set the buffer size and striping unit to 4 MiB [242],

```
MPI_Info_set(info,"romio_cb_write","enable", error) ,
MPI_Info_set(info,"cb_buffer_size","4194304", error) ,
MPI_Info_set(info,"striping_unit","4194304", error) ,
```

where `cb` is for collective buffering.

In parallel file systems, the chunk size should be a multiple of the block size of the file system. *GPFS* and *LUSTR*E have locking protocols that operate on blocks (*GPFS*) or pages (*LUSTRE*) of 4 MiB size per default. Parallel file systems thereby have a way larger block size than standard non-parallel file systems that commonly use 4 kiB. The scratch file systems at the LRZ have maximal bandwidths between 30 and 60 GB/s.

Appendix B

Code Modernization

Some software tools that assisted in the process of refactoring, optimizing and modernizing the code deserve being mentioned.

The source code has been tidied up and modernized with the help of *Clang-tidy* according to aspects of the *C++ Core Guidelines* and formatted in the *LLVM*-style with the help of *Clang-format*. Further, the *Clang Static Analyzer* has been used in conjunction with diagnostics to rectify unsafe code segments. Debugging was performed in the context of some subtle synchronization problems with the help of *LLDB*. All these tools are part of the *LLVM* project [243, 244].

Furthermore, compiler warnings and optimization reports of various vendors, *Clang*, *Intel*®, and *GCC*, were of great help in the process of code modernization and in order to get rid of dangerous flaws.

Extensive "consting" and using static declarations form valuable hints to the compiler to let the code work more efficiently. Further, a compactification of the time-evolution functions, where unnecessarily large arrays kept redundant data and loops copied already present values, was carried out. This modification decreased the memory load, while at the same time increasing the throughput.

The computational load could be reduced making use of *C++* performance tweaks. Modern *C++* features have been built in. E.g., use of the *Boost C++ Libraries* became superfluous thanks to the incorporation of many of its features into later *C++* standards [174]. In addition, the upgrade to the latest *CVODE* versions led to extended functionality and improved stability [168, 169].

The whole interface has been revamped to have all necessary control via command line arguments, outsourcing all further functionalities to modules. Therefore, the executable has to be built only one on a system. The building process is outermost simplified, automated and cross-platform compatible through the use of *CMake*[178].

Comment blocks are inserted into the code in order to generate a full code documentation with the help of *Doxygen* [153]. The code reference is made available in the public software repository [148].

List of Figures

List of Tables

List of Listings

Bibliography

[1] R. D. Mattuck, A guide to Feynman diagrams in the many-body problem, 2nd Edition, Dover Publications, 1992.

[2] F. Bissey, F.-G. Cao, A. R. Kitson, A. I. Signal, D. B. Leinweber, B. G. Lasscock, A. G. Williams, Gluon flux-tube distribution and linear confinement in baryons, Physical Review D 76 (11) (2007). doi:10.1103/physrevd.76.114512.

[3] C. Kohlfürst, Electron-positron pair production in inhomogeneous electromagnetic fields, Ph.D. thesis, U. Graz (2015). arXiv:1512.06082, doi:10.48550/arXiv.1512.06082.

[4] W. E. Lamb, R. C. Retherford, Fine structure of the hydrogen atom by a microwave method, Phys. Rev. 72 (1947) 241–243. doi:10.1103/PhysRev.72.241.

[5] P. J. Mohr, Lamb Shift in a Strong Coulomb Potential, Phys. Rev. Lett. 34 (1975) 1050–1052. doi:10.1103/PhysRevLett.34.1050.

[6] W. Johnson, G. Soff, The Lamb shift in hydrogen-like atoms, $1 \leq Z \leq 110$, Atomic Data and Nuclear Data Tables 33 (3) (1985) 405–446. doi:10.1016/0092-640X(85)90010-5.

[7] P. A. M. Dirac, The quantum theory of the electron, Proceedings of the Royal Society of London. Series A 117 (778) (1928) 610–624. doi:10.1098/rspa.1928.0023.

[8] H. Casimir, On the Attraction Between Two Perfectly Conducting Plates, Indag. Math. 10 (1948) 261–263. URL https://www.mit.edu/~kardar/research/seminars/Casimir/Casimir1948.pdf

[9] H. B. G. Casimir, D. Polder, The Influence of Retardation on the London-van der Waals Forces, Phys. Rev. 73 (1948) 360–372. doi:10.1103/PhysRev.73.360.

[10] V. M. Mostepanenko, N. N. Trunov, The Casimir effect and its applications, Soviet Physics Uspekhi 31 (11) (1988) 965. doi:10.1070/PU1988v031n11ABEH005641.

[11] A. W. Rodriguez, F. Capasso, S. G. Johnson, The Casimir effect in microstructured geometries, Nature Photonics 5 (4) (2011) 211–221. doi:10.1038/nphoton.2011.39.

[12] J. M. Pate, M. Goryachev, R. Y. Chiao, J. E. Sharping, M. E. Tobar, Casimir spring and dilution in macroscopic cavity optomechanics, Nature Physics 16 (11) (2020) 1117–1122. doi:10.1038/s41567-020-0975-9.

[13] Q.-D. Jiang, F. Wilczek, Chiral Casimir forces: Repulsive, enhanced, tunable, Phys. Rev. B 99 (2019) 125403. doi:10.1103/PhysRevB.99.125403.

[14] H. Gies, Strong laser fields as a probe for fundamental physics, The European Physical Journal D 55 (2) (2009) 311–317. doi:10.1140/epjd/e2009-00006-0.

[15] H. Gies, External fields as a probe for fundamental physics, Journal of Physics A: Mathematical and Theoretical 41 (16) (2008) 164039. doi:10.1088/1751-8113/41/16/164039.

[16] F. Karbstein, A. Blinne, H. Gies, M. Zepf, Boosting Quantum Vacuum Signatures by Coherent Harmonic Focusing, Phys. Rev. Lett. 123 (2019) 091802. doi:10.1103/PhysRevLett.123.091802.

[17] J. Schwinger, On Gauge Invariance and Vacuum Polarization, Phys. Rev. 82 (1951) 664–679. doi:10.1103/PhysRev.82.664.

[18] F. Sauter, Über das Verhalten eines Elektrons im homogenen elektrischen Feld nach der relativistischen Theorie Diracs, Zeitschrift für Physik 69 (11-12) (1931) 742–764. doi:10.1007/BF01339461.

[19] H. Euler, B. Kockel, Über die Streuung von Licht an Licht nach der Diracschen Theorie, Naturwissenschaften 23 (15) (1935) 246–247. doi:10.1007/BF01493898.

[20] W. Heisenberg, H. Euler, Folgerungen aus der Diracschen Theorie des Positrons, Zeitschrift für Physik 98 (11) (1936) 714–732. doi:10.1007/BF01343663.

[21] V. Weisskopf, The electrodynamics of the vacuum based on the quantum theory of the electron, Kong. Dan. Vid. Sel. Mat. Fys. Med. 14N6 (1936) 1–39.
URL http://www.neo-classical-physics.info/uploads/3/0/6/5/3065888/weisskopf_-_electrodynamics.pdf

[22] W. Dittrich, H. Gies, Probing the Quantum Vacuum: Perturbative Effective Action Approach in Quantum Electrodynamics and its Application, Vol. 166 of Springer Tracts in Modern Physics, Springer Berlin Heidelberg, 2000. doi:10.1007/3-540-45585-X.

[23] F. Karbstein, Probing Vacuum Polarization Effects with High-Intensity Lasers, Particles 3 (1) (2020) 39–61. doi:10.3390/particles3010005.

[24] C. Kohlfürst, M. Mitter, G. von Winckel, F. Hebenstreit, R. Alkofer, Optimizing the pulse shape for Schwinger pair production, Phys. Rev. D 88 (2013) 045028. doi:10.1103/PhysRevD.88.045028.

[25] A. D. Piazza, Strong-field QED in intense laser fields, Particle and Astroparticle Theory Seminar Max Planck Institute for Nuclear Physics, Heidelberg (2017).
URL https://www.mpi-hd.mpg.de/lin/seminar_theory/talks/Talk_diPiazza_100717.pdf

[26] G. V. Dunne, The Heisenberg–Euler Effective Action: 75 years on, International Journal of Modern Physics A 27 (15) (2012) 1260004. doi:10.1142/S0217751X12600044.

[27] P. W. Milonni, The Quantum Vacuum: An Introduction to Quantum Electrodynamics, Academic press, 1994. doi:10.1016/C2009-0-21295-5.

[28] Y. B. Zel'dovich, The cosmological constant and the theory of elementary particles, Soviet Physics Uspekhi 11 (3) (1968) 381. doi:10.1070/PU1968v011n03ABEH003927.

[29] U. Leonhardt, The case for a Casimir cosmology, Philosophical Transactions of the Royal Society A 378 (2177) (2020) 20190229. doi:10.1098/rsta.2019.0229.

[30] S. Weinberg, The cosmological constant problem, Rev. Mod. Phys. 61 (1989) 1–23. doi:10.1103/RevModPhys.61.1.

[31] N. Itzhaki, A comment on technical naturalness and the cosmological constant, Journal of High Energy Physics 8 (2006) 020. doi:10.1088/1126-6708/2006/08/020.

[32] C. P. Burgess, The Cosmological Constant Problem: Why it's hard to get Dark Energy from Micro-physics (2013). arXiv:1309.4133, doi:10.48550/arXiv.1309.4133.

[33] S. W. Hawking, Black hole explosions?, Nature 248 (5443) (1974) 30–31. doi:10.1038/248030a0.

[34] L. C. B. Crispino, A. Higuchi, G. E. A. Matsas, The Unruh effect and its applications, Rev. Mod. Phys. 80 (2008) 787–838. doi:10.1103/RevModPhys.80.787.

[35] E. T. Akhmedov, D. Singleton, On the physical meaning of the Unruh effect, JETP Letters 86 (9) (2008) 615–619. doi:10.1134/S0021364007210138.

[36] J. M. Ezquiaga, J. García-Bellido, V. Vennin, Massive Galaxy Clusters Like El Gordo Hint at Primordial Quantum Diffusion, Phys. Rev. Lett. 130 (2023) 121003. doi:10.1103/PhysRevLett.130.121003.

[37] T. Blum, P. A. Boyle, V. Gülpers, T. Izubuchi, L. Jin, C. Jung, A. Jüttner, C. Lehner, A. Portelli, J. T. Tsang, Calculation of the Hadronic Vacuum Polarization Contribution to the Muon Anomalous Magnetic Moment, Phys. Rev. Lett. 121 (2018) 022003. doi:10.1103/PhysRevLett.121.022003.

[38] M. Davier, A. Hoecker, B. Malaescu, Z. Zhang, A new evaluation of the hadronic vacuum polarisation contributions to the muon anomalous magnetic moment and to α (mz2), The European Physical Journal C 80 (3) (2020) 1–13. doi:10.1140/epjc/s10052-020-7792-2.

[39] M. Knecht, On some short-distance properties of the fourth-rank hadronic vacuum polarization tensor and the anomalous magnetic moment of the muon, Journal of High Energy Physics 2020 (8) (2020) 1–29. doi:10.1007/JHEP08(2020)056.

[40] T. Aoyama, N. Asmussen, M. Benayoun, J. Bijnens, T. Blum, M. Bruno, I. Caprini, C. Carloni Calame, M. Cè, G. Colangelo, et al., The anomalous magnetic moment of the muon in the Standard Model, Physics Reports 887 (2020) 1–166. doi:10.1016/j.physrep.2020.07.006.

[41] B. Abi, T. Albahri, S. Al-Kilani, D. Allspach, L. P. Alonzi, A. Anastasi, A. Anisenkov, F. Azfar, K. Badgley, S. Baeßler, et al., Measurement of the Positive Muon Anomalous Magnetic Moment to 0.46 ppm, Phys. Rev. Lett. 126 (2021) 141801. doi:10.1103/PhysRevLett.126.141801.

[42] S. Borsanyi, Z. Fodor, J. N. Guenther, C. Hoelbling, S. D. Katz, L. Lellouch, T. Lippert, K. Miura, L. Parato, K. K. Szabo, et al., Leading hadronic contribution to the muon magnetic moment from lattice QCD, Nature 593 (7857) (2021) 51–55. doi:10.1038/s41586-021-03418-1.

[43] A. I. Berdyugin, N. Xin, H. Gao, S. Slizovskiy, Z. Dong, S. Bhattacharjee, P. Kumaravadivel, S. Xu, L. A. Ponomarenko, M. Holwill, et al., Out-of-equilibrium criticalities in graphene superlattices, Science 375 (6579) (2022) 430–433. doi:10.1126/science.abi8627.

[44] H. Rottke, R. Y. Engel, D. Schick, J. O. Schunck, P. S. Miedema, M. C. Borchert, M. Kuhlmann, N. Ekanayake, S. Dziarzhytski, G. Brenner, et al., Probing electron and hole colocalization by resonant four-wave mixing spectroscopy in the extreme ultraviolet, Science Advances 8 (20) (2022) eabn5127. doi:10.1126/sciadv.abn5127.

[45] G. Breit, J. A. Wheeler, Collision of Two Light Quanta, Phys. Rev. 46 (1934) 1087–1091. doi:10.1103/PhysRev.46.1087.

[46] O. J. Pike, F. Mackenroth, E. G. Hill, S. J. Rose, A photon–photon collider in a vacuum hohlraum, Nature Photonics 8 (6) (2014) 434–436. doi:10.1038/nphoton.2014.95.

[47] A. Golub, S. Villalba-Chávez, H. Ruhl, C. Müller, Linear Breit-Wheeler pair production by high-energy bremsstrahlung photons colliding with an intense x-ray laser pulse, Phys. Rev. D 103 (2021) 016009. doi:10.1103/PhysRevD.103.016009.

[48] C. Kohlfürst, H. Gies, R. Alkofer, Effective Mass Signatures in Multiphoton Pair Production, Phys. Rev. Lett. 112 (2014) 050402. doi:10.1103/PhysRevLett.112.050402.

[49] M. Ruf, G. R. Mocken, C. Müller, K. Z. Hatsagortsyan, C. H. Keitel, Pair Production in Laser Fields Oscillating in Space and Time, Phys. Rev. Lett. 102 (2009) 080402. doi:10.1103/PhysRevLett.102.080402.

[50] C. Kohlfürst, R. Alkofer, On the effect of time-dependent inhomogeneous magnetic fields in electronpositron pair production, Physics Letters B 756 (2016) 371 – 375. doi:10.1016/j.physletb.2016.03.027.

[51] G. Bimonte, E. Calloni, G. Esposito, L. Milano, L. Rosa, Towards measuring variations of casimir energy by a superconducting cavity, Physical Review Letters 94 (18) (2005). doi:10.1103/physrevlett.94.180402.

[52] J. Cripe, N. Aggarwal, R. Lanza, A. Libson, R. Singh, P. Heu, D. Follman, G. D. Cole, N. Mavalvala, T. Corbitt, Measurement of quantum back action in the audio band at room temperature, Nature 568 (7752) (2019) 364–367. doi:10.1038/s41586-019-1051-4.

[53] H. Yu, L. McCuller, M. Tse, N. Kijbunchoo, L. Barsotti, N. Mavalvala, J. Betzwieser, C. D. Blair, S. E. Dwyer, A. Effler, et al., Quantum correlations between light and the kilogram-mass mirrors of LIGO, Nature 583 (7814) (2020) 43–47. doi:10.1038/s41586-020-2420-8.

[54] K. Y. Fong, H.-K. Li, R. Zhao, S. Yang, Y. Wang, X. Zhang, Phonon heat transfer across a vacuum through quantum fluctuations, Nature 576 (7786) (2019) 243–247. doi:10.1038/s41586-019-1800-4.

[55] J. Ahn, Z. Xu, J. Bang, P. Ju, X. Gao, T. Li, Ultrasensitive torque detection with an optically levitated nanorotor, Nature Nanotechnology 15 (2) (2020) 89–93. doi:10.1038/s41565-019-0605-9.

[56] M. G. Millis, G. J. Maclay, J. Hammer, R. Clark, M. George, Y. Kim, A. Kir, Study of Vacuum Energy Physics for Breakthrough Propulsion, Tech. rep., NASA (2004).
URL https://ntrs.nasa.gov/citations/20040171927

[57] T. C. N. Boekholt, S. F. Portegies Zwart, M. Valtonen, Gargantuan chaotic gravitational three-body systems and their irreversibility to the Planck length, Monthly Notices of the Royal Astronomical Society 493 (3) (2020) 3932–3937. doi:10.1093/mnras/staa452.

[58] I.-C. Benea-Chelmus, F. F. Settembrini, G. Scalari, J. Faist, Electric field correlation measurements on the electromagnetic vacuum state, Nature 568 (7751) (2019) 202–206. doi:10.1038/s41586-019-1083-9.

[59] J. C. Maxwell, A Dynamical Theory of the Electromagnetic Field, Philosophical Transactions of the Royal Society of London 155 (1865) 459–512. doi:10.1098/rstl.1865.0008.

[60] R. Karplus, M. Neuman, Non-Linear Interactions between Electromagnetic Fields, Phys. Rev. 80 (1950) 380–385. doi:10.1103/PhysRev.80.380.

[61] R. Karplus, M. Neuman, The Scattering of Light by Light, Phys. Rev. 83 (1951) 776–784. doi:10.1103/PhysRev.83.776.

[62] J. Mckenna, P. M. Platzman, Nonlinear Interaction of Light in a Vacuum, Phys. Rev. 129 (1963) 2354–2360. doi:10.1103/PhysRev.129.2354.

[63] F. Moulin, D. Bernard, F. Amiranoff, Photon-photon elastic scattering in the visible domain, Zeitschrift für Physik C: Particles and Fields 72 (4) (1996) 607. doi:10.1007/s002880050282.

[64] F. Moulin, D. Bernard, Four-wave interaction in gas and vacuum: definition of a third-order nonlinear effective susceptibility in vacuum: $\chi^{(3)}_{\text{vacuum}}$, Optics Communications 164 (1) (1999) 137–144. doi:10.1016/S0030-4018(99)00169-8.

[65] D. Bernard, F. Moulin, F. Amiranoff, A. Braun, J. P. Chambaret, G. Darpentigny, G. Grillon, S. Ranc, F. Perrone, Search for stimulated photon-photon scattering in vacuum, The European Physical Journal D - Atomic, Molecular, Optical and Plasma Physics 10 (1) (2000) 141–145. doi:10.1007/s100530050535.

[66] E. Lundström, G. Brodin, J. Lundin, M. Marklund, R. Bingham, J. Collier, J. T. Mendonça, P. Norreys, Using High-Power Lasers for Detection of Elastic Photon-Photon Scattering, Phys. Rev. Lett. 96 (2006) 083602. doi:10.1103/PhysRevLett.96.083602.

[67] J. Lundin, M. Marklund, E. Lundström, G. Brodin, J. Collier, R. Bingham, J. T. Mendonça, P. Norreys, Analysis of four-wave mixing of high-power lasers for the detection of elastic photon-photon scattering, Phys. Rev. A 74 (2006) 043821. doi:10.1103/PhysRevA.74.043821.

[68] D. Tommasini, A. Ferrando, H. Michinel, M. Seco, Precision tests of QED and non-standard models by searching photon-photon scattering in vacuum with high power lasers, JHEP 11 (2009) 043. doi:10.1088/1126-6708/2009/11/043.

[69] G. Y. Kryuchkyan, K. Z. Hatsagortsyan, Bragg Scattering of Light in Vacuum Structured by Strong Periodic Fields, Phys. Rev. Lett. 107 (2011) 053604. doi:10.1103/PhysRevLett.107.053604.

[70] B. King, C. H. Keitel, Photon–photon scattering in collisions of intense laser pulses, New Journal of Physics 14 (10) (2012) 103002. doi:10.1088/1367-2630/14/10/103002.

[71] V. Dinu, T. Heinzl, A. Ilderton, M. Marklund, G. Torgrimsson, Photon polarization in light-by-light scattering: Finite size effects, Phys. Rev. D 90 (2014) 045025. doi:10.1103/PhysRevD.90.045025.

[72] H. Gies, F. Karbstein, C. Kohlfürst, N. Seegert, Photon-photon scattering at the high-intensity frontier, Phys. Rev. D 97 (2018) 076002. doi:10.1103/PhysRevD.97.076002.

[73] B. King, H. Hu, B. Shen, Three-pulse photon-photon scattering, Phys. Rev. A 98 (2018) 023817. doi:10.1103/PhysRevA.98.023817.

[74] J. Toll, The dispersion relation for light and its application to problems involving electron pairs, Ph.D. thesis, Princeton University, (unpublished) (1952).

[75] R. Baier, P. Breitenlohner, The vacuum refraction index in the presence of external fields, Il Nuovo Cimento B Series 10 47 (1) (1967) 117–120. doi:10.1007/BF02712312.

[76] E. Brezin, C. Itzykson, Polarization Phenomena in Vacuum Nonlinear Electrodynamics, Phys. Rev. D 3 (1971) 618–621. doi:10.1103/PhysRevD.3.618.

[77] J. S. Heyl, L. Hernquist, Birefringence and dichroism of the QED vacuum, Journal of Physics A: Mathematical and General 30 (18) (1997) 6485. doi:10.1088/0305-4470/30/18/022.

[78] A. N. Luiten, J. C. Petersen, Detection of vacuum birefringence using intense laser pulses, Physics Letters A 330 (6) (2004) 429 – 434. doi:10.1016/j.physleta.2004.08.020.

[79] T. Heinzl, B. Liesfeld, K.-U. Amthor, H. Schwoerer, R. Sauerbrey, A. Wipf, On the observation of vacuum birefringence, Optics Communications 267 (2) (2006) 318 – 321. doi:10.1016/j.optcom.2006.06.053.

[80] A. Di Piazza, K. Z. Hatsagortsyan, C. H. Keitel, Light Diffraction by a Strong Standing Electromagnetic Wave, Phys. Rev. Lett. 97 (2006) 083603. doi:10.1103/PhysRevLett.97.083603.

[81] B. J. King, Vacuum polarisation effects in intense laser fields, Ph.D. thesis, Ruprecht-Karls-Universität Heidelberg (2010).
URL https://archiv.ub.uni-heidelberg.de/volltextserver/10846

[82] V. Dinu, T. Heinzl, A. Ilderton, M. Marklund, G. Torgrimsson, Vacuum refractive indices and helicity flip in strong-field QED, Phys. Rev. D 89 (2014) 125003. doi:10.1103/PhysRevD.89.125003.

[83] F. Karbstein, H. Gies, M. Reuter, M. Zepf, Vacuum birefringence in strong inhomogeneous electromagnetic fields, Phys. Rev. D 92 (2015) 071301. doi:10.1103/PhysRevD.92.071301.

[84] H.-P. Schlenvoigt, T. Heinzl, U. Schramm, T. E. Cowan, R. Sauerbrey, Detecting vacuum birefringence with x-ray free electron lasers and high-power optical lasers: a feasibility study, Physica Scripta 91 (2) (2016) 023010. doi:10.1088/0031-8949/91/2/023010.

[85] F. Karbstein, C. Sundqvist, Probing vacuum birefringence using x-ray free electron and optical high-intensity lasers, Physical Review D 94 (1) (2016). doi:10.1103/physrevd.94.013004.

[86] B. King, N. Elkina, Vacuum birefringence in high-energy laser-electron collisions, Phys. Rev. A 94 (2016) 062102. doi:10.1103/PhysRevA.94.062102.

[87] S. Bragin, S. Meuren, C. H. Keitel, A. Di Piazza, High-Energy Vacuum Birefringence and Dichroism in an Ultrastrong Laser Field, Phys. Rev. Lett. 119 (2017) 250403. doi:10.1103/PhysRevLett.119.250403.

[88] F. Karbstein, Vacuum birefringence in the head-on collision of x-ray free-electron laser and optical high-intensity laser pulses, Phys. Rev. D 98 (2018) 056010. doi:10.1103/PhysRevD.98.056010.

[89] S. Ataman, Vacuum birefringence detection in all-optical scenarios, Phys. Rev. A 97 (2018) 063811. doi:10.1103/PhysRevA.97.063811.

[90] F. Karbstein, E. A. Mosman, X-ray photon scattering at a focused high-intensity laser pulse, Phys. Rev. D 100 (2019) 033002. doi:10.1103/PhysRevD.100.033002.

[91] E. A. Mosman, F. Karbstein, Vacuum birefringence and diffraction at an x-ray free-electron laser: From analytical estimates to optimal parameters, Phys. Rev. D 104 (2021) 013006. doi:10.1103/PhysRevD.104.013006.

[92] F. Karbstein, C. Sundqvist, K. S. Schulze, I. Uschmann, H. Gies, G. G. Paulus, Vacuum birefringence at x-ray free-electron lasers, New Journal of Physics 23 (9) (2021) 095001. doi:10.1088/1367-2630/ac1df4.

[93] D. Tommasini, H. Michinel, Light by light diffraction in vacuum, Phys. Rev. A 82 (2010) 011803. doi:10.1103/PhysRevA.82.011803.

[94] B. King, A. Di Piazza, C. H. Keitel, A matterless double slit, Nature Photonics 4 (2) (2010) 92–94. doi:10.1038/nphoton.2009.261.

[95] B. King, A. Di Piazza, C. H. Keitel, Double-slit vacuum polarization effects in ultraintense laser fields, Phys. Rev. A 82 (2010) 032114. doi:10.1103/PhysRevA.82.032114.

[96] Y. Monden, R. Kodama, Enhancement of Laser Interaction with Vacuum for a Large Angular Aperture, Phys. Rev. Lett. 107 (2011) 073602. doi:10.1103/PhysRevLett.107.073602.

[97] F. Karbstein, R. R. Q. P. T. Oude Weernink, X-ray vacuum diffraction at finite spatiotemporal offset, Phys. Rev. D 104 (2021) 076015. doi:10.1103/PhysRevD.104.076015.

[98] H. Gies, F. Karbstein, N. Seegert, Quantum reflection as a new signature of quantum vacuum nonlinearity, New Journal of Physics 15 (8) (2013) 083002. doi:10.1088/1367-2630/15/8/083002.

[99] H. Gies, F. Karbstein, N. Seegert, Quantum reflection of photons off spatio-temporal electromagnetic field inhomogeneities, New Journal of Physics 17 (4) (2015) 043060. doi:10.1088/1367-2630/17/4/043060.

[100] V. Yakovlev, Incoherent electromagnetic wave scattering in a Coulomb field, Sov. Phys. JETP 24 (1967) 411. URL http://www.jetp.ras.ru/cgi-bin/dn/e_024_02_0411.pdf

[101] A. Di Piazza, K. Z. Hatsagortsyan, C. H. Keitel, Nonperturbative Vacuum-Polarization Effects in Proton-Laser Collisions, Phys. Rev. Lett. 100 (2008) 010403. doi:10.1103/PhysRevLett.100.010403.

[102] H. Gies, F. Karbstein, R. Shaisultanov, Laser photon merging in an electromagnetic field inhomogeneity, Phys. Rev. D 90 (2014) 033007. doi:10.1103/PhysRevD.90.033007.

[103] H. Gies, F. Karbstein, N. Seegert, Photon merging and splitting in electromagnetic field inhomogeneities, Phys. Rev. D 93 (2016) 085034. doi:10.1103/PhysRevD.93.085034.

[104] P. Bhartia, S. Valluri, Non-linear scattering of light in the limit of ultra-strong fields, Canadian Journal of Physics 56 (8) (1978) 1122–1132. doi:10.1139/p78-147.

[105] S. R. Valluri, P. Bhartia, An analytical proof for the generation of higher harmonics due to the interaction of plane electromagnetic waves, Canadian Journal of Physics 58 (1) (1980) 116–122. doi:10.1139/p80-019.

[106] Z. Bialynicka-Birula, Nonlinear phenomena in the propagation of electromagnetic waves in the magnetized vacuum, Physica D: Nonlinear Phenomena 2 (3) (1981) 513–524. doi:10.1016/0167-2789(81)90025-7.

[107] A. E. Kaplan, Y. J. Ding, Field-gradient-induced second-harmonic generation in magnetized vacuum, Phys. Rev. A 62 (2000) 043805. doi:10.1103/PhysRevA.62.043805.

[108] A. Di Piazza, K. Z. Hatsagortsyan, C. H. Keitel, Harmonic generation from laser-driven vacuum, Phys. Rev. D 72 (2005) 085005. doi:10.1103/PhysRevD.72.085005.

[109] A. Fedotov, N. Narozhny, Generation of harmonics by a focused laser beam in the vacuum, Physics Letters A 362 (1) (2007) 1–5. doi:10.1016/j.physleta.2006.09.085.

[110] N. B. Narozhny, A. M. Fedotov, Third-harmonic generation in a vacuum at the focus of a high-intensity laser beam, Laser Physics 17 (4) (2007) 350–357. doi:10.1134/S1054660X0704010X.

[111] B. King, P. Böhl, H. Ruhl, Interaction of photons traversing a slowly varying electromagnetic background, Phys. Rev. D 90 (2014) 065018. doi:10.1103/PhysRevD.90.065018.

[112] P. Böhl, B. King, H. Ruhl, Vacuum high-harmonic generation in the shock regime, Phys. Rev. A 92 (2015) 032115. doi:10.1103/PhysRevA.92.032115.

[113] P. A. Böhl, Vacuum harmonic generation in slowly varying electromagnetic backgrounds, Ph.D. thesis, LMU (2016). URL https://edoc.ub.uni-muenchen.de/19887

[114] H. Kadlecová, G. Korn, S. V. Bulanov, Electromagnetic shocks in the quantum vacuum, Physical Review D 99 (3) (2019). doi:10.1103/physrevd.99.036002.

[115] P. V. Sasorov, F. Pegoraro, T. Z. Esirkepov, S. V. Bulanov, Generation of high order harmonics in Heisenberg–Euler electrodynamics, New Journal of Physics 23 (10) (2021) 105003. doi:10.1088/1367-2630/ac28cb.

[116] S. L. Adler, J. N. Bahcall, C. G. Callan, M. N. Rosenbluth, Photon Splitting in a Strong Magnetic Field, Phys. Rev. Lett. 25 (1970) 1061–1065. doi:10.1103/PhysRevLett.25.1061.

[117] Z. Bialynicka-Birula, I. Bialynicki-Birula, Nonlinear Effects in Quantum Electrodynamics. Photon Propagation and Photon Splitting in an External Field, Phys. Rev. D 2 (1970) 2341–2345. doi:10.1103/PhysRevD.2.2341.

[118] S. L. Adler, Photon splitting and photon dispersion in a strong magnetic field, Annals of Physics 67 (2) (1971) 599 – 647. doi:10.1016/0003-4916(71)90154-0.

[119] V. O. Papanjan, V. I. Ritus, Vacuum polarization and photon splitting in an intense field, Tech. rep., Akad. Nauk Moscow. Fiz. Inst. P. N. Lebedev, Moscow (1971).
URL http://cds.cern.ch/record/1056277

[120] R. J. Stoneham, Phonon splitting in the magnetised vacuum, Journal of Physics A: Mathematical and General 12 (11) (1979) 2187–2203. doi:10.1088/0305-4470/12/11/028.

[121] V. N. Baier, A. I. Milstein, R. Z. Shaisultanov, Photon Splitting in a Very Strong Magnetic Field, Phys. Rev. Lett. 77 (1996) 1691–1694. doi:10.1103/PhysRevLett.77.1691.

[122] S. L. Adler, C. Schubert, Photon Splitting in a Strong Magnetic Field: Recalculation and Comparison with Previous Calculations, Phys. Rev. Lett. 77 (1996) 1695–1698. doi:10.1103/PhysRevLett.77.1695.

[123] A. Di Piazza, A. I. Milstein, C. H. Keitel, Photon splitting in a laser field, Phys. Rev. A 76 (2007) 032103. doi:10.1103/PhysRevA.76.032103.

[124] C. N. Danson, C. Haefner, J. Bromage, T. Butcher, J.-C. F. Chanteloup, E. A. Chowdhury, A. Galvanauskas, L. A. Gizzi, J. Hein, D. I. Hillier, et al., Petawatt and exawatt class lasers worldwide, High Power Laser Science and Engineering 7 (2019) e54. doi:10.1017/hpl.2019.36.

[125] M. Scholz, FEL Performance Achieved at European XFEL, in: Proc. 9th International Particle Accelerator Conference (IPAC'18), Vancouver, BC, Canada, April 29-May 4, 2018, no. 9 in International Particle Accelerator Conference, JACoW Publishing, Geneva, Switzerland, 2018, pp. 29–33. doi:10.18429/JACoW-IPAC2018-MOZGBD2.

[126] M. Marklund, J. Lundin, Quantum vacuum experiments using high intensity lasers, The European Physical Journal D 55 (2) (2009) 319. doi:10.1140/epjd/e2009-00169-6.

[127] A. Di Piazza, C. Müller, K. Z. Hatsagortsyan, C. H. Keitel, Extremely high-intensity laser interactions with fundamental quantum systems, Rev. Mod. Phys. 84 (2012) 1177–1228. doi:10.1103/RevModPhys.84.1177.

[128] B. King, T. Heinzl, Measuring vacuum polarization with high-power lasers, High Power Laser Science and Engineering 4 (2016) e5. doi:10.1017/hpl.2016.1.

[129] W. Dittrich, M. Reuter, Effective Lagrangians in quantum electrodynamics, Springer, 1985. doi:10.1007/3-540-15182-6.

[130] G. V. Dunne, Heisenberg–Euler Effective Lagrangians: Basics and Extensions, in: From Fields to Strings: Circumnavigating Theoretical Physics: Ian Kogan Memorial Collection (In 3 Volumes), World Scientific, 2005, pp. 445–522. doi:10.1142/9789812775344_0014.

[131] G. V. Dunne, New strong-field QED effects at extreme light infrastructure, The European Physical Journal D 55 (2) (2009) 327. doi:10.1140/epjd/e2009-00022-0.

[132] T. Heinzl, A. Ilderton, Exploring high-intensity QED at ELI, The European Physical Journal D 55 (2) (2009) 359–364. doi:10.1140/epjd/e2009-00113-x.

[133] F. Karbstein, The quantum vacuum in electromagnetic fields: From the Heisenberg-Euler effective action to vacuum birefringence, in: Quantum Field Theory at the Limits: from Strong Fields to Heavy Quarks, Proceedings of the Helmholtz International Summer School 2016, Verlag Deutsches Elektronen-Synchrotron, Hamburg, 2017, pp. 44–57. doi:10.3204/DESY-PROC-2016-04/Karbstein.

[134] T. Inada, T. Yamazaki, T. Yamaji, Y. Seino, X. Fan, S. Kamioka, T. Namba, S. Asai, Probing Physics in Vacuum Using an X-ray Free-Electron Laser, a High-Power Laser, and a High-Field Magnet, Applied Sciences 7 (7) (2017). doi:10.3390/app7070671.

[135] M. Marklund, P. K. Shukla, Nonlinear collective effects in photon-photon and photon-plasma interactions, Rev. Mod. Phys. 78 (2006) 591–640. doi:10.1103/RevModPhys.78.591.

[136] R. Battesti, J. Beard, S. Böser, N. Bruyant, D. Budker, S. A. Crooker, E. J. Daw, V. V. Flambaum, T. Inada, I. G. Irastorza, et al., High magnetic fields for fundamental physics, Physics Reports 765-766 (2018) 1–39. doi:10.1016/j.physrep.2018.07.005.

[137] G. V. Dunne, T. M. Hall, Borel summation of the derivative expansion and effective actions, Phys. Rev. D 60 (1999) 065002. doi:10.1103/PhysRevD.60.065002.

[138] H. Gies, L. Roessler, Vacuum polarization tensor in inhomogeneous magnetic fields, Phys. Rev. D 84 (2011) 065035. doi:10.1103/PhysRevD.84.065035.

[139] E. Tiesinga, P. J. Mohr, D. B. Newell, B. N. Taylor, CODATA recommended values of the fundamental physical constants: 2018, Rev. Mod. Phys. 93 (2021) 025010. doi:10.1103/RevModPhys.93.025010.

[140] T. Heinzl, Strong-field QED and high-power lasers, International Journal of Modern Physics A 27 (15) (2012) 1260010. doi:10.1142/S0217751X1260010X.

[141] L. Klar, Quantum vacuum nonlinearities in the all-optical regime, Ph.D. thesis, Friedrich-Schiller-Universität Jena (2022).
URL https://www.db-thueringen.de/receive/dbt_mods_00055002

[142] M. Peskin, D. V. Schroeder, An Introduction To Quantum Field Theory, 1st Edition, CRC Press, 1995. doi:10.1201/9780429503559.

[143] A. Blinne, H. Gies, F. Karbstein, C. Kohlfürst, M. Zepf, All-optical signatures of quantum vacuum nonlinearities in generic laser fields, Phys. Rev. D 99 (2019) 016006. doi:10.1103/PhysRevD.99.016006.

[144] A. P. Domenech, H. Ruhl, An implicit ODE-based numerical solver for the simulation of the Heisenberg-Euler equations in 3+1 dimensions (2017). arXiv:1607.00253, doi:10.48550/arXiv.1607.00253.

[145] A. Pons Domenech, Simulation of quantum vacuum in higher dimensions, Ph.D. thesis, LMU (2018).
URL https://edoc.ub.uni-muenchen.de/21885

[146] A. Lindner, B. Ölmez, H. Ruhl, Numerical Simulations of the Nonlinear Quantum Vacuum in the Heisenberg-Euler Weak-Field Expansion (2021). arXiv:2109.08121, doi:10.48550/arXiv.2202.09680.

[147] A. Lindner, B. Ölmez, H. Ruhl, Numerical simulations of the nonlinear quantum vacuum in the Heisenberg-Euler weak-field expansion, Journal of Computational Physics: X 17 (2023) 100124. doi:10.1016/j.jcpx.2023.100124.

[148] A. Lindner, B. Ölmez, H. Ruhl, HEWES: Heisenberg–Euler weak-field expansion simulator, Software Impacts 15 (2023) 100481. doi:https://doi.org/10.1016/j.simpa.2023.100481.

[149] C. Shannon, Communication in the presence of noise, Proceedings of the IEEE 86 (2) (1998) 447–457. doi:10.1109/JPROC.1998.659497.

[150] A. Lindner, HEWES project extra and supplementary material, Mendeley Data (2023). doi:10.17632/f9wntyw39x.3.

[151] Wolfram Research, Inc., Mathematica, Version 13.2, Champaign, IL (2022).
URL https://www.wolfram.com/mathematica

[152] A. Lindner, HEWES Benchmarking Supplementary Analyses, Notebook Archive (2022).
URL https://notebookarchive.org/2022-08-eb2cjxb

[153] A. Köhler, Der C/C++-Projektbegleiter: C/C++ Projekte planen, dokumentieren, bauen und testen, dpunkt.verlag GmbH, Heidelberg, 2007.

[154] F. Karbstein, R. Shaisultanov, Stimulated photon emission from the vacuum, Phys. Rev. D 91 (2015) 113002. doi:10.1103/PhysRevD.91.113002.

[155] K. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Transactions on Antennas and Propagation 14 (3) (1966) 302–307. doi:10.1109/TAP.1966.1138693.

[156] H. Gies, F. Karbstein, C. Kohlfürst, All-optical signatures of strong-field QED in the vacuum emission picture, Phys. Rev. D 97 (2018) 036022. doi:10.1103/PhysRevD.97.036022.

[157] A. Blinne, S. Kuschel, S. Tietze, M. Zepf, Efficient retrieval of phase information from real-valued electromagnetic field data, Journal of Computational Physics: X 1 (2019) 100019. doi:10.1016/j.jcpx.2019.100019.

[158] F. Karbstein, Vacuum Birefringence as a Vacuum Emission Process (2015). doi:10.48550/ARXIV.1510.03178.

[159] L. Klar, Detectable Optical Signatures of QED Vacuum Nonlinearities Using High-Intensity Laser Fields, Particles 3 (1) (2020) 223–233. doi:10.3390/particles3010018.

[160] F. Karbstein, E. A. Mosman, Enhancing quantum vacuum signatures with tailored laser beams, Phys. Rev. D 101 (2020) 113002. doi:10.1103/PhysRevD.101.113002.

[161] F. Karbstein, Vacuum Birefringence at the Gamma Factory, Annalen der Physik 534 (3) (2022) 2100137. doi:10.1002/andp.202100137.

[162] H. Gies, F. Karbstein, L. Klar, Quantum vacuum signatures in multicolor laser pulse collisions, Phys. Rev. D 103 (2021) 076009. doi:10.1103/PhysRevD.103.076009.

[163] A. Blinne, H. Gies, F. Karbstein, C. Kohlfürst, M. Zepf, Photon-Photon Scattering at the High-Intensity Frontier: Paraxial Beams, Journal of Physics: Conference Series 1206 (2019) 012016. doi:10.1088/1742-6596/1206/1/012016.

[164] A. Blinne, H. Gies, F. Karbstein, C. Kohlfürst, M. Zepf, The Vacuum Emission Picture Beyond Paraxial Approximation, Journal of Physics: Conference Series 1206 (2019) 012017. doi:10.1088/1742-6596/1206/1/012017.

[165] T. Grismayer, R. Torres, P. Carneiro, F. Cruz, R. A. Fonseca, L. O. Silva, Quantum Electrodynamics vacuum polarization solver, New Journal of Physics 23 (9) (2021) 095005. doi:10.1088/1367-2630/ac2004.

[166] A. Taflove, S. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, 3rd Edition, Artech House, 2005.

[167] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, C. S. Woodward, SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Trans. Math. Softw. 31 (3) (2005) 363396. doi:10.1145/1089014.1089020.

[168] D. J. Gardner, D. R. Reynolds, C. S. Woodward, C. J. Balos, Enabling New Flexibility in the SUNDIALS Suite of Nonlinear and Differential/Algebraic Equation Solvers, ACM Trans. Math. Softw. 48 (3) (2022). doi:10.1145/3539801.

[169] A. C. Hindmarsh, R. Serban, C. J. Balos, D. J. Gardner, D. R. Reynolds, C. S. Woodward, User Documentation for CVODE, v6.5.1 (2023).

[170] F. R. Moulton, Discussions: New Methods in Exterior Ballistics, The American Mathematical Monthly 35 (5) (1928) 246–250. doi:10.2307/2299587.

[171] E. Hairer, S. P. Nørsett, G. Wanner, Solving Ordinary Differential Equations. I, Nonstiff Problems, Springer Series in Computational Mathematics, Springer Berlin, Heidelberg, 1993. doi:10.1007/978-3-540-78862-1.

[172] L. Clarke, I. Glendinning, R. Hempel, The MPI Message Passing Interface Standard, in: K. M. Decker, R. M. Rehmann (Eds.), Programming Environments for Massively Parallel Distributed Systems, Birkhäuser Basel, Basel, 1994, pp. 213–218. doi:10.1007/978-3-0348-8534-8_21.

[173] T. G. Mattson, Y. H. He, A. E. Koniges, The OpenMP Common Core: Making OpenMP Simple Again, MIT Press, 2019.
URL http://ompcore.com

[174] B. Stroustrup, Thriving in a crowded and changing world: C++ 20062020, Proc. ACM Program. Lang. 4 (HOPL) (2020). doi:10.1145/3386320.

[175] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, et al., Jupyter Notebooks – a publishing format for reproducible computational workflows, in: F. Loizides, B. Schmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas, IOS Press, 2016, pp. 87 – 90. doi:10.3233/978-1-61499-649-1-87.

[176] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[177] J. Ahrens, B. Geveci, C. Law, ParaView: An End-User Tool for Large-Data Visualization, in: C. D. Hansen, C. R. Johnson (Eds.), Visualization Handbook, Butterworth-Heinemann, Burlington, 2005, pp. 717–731. doi:10.1016/B978-012387582-2/50038-1.

[178] K. Martin, B. Hoffman, An Open Source Approach to Developing Software in a Small Organization, IEEE Software 24 (1) (2007) 46–53. doi:10.1109/MS.2007.5.

[179] A. Clyburne-Sherin, X. Fei, S. A. Green, Computational reproducibility via containers in social psychology, Meta-Psychology 3 (2019). doi:10.15626/MP.2018.892.

[180] A. Lindner, HEWES: Heisenberg-Euler Weak-Field Expansion Simulator [Source Code], Code Ocean (2023). doi:10.24433/CO.5672141.v1.

[181] W.-y. Tsai, T. Erber, Propagation of photons in homogeneous magnetic fields: Index of refraction, Phys. Rev. D 12 (1975) 1132–1137. doi:10.1103/PhysRevD.12.1132.

[182] G. Zavattini, U. Gastaldi, R. Pengo, G. Ruoso, F. D. Valle, E. Milotti, Measuring the magnetic birefringence of vacuum: the PVLAS experiment, International Journal of Modern Physics A 27 (15) (2012) 1260017. doi:10.1142/S0217751X12600172.

[183] A. Cadène, P. Berceau, M. Fouché, R. Battesti, C. Rizzo, Vacuum magnetic linear birefringence using pulsed fields: status of the BMV experiment, The European Physical Journal D 68 (1) (2014) 16. doi:10.1140/epjd/e2013-40725-9.

[184] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, T. B. Schardl, There's plenty of room at the Top: What will drive computer performance after Moore's law?, Science 368 (6495) (2020) eaam9744. doi:10.1126/science.aam9744.

[185] J. Dongarra, P. Luszczek, A. Petitet, The LINPACK Benchmark: past, present and future, Concurrency and Computation: Practice and Experience 15 (9) (2003) 803–820. doi:10.1002/cpe.728.

[186] J. Dongarra, P. Luszczek, Encyclopedia of Parallel Computing, Springer, Boston, MA, 2011, Ch. TOP500, p. 20552057. doi:10.1007/978-0-387-09766-4_157.

[187] G. Hager, G. Wellein, Introduction to High Performance Computing for Scientists and Engineers, Taylor & Francis Group, Baton Rouge, 2010.

[188] J. Samuel, M. Brennan-Tonetta, Y. Samuel, P. Subedi, J. Smith, Strategies for Democratization of Supercomputing: Availability, Accessibility and Usability of High Performance Computing for Education and Practice of Big Data Analytics (2021). doi:10.48550/ARXIV.2104.09091.

[189] A. Gupta, L. V. Kale, F. Gioachin, V. March, C. H. Suen, B.-S. Lee, P. Faraboschi, R. Kaufmann, D. Milojicic, The Who, What, Why, and How of High Performance Computing in the Cloud, in: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Vol. 1, 2013, pp. 306–314. doi:10.1109/CloudCom.2013.47.

[190] P. Böhl, J. Coles, G. Mathias, C. Guillen, N. Patel, J. Weidendorfer, HPC Code Optimisation Workshop, Leibniz-Rechenzentrum (LRZ) (2022).
URL https://doku.lrz.de/display/PUBLIC/PRACE+Course%3A+HPC+Code+Optimisation+Workshop+2022

[191] G. Hager, G. Wellein, Node-Level Performance Engineering, Leibniz-Rechenzentrum (LRZ) (2020).
URL https://moodle.nhr.fau.de/course/view.php?id=4

[192] N. Nethercote, J. Seward, Valgrind: A Program Supervision Framework, Electronic Notes in Theoretical Computer Science 89 (2) (2003) 44–66, RV '2003, Run-time Verification (Satellite Workshop of CAV '03). doi:https://doi.org/10.1016/S1571-0661(04)81042-9.

[193] J. Treibig, G. Hager, G. Wellein, LIKWID: A Lightweight Performance-Oriented Tool Suite for X86 Multicore Environments, in: Proceedings of the 2010 39th International Conference on Parallel Processing Workshops, ICPPW '10, IEEE Computer Society, USA, 2010, p. 207216. doi:10.1109/ICPPW.2010.38.

[194] M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, B. Mohr, The Scalasca Performance Toolset Architecture, Concurrency and Computation: Practice and Experience 22 (6) (2010) 702719.
URL https://apps.fz-juelich.de/jsc-pubsystem/aigaion/attachments/sthec08-special.pdf-c565f219e0f9e1814e48c543dfe71039.pdf

[195] A. Knüpfer, C. Rössel, D. a. Mey, S. ff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, et al., Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir, in: Tools for High Performance Computing 2011, Springer Berlin Heidelberg, 2012, pp. 79–91. doi:10.1007/978-3-642-31476-6_7.

[196] Intel® Corporation, Intel® oneAPI HPC Toolkit (2023).
URL https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit.html

[197] B. Wylie, C. Feld, P. Saviankou, Parallel Performance Analysis using Scalasca, UK National Supercomputer Center (EPCC) (2021).
URL https://github.com/EPCCed/archer2-scalasca-2021-07-27

[198] D. Henty, ARCHER2 Advanced MPI course, EPCC (2021).
URL https://github.com/EPCCed/archer2-AMPP-2021-07-14

[199] D. Tullsen, Pipeline Hazards (2005).
URL https://cseweb.ucsd.edu//classes/wi05/cse240a/pipe2.pdf

[200] C. Blaas-Schenner, D. Fischak, G. Hager, R. Rabenseifner, Introduction to Hybrid Programming in HPC, Vienna Scientific Cluster (VSC) (2021).
URL https://moodle.nhr.fau.de/course/view.php?id=27

[201] M. Bull, Advanced OpenMP, UK National Supercomputer Center (EPCC) (2021).
URL https://github.com/EPCCed/archer2-advanced-OpenMP/tree/2021-10-05

[202] MPI Forum, MPI: A Message-Passing Interface Standard Version 4.0 (2021).
URL https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf

[203] L. Einkemmer, C. Blaas-Schenner, Shared memory parallelization with OpenMP, Vienna Scientific Cluster (VSC) (2021).
URL https://vsc.ac.at//training/materials/openmp

[204] C. Blaas-Schenner, R. Rabenseifner, Parallelization with MPI, Vienna Scientific Cluster (VSC) (2021).
URL https://vsc.ac.at//training/materials/mpi

[205] G. M. Amdahl, Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, in: Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring), Association for Computing Machinery, New York, NY, USA, 1967, p. 483485. doi:10.1145/1465482.1465560.

[206] J. L. Gustafson, Reevaluating Amdahl's Law, Commun. ACM 31 (5) (1988) 532533. doi:10.1145/42411.42415.

[207] D. Henty, Efficient Parallel IO on ARCHER2, UK National Supercomputer Center (EPCC) (2022).
URL https://github.com/EPCCed/archer2-parallelIO-2022-08-23

[208] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, K. Riley, 24/7 Characterization of petascale I/O workloads, in: 2009 IEEE International Conference on Cluster Computing and Workshops, 2009, pp. 1–10. doi:10.1109/CLUSTR.2009.5289150.

[209] K. Atkinson, W. Han, D. E. Stewart, Numerical solution of ordinary differential equations, John Wiley & Sons, 2009. doi:10.1002/9781118164495.

[210] E. Süli, Numerical Solution of Ordinary Differential Equations, Lecture notes, University of Oxford (2022). URL https://people.maths.ox.ac.uk/suli/nsodes.pdf

[211] J. D. Lambert, Numerical Methods for Ordinary Differential Systems: The Intitial Value Problem, John Wiley & Sons, 1991.

[212] F. Bashforth, J. C. Adams, An Attempt to Test the Theories of Capillary Action by Comparing the Theoretical and Measured Forms of Drops of Fluid. With an Explanation of the Method of Integration Employed in Constucting the Tables which Give the Theoretical Forms of Such Drops, Cambridge University Press, Cambridge, 1883. URL https://archive.org/details/attempttest00bashrich

[213] P. N. Brown, G. D. Byrne, A. C. Hindmarsh, VODE: A Variable-Coefficient ODE Solver, SIAM Journal on Scientific and Statistical Computing 10 (5) (1989) 1038–1051. doi:10.1137/0910062.

[214] S. D. Cohen, A. C. Hindmarsh, P. F. Dubois, CVODE, A Stiff/Nonstiff ODE Solver in C, Computer in Physics 10 (2) (1996) 138–143. doi:10.1063/1.4822377.

[215] D. G. Anderson, Iterative procedures for nonlinear integral equations, J. ACM 12 (4) (1965) 547560. doi:10.1145/321296.321305.

[216] R. Rasche, A Numerical ODE-based Solver for the Integration of the Heisenberg-Euler Equations on an Adaptive Grid, Master thesis, LMU Munich, (unpublished) (2022).

[217] B. Oelmez, High Frequency Radiation Fields in Slowly Varying Strong Field Electromagnetic Backgrounds, Master thesis, LMU Munich, (unpublished) (2020).

[218] I. Bialynicki-Birula, P. Górnicki, J. Rafelski, Phase-space structure of the dirac vacuum, Phys. Rev. D 44 (1991) 1825–1835. doi:10.1103/PhysRevD.44.1825.

[219] D. Vasak, M. Gyulassy, H.-T. Elze, Quantum transport theory for abelian plasmas, Annals of Physics 173 (2) (1987) 462 – 492. doi:10.1016/0003-4916(87)90169-2.

[220] P. Zhuang, U. Heinz, Relativistic Quantum Transport Theory for Electrodynamics, Annals of Physics 245 (2) (1996) 311–338. doi:https://doi.org/10.1006/aphy.1996.0011.

[221] E. Wigner, On the Quantum Correction For Thermodynamic Equilibrium, Phys. Rev. 40 (1932) 749–759. doi:10.1103/PhysRev.40.749.

[222] P. Carruthers, F. Zachariasen, Quantum collision theory with phase-space distributions, Rev. Mod. Phys. 55 (1983) 245–285. doi:10.1103/RevModPhys.55.245.

[223] H. Elze, Relativistic quantum transport theory, AIP Conference Proceedings 631 (1) (2002) 229–252. doi:10.1063/1.1513683.

[224] D. Han, Y. S. Kim, M. E. Noz, Illustrative example of Feynmans rest of the universe, American Journal of Physics 67 (1) (1999) 61–66. doi:10.1119/1.19192.

[225] P. A. M. Dirac, Discussion of the infinite distribution of electrons in the theory of the positron, Mathematical Proceedings of the Cambridge Philosophical Society 30 (2) (1934) 150163. doi:10.1017/S030500410001656X.

[226] S. R. De Groot, W. A. Van Leeuwen, C. G. Van Weert, Relativistic Kinetic Theory: Principles and Applications, Elsevier North-Holland, Amsterdam, 1980.

[227] H.-T. Elze, M. Gyulassy, D. Vasak, Transport equations for the QCD quark Wigner operator, Nuclear Physics B 276 (3) (1986) 706 – 728. doi:10.1016/0550-3213(86)90072-6.

[228] H.-T. Elze, M. Gyulassy, D. Vasak, Transport equations for the QCD gluon Wigner operator, Physics Letters B 177 (3) (1986) 402 – 408. doi:10.1016/0370-2693(86)90778-1.

[229] C. Kohlfürst, Phase-space analysis of the Schwinger effect in inhomogeneous electromagnetic fields, The European Physical Journal Plus 133 (5) (2018) 191. doi:10.1140/epjp/i2018-12062-6.

[230] B. S. Xie, Z. L. Li, S. Tang, Electron-positron pair production in ultrastrong laser fields, Matter and Radiation at Extremes 2 (5) (2017) 225–242. doi:10.1016/j.mre.2017.07.002.

[231] J. C. R. Bloch, V. A. Mizerny, A. V. Prozorkevich, C. D. Roberts, S. M. Schmidt, S. A. Smolyansky, D. V. Vinnik, Pair creation: Back reactions and damping, Phys. Rev. D 60 (1999) 116011. doi:10.1103/PhysRevD.60.116011.

[232] S. A. Smolyansky, G. Roepke, S. Schmidt, D. Blaschke, V. D. Toneev, A. V. Prozorkevich, Dynamical derivation of a quantum kinetic equation for particle production in the Schwinger mechanism (1997). arXiv:hep-ph/9712377, doi:10.48550/arXiv.hep-ph/9712377.

[233] J. M. Eisenberg, G. Kalbermann, Pair production in transport equations, Phys. Rev. D 37 (1988) 1197–1201. doi:10.1103/PhysRevD.37.1197.

[234] C. Best, P. Gornicki, W. Greiner, The Phase-Space Structure of the Klein-Gordon Field, Annals of Physics 225 (2) (1993) 169–190. doi:https://doi.org/10.1006/aphy.1993.1055.

[235] C. Best, J. M. Eisenberg, Pair creation in transport equations using the equal-time Wigner function, Phys. Rev. D 47 (1993) 4639–4646. doi:10.1103/PhysRevD.47.4639.

[236] F. Hebenstreit, Schwinger effect in inhomogeneous electric fields, Ph.D. thesis, U. Graz (2011). arXiv:1106.5965, doi:10.48550/arXiv.1106.5965.

[237] F. Hebenstreit, R. Alkofer, H. Gies, Schwinger pair production in space- and time-dependent electric fields: Relating the Wigner formalism to quantum kinetic theory, Phys. Rev. D 82 (2010) 105026. doi:10.1103/PhysRevD.82.105026.

[238] F. Hebenstreit, R. Alkofer, H. Gies, Particle Self-Bunching in the Schwinger Effect in Spacetime-Dependent Electric Fields, Phys. Rev. Lett. 107 (2011) 180403. doi:10.1103/PhysRevLett.107.180403.

[239] I. A. Aleksandrov, C. Kohlfürst, Pair production in temporally and spatially oscillating fields, Phys. Rev. D 101 (2020) 096009. doi:10.1103/PhysRevD.101.096009.

[240] A. B. Yoo, M. A. Jette, M. Grondona, SLURM: Simple Linux Utility for Resource Management, in: D. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing, Springer Berlin Heidelberg, 2003, pp. 44–60. doi:10.1007/10968987_3.

[241] P. Wautelet, Best practices for parallel IO and MPI-IO hints, PATC Training session Parallel filesystems and parallel IO libraries, Maison de la Simulation (2015).
URL http://www.idris.fr/media/docs/docu/idris/idris_patc_hints_proj.pdf

[242] Leibniz Rechenzentrum (LRZ), Best Practices, Hints and Optimizations for IO (2023).
URL https://doku.lrz.de/x/9gLyAg

[243] C. Lattner, V. Adve, LLVM: a compilation framework for lifelong program analysis and transformation, in: International Symposium on Code Generation and Optimization, 2004. CGO 2004., 2004, pp. 75–86. doi:10.1109/CGO.2004.1281665.

[244] C. Lattner, LLVM and Clang: Advancing compiler technology, FOSDEM 2011: Free and Open Source Developers' European Meeting, Brussels, Belgium (2011).
URL https://llvm.org/pubs/2011-02-FOSDEM-LLVMAndClang.html

Funding and Data Statement

Funding

Initial funding for the project was provided by the Munich Cluster of Excellence “Munich-Centre for Advanced Photonics” (MAP), now Centre for Advanced Laser Applications (CALA), by the International Max-Planck Research School for Advanced Photonic Sciences (IMPRS-APS), by the German Research Foundation under Grant No. 229633566 withing the Research Unit TRR 18 on relativistic laser-plasma-dynamics.
The code has been refurbished and further developed as part of the German Research Foundation Research Unit FOR 2783 “Probing the Quantum Vacuum at the High-Intensity Frontier”. This work has been funded by the German Research Foundation under Grant Nos. 416611371; 416607684.
Large parts of the computations during the production and verification process have been performed on the *KCS* cluster computing system of the Arnold Sommerfeld Center (ASC) for Theoretical Physics at the LMU Munich, hosted at the Leibniz-Rechenzentrum (LRZ) in Garching and funded by the German Research Foundation under Grant No. 409562408.

Data and code availability

The simulation data produced for evaluation in this work amount to more than 220 gigabytes in size. They are archived on servers of the ASC, hosted by the LRZ, in compliance with the regulations of the German Research Foundation. Included are simulation data, analyses performed with presented tools, collected integrator metrics, the development git repository, and working notes. The raw data of a simulation consist of the electromagnetic field components at every grid point at a given time step. In some cases, to save on storage, the data has undergone some form of downsampling or a postprocessing filtered out certain field components of interest.
The code is publicly available under the BSD 3-Clause License and maintained on an institutional *GitLab* server: https://gitlab.physik.uni-muenchen.de/ls-ruhl/hewes.
A reproducible code capsule is published on *Code Ocean*:
https://codeocean.com/capsule/3187285/tree/v1.
There is a *Mendeley Data* repository containing extra and supplementary materials:
https://data.mendeley.com/datasets/f9wntyw39x.
Some *Mathematica* analyses are also available on the *Notebook Archive*:
https://notebookarchive.org/2022-08-eb2cjxb.

Workshops

I am thankful for a lot of inspiration and technical knowledge gain through free online workshops during the time of the pandemic. They enabled me to derive much useful further thinking, facilitating the present work. I participated at the online workshops

- "Introduction to Hybrid Programming in HPC", June 15-17, 2021, hosted at the Vienna Scientific Cluster (VSC) and organized by Claudia Blaas-Schenner and David Fischak, with lectures by Georg Hager and Rolf Rabenseifner;
- "ARCHER2 Advanced *MPI* course", July 14 and 16, 2021, hosted at the UK National Supercomputer Center (EPCC) by David Henty;
- "Parallel Performance Analysis using Scalasca", July 27 and 29, 2021, hosted by Brian Wylie, Christian Feld, and Pavel Saviankou at EPCC;
- "Efficient Parallel IO on ARCHER2", January 11, 2022, hosted by David Henty, also at EPCC;
- "HPC Code Optimisation Workshop", from June 27 to 29, 2022 at the Leibniz-Rechenzentrum (LRZ), with lectures from Patrick Böhl, Jonathan Coles, Gerald Mathias, Carla Guillen, Nisarg Patel, and Josef Weidendorfer; and
- "CMake workshop" at September 1 and 2, 2022, at the EuroCC National Competence Centre Sweden (ENCCS), taught by Roberto Di Remigio.

Moreover, the ARCHER2 training team perpetually provides the lecture and tutorial materials, including videos, of most teaching courses free to use for anyone. This enabled me to catch up with *OpenMP*, *MPI* and modern *C++*.

I highly appreciate the efforts of the Partnership for Advanced Computing in Europe (PRACE) training center in part-funding, supporting, gathering and announcing workshops from their member institutions, making them free for any EU citizen.

I further participated at the on-site "1st Pan-European Advanced School on Statistics in High-Energy Physics", October 28 to November 1, 2019, at the Deutsches Elektronen-Synchrotron (DESY) in Hamburg.

Publications

Numerical simulations of the nonlinear quantum vacuum in the Heisenberg–Euler weak-field expansion
Andreas Lindner, Baris Ölmez, and Hartmut Ruhl
Journal of Computational Physics: X 17 (2023) 100124.

HEWES: Heisenberg–Euler weak-field expansion simulator
Andreas Lindner, Baris Ölmez, and Hartmut Ruhl
Software Impacts 15C (2023) 100481

Acknowledgments

First of all, I want to thank my supervisor Hartmut Ruhl. For his trust in me and the opportunities he opened up for me, not only with the exciting topic of my PhD project. It is highly valued by your students that you are available to work even at the most unconventional times. I am looking forward to many more humorous off-topic chats!

Being a member of the DFG Research Unit FOR 2783 dedicated to the quantum vacuum at the high-intensity frontier has been an honor to me. I am grateful for the chance to contribute to the advancement of the research in this fascinating field. Special thanks are due to Felix Karbstein for fruitful discussions and Holger Gies, spokesperson of the research unit, for helpful hints and examining my work.

Many thanks are also due to Gerhard Buchalla, Thomas Kuhr, Martin Kerscher, and Dirk-André Deckert for agreeing to join my examination committee. During my days in high-energy physics I benefited from having as supervisors Gerhard Buchalla on the theoretical side and Thomas Kuhr on the experimental side. I could learn a lot from you and have always appreciated your kindness.

I am indebted to Baris Ölmez, Johannes Halbinger, Rasmus Rasche, and Yukiko Maya Song for proofreading parts of the manuscript and valuable comments.

I am fortunate to have had the possibility to cooperate with my colleague, but in the first place good friend, Baris Ölmez, who has been backing me for many years. Our gallows humor and your endless encouragement helped me get through the hard days that inevitably come during a PhD project. Since our first days as physics students we have been forming a trio with Johannes Halbinger and hopefully this friendship will last for many years to come.

Last but not least, I feel blessed to have my girlfriend Anna-Theresa, a wonderful companion over all the years, my dear sister, and my parents who enabled me to follow my passion and always supported me unconditionally on my path.