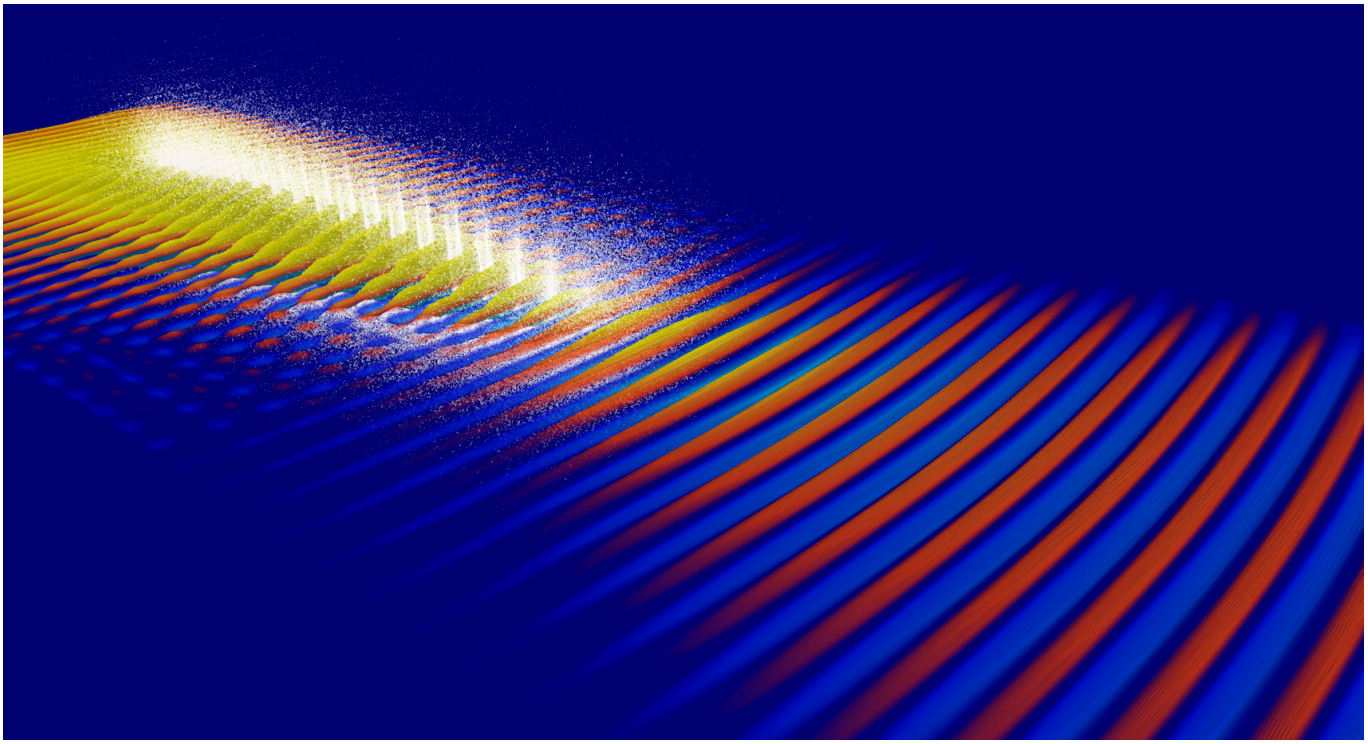
MITHRA 2.0

A Full-wave Simulation Tool for Free Electron Lasers



Mithra, also spelled Mithras, Sanskrit Mitra, in ancient Indo-Iranian mythology, is the god of light, whose cult spread from India in the east to as far west as Spain, Great Britain, and Germany. The first written mention of the Vedic Mitra dates to 1400 BC. His worship spread to Persia and, after the defeat of the Persians by Alexander the Great, throughout the Hellenic world. In the 3rd and 4th centuries AD, the cult of Mithra, carried and supported by the soldiers of the Roman Empire, was the chief rival to the newly developing religion of Christianity. The Roman emperors Commodus and Julian were initiates of Mithraism, and in 307 Diocletian consecrated a temple on the Danube River to Mithra, "Protector of the Empire."

According to myth, Mithra was born, bearing a torch and armed with a knife, beside a sacred stream and under a sacred tree, a child of the earth itself. He soon rode, and later killed, the life-giving cosmic bull, whose blood fertilizes all vegetation. Mithra's slaying of the bull was a popular subject of Hellenic art and became the prototype for a bull-slaying ritual of fertility in the Mithraic cult.

As god of light, Mithra was associated with the Greek sun god, Helios, and the Roman Sol Invictus. He is often paired with Anahita, goddess of the fertilizing waters.

Source: Encyclopaedia Britannica

Contents

List of Tables

List of Figures

Chapter 1

Preface to the New Version

The effort towards developing a code that offers accurate simulation of free-electron lasers (FEL) based on first-principle equation started in the framework of project AXSIS at DESY-Center for Free-Electron Laser science (CFEL). The project aimed at coherent X-ray radiation through novel schemes based on inverse-Compton scattering (ICS), i.e. interaction of a relativistic electron beam with a counter-propagating laser pulse. The possibility to achieve a coherent FEL radiation in a wiggling motion with undulator period as small as optical wavelength was at the time under debate. Still ongoing discussions are held in the FEL and accelerator community on the difficulties and challenges in achieving FEL gain in an ICS process. An analogous state was observed in projects aiming at coherent radiation in laser plasma wake-field acceleration (LPWA). A remarkable missing ingredient in all of the above discussions was a full-wave simulation tool that solves for the field and particle evolution in the FEL undulator.

In fact, many of the proposed novel schemes pursuing coherent FEL radiation violate the basic assumptions in FEL theory. This, of course, does not mean that such new schemes incorporatng brilliant ideas should be abandoned. However, violation of such assumptions leads to the invalidity of typical approximations that are originiated from these assumptions and typically considered in established FEL simulation tools. This situation provided me the motivation to develop a full-wave simulation tool for FEL process and the outcome is presented in this manual as the software MITHRA. Certainly, the software development attempts strongly benefited from discussions with a number of colleagues, which are highly appreciated here. Among them, I acknowledge the discussions with Prof. Alireza Yahaghi at CFEL. The collaboration with Dr. PD. Andreas Adelmann and his group was also very fruitful in further enhancement of the tool. Particularly, the work of Arnau Albà in debugging the code, checking all the implemented algorithms and reviewing the software manual is highly acknowledged. Here, I express my gratitude to Prof. William Graves for carefully studying the manual and suggesting useful comments. Eventually, my sincere gratitude to Prof. Niels Kuster for his support by providing a wonderful working environment at IT'IS foundation that was a critical factor in the latest development of MITHRA. Most of all, I acknowledge the support from Swiss National Science Foundation (SNSF) for funding the code development under the Spark grant CRSK-2_190840.

After the software MITHRA was fully developed, I thought it might be of limited usage for the community. The main shortcoming in using the code is the long simulation times required for the investigation of various interactions. Even the smallest FEL examples and simplest undulator radiation simulations require runs on massively parallel processors. This implicitly shows the utmost advantage of approximations in simulation of sophisticated instruments like a FEL. Notwithstanding, I gradually observed increasing interest in using the code MITHRA. Currently, MITHRA is being used in projects at SLAC, PSI, DESY, and ASU aiming at novel FEL concepts. To meet the needs of different projects, improvement of the MITHRA software was needed and additionally new features had to be implemented. In addition, using the code in new projects revealed small bugs which had to be fixed. This inspiring situation motivated me to prepare a second version of the software that successes the first version presented in [1]. Working on different aspects of the software to meet the needs of new projects is an endless effort. As a result, there still exist features that are foreseen to be implemented in the future.

A list of new features and applied changes in version 2.0 released with this manual is as the following:

- In the new version, user can specify the field update algorithm to be based on non-standard finite-difference or a simple finite-difference by setting the new parameter `solver` in the `MESH` group.
- An option named as `optimize-bunch-position` is added that assures the bunch residing in the middle of the computational domain after passing through the undulator entrance.
- The job file in the new version accepts a parameter named as `total-distance`, which tells the solver to run the simulation until the last particle passes a point staying at this distance from the coordinate origin.
- In the several years of using this code, I never saw a case where bunches should be initialized in the middle of a simulation.

Therefore, this option is removed from the code. This means that the parameter *bunch-time-start* is no more accepted in the job-file.

- Instead, a new option is added that enables the user to start the simulation from a previous time compared to the initial time considered by the solver. This option is activated by entering a non-zero and of course positive value for the parameter *initial-time-back-shift* in the job-file.
- A new group in the job file is added which is named as external-field. Through this group, fields of other devices than the undulator will be added to the simulation. Currently, addition of external electromagnetic waves to the interaction is implemented.
- Followed by the feedback from users which was inline with my own experience, the parallelization based on combined shared and distributed memory scheme (i.e. OpenMP and MPI) was not desired. Therefore, in the next versions parallelization is merely done based on distributed memory approach using MPI. Therefore, the previously existing parameter *number-of-threads* is no more parsed in the job file.
- The old version of MITHRA was written such that the update of motion equation was parallelized only for particles residing inside the computational domain. This leads to long computation times when particles are traveling outside the simulation domain. In the new version, the motion update for the whole bunch is distributed among available processors.
- Possibility to adjust a bunching factor phase in the bunch initialization is added through *bunching-factor-phase* parameter.
- The first version of the code was written such that the cumulative parameters of the bunch are first transferred to the moving coordinate system and subsequently the particles are initialized according to these parameters. Such a solution works only for simple bunch distributions which are thoroughly determined by their cumulative parameters. A more general approach is to generate the bunch in the laboratory frame and transfer each macro-particle according to the Lorentz transformation into the moving frame. The new version of the code considers such a scheme in the bunch initialization.
- In the new version, simulation of a Self-Amplified Spontaneous Emission (SASE) FEL is now possible. A new boolean parameter *shot-noise* is added. When it is set to true, a shot noise is calculated based on real number of electrons and subsequently introduced to the bunch. The implementation algorithm for the shot-noise is also added to the manual.
- In the previous version, the *bunch-initialization* subgroup could be repeated to initialize multiple bunches in a single simulation. While this feature is kept in the new version, the *bunch-initialization* subgroup now accepts arbitrary number of *position* vectors. As a result, at each position, determined by the position vector, the bunch is initialized. This feature is useful in initializing an array of bunches to be injected into the undulator.
- Because of an application of the code, a new field type is added to the code named as truncated-plane-wave. This is fundamentally similar to plane-wave that is confined to an elliptical region determined by the two radius parameters.
- Similarly, a new field type for simulation of beams interacting with super Guassian beams is added to the code named as *super-gaussian-beam*. The fields of a super Gaussian beam is evaluated as a superposition of several Gaussian beams depending on the beam order. This order is given to the code through *order-parallel* and *order-perpendicular* which determine the order of the super Gaussian beam parallel and perpendicular to the polarization, respectively.
- All the field types also have a standing counterpart, i.e. *standing-plane-wave* and *standing-super-gaussian-beam*, which represent the cases where these beams propagate inside a cavity forming a standing wave.
- The names *rayleigh-radius-parallel* and *rayleigh-radius-perpendicular* are changed to *radius-parallel* and *radius-perpendicular*.
- The name *variance* is changed to *pulse-length*.
- The parameter *resolution* in the field-sampling category as well as the radiation-power subgroup is changed to *number-of-points* which is more meaningful. Similarly, in the radiation-power subgroup, the *normalized-frequency-resolution* is changed to the *number-of-frequency-points*. With these changes in parameter names, the definitions of the given values are correspondingly changed.
- In the new version, the possibility to save 2D visualization data over Cartesian planes is added. In the field-visualization subgroup two parameters *type* and *plane* are added, which determine the type of the visualization (2D in-plane or 3D all-domain) and plane of the 2D data (xy, xz, or yz) respectively. Moreover, the field-visualization subgroup can be repeated in order to obtain different visualizations of the radiated fields.
- Several changes are applied to the undulator part. First, different undulator types are now introduced as subgroups in the undulator section. In the new version, a subgroup named *static-undulator-array* is added that defines an array of undulators with or without the tapering of the undulator parameter. For detailed description on how undulator arrays are introduced to the code, the user interface chapter can be studied. In addition, the undulator group is now repeatable, meaning that several undulators with different types can be given and superposed in a single FEL simulation.
- Possibility to visualize the radiated power in front of the bunch over a plane perpendicular to the undulator axis is an added feature to the software. This feature is added through the subgroup *power-visualization*.

- Another new feature is added by Arnau Albà to the software that visualizes the bunch in the lab frame. This is done by placing a screen at a certain position in the undulator and visualizing the electron bunch passing through this screen. This feature is added through the subgroup `bunch-profile-lab-frame`.
- Besides this manual, a new reference card is prepared in addition to the chapter on user interface. The content of this reference card can be used as a cheat sheet when using MITHRA. The reference card is available both separately and as a chapter in this manual.

In future, adding the following aspects to the software are planned:

- Adding the support for computation on GPU cards
- Adding the possibility of considering slow-wave approximation in time, space and both to obtain a fast computation with the cost of less accuracy
- Computing the bunching factor of the bunch as an output parameter. Currently, it can be extracted by saving the bunch profile with a certain rhythm and performing post-processing separately after the simulation.
- Computing the total radiated energy as an output parameter. Currently, it can be extracted by sampling the radiated power and performing a time-dependent integral over the radiated power.
- Implementing a far-field transformation technique to more accurately estimate the radiated power. This will avoid the problem of power underestimation due to limited area of the power-sampling plane in front of the bunch.
- Implementing UPML boundary condition to minimize the computational domain for FEL calculations
- Implementing quadrupole lattices in the region between undulator modules in an undulator array. These quadrupoles will be implemented as an external field subgroup.

Moreover, the previously presented examples are all analyzed again with the new version and new results are illustrated in this manual. In some occasions, we observed small changes compared to the old results, which are believed to happen after the removal of bugs in the previous release. I plan to update the list of examples with the new projects where MITHRA is being used. However, this task can be done only after the ongoing projects are closed and the results are disseminated. Owing to my dedication to develop open-source softwares, I have placed the code in github for any interested user to download the code and work with it. The source codes are available under the link https://github.com/aryafallahi/mithra. Eventually, I welcome any feedback from users of the code which will be an indispensable help for further improving the software performance. Besides, I appreciate if the users cite my article about the code [1] in publications of the projects in which MITHRA is used.

Arya Fallahi
Foundation for Research on Information Technology in Society (IT'IS Foundation)
Swiss Federal Institute of Technology (ETH Zürich)

Chapter 2

Introduction

Free Electron Lasers (FELs) are currently serving as promising and viable solutions for the generation of radiation in the whole electromagnetic spectrum ranging from microwaves to hard X-rays [2, 3, 4]. Particularly, in portions of spectrum where common solutions like lasers and other electronic sources do not offer efficient schemes, FEL based devices attract considerable attention and interest. For example, soft and hard X-ray radiation sources as well as THz frequency range are parts of the spectrum where FEL sources are widely used. In the optical regime, lasers currently serve as the most popular sources, where radiation is generated and amplified based on the stimulated emission. More accurately, the excited electrons of the gain medium emit coherent photons when changing the energy level to the ground state [5]. Since the energy bands of different gain media are fixed curves determined by the material atomic lattice, there are only specific wavelengths obtainable from lasers operating based on stimulated emission in a gain medium. In contrast, there exist vacuum electronic devices like gyrotrons, klystrons and travelling wave tubes (TWT), in which free electrons travelling along a certain trajectory transform kinetic energy to an electromagnetic wave [6]. Although, these sources are usually not as efficient as medium based lasers, their broadband operations make them promising in portions of the spectrum where no gain media is available.

In a free electron laser, relativistic electrons provided from linear accelerators travel through a static undulator and experience a wiggling motion. The undulator performance is categorized into two main regimes: (*i*) in a short undulator, each electron radiates as an independent moving charge, which yields an incoherent radiation of electron bunch. Therefore, the radiation power and intensity is linearly proportional to the number of electrons. (*ii*) For long interaction lengths, the radiated electromagnetic wave interacts with the bunch and the well-known micro-bunching phenomenon takes place. Micro-bunching leads to a periodic modulation of charge density inside the bunch with the periodicity equal to the radiation wavelength. This effect results in a coherent radiation scaling with the square of the bunch numbers. Coherent X-ray have shown unprecedented promises in enabling biologists, chemists and material scientists to study various evolutions and interactions with nanometer and sub-nanometer resolutions [7].

Owing to the desire of hard X-ray FEL machines for electrons with ultrarelativistic energies (0.5-1 GeV), these sources are usually giant research facilities with high operation costs and energy consumption. Therefore, it is crucial and additionally very useful to develop sophisticated simulation tools, which are able to capture the important features in a FEL radiation process. Such tools will be very helpful for designing and optimizing a complete FEL facility and additionally useful for detailed investigation of important effects. The last decade had witnessed extensive research efforts aiming to develop such simulation tools. As a result, various softwares like Genesis 1.3 [8], MEDUSA [9], TDA3D [10, 11], GINGER [12], PERSEO [13], EURA [14], RON [15], FAST [16], CHIMERA (previously PlaRes) [17] and Puffin [18] are developed and introduced to the community. However, all the currently existing simulation softwares are usually written to tackle special cases and therefore particular assumptions or approximations have been considered in their development [19]. Some of the common approximations in FEL simulation are tabulated in Table 2.1. The main goal in the presented software is the analysis of the FEL interaction without considering any of the above approximation. The outcome of the research and effort will be a sophisticated software with heavy computation loads. Nonetheless, it provides a tool for testing the validity of various approximations in different operation regimes and also a reliable approach for preparing the final design of a FEL facility.

Besides the wide investigations and studies on the conventional X-ray FELs, recently research efforts have been devoted to building compact X-ray FELs, where novel schemes for generating X-ray radiations in a so-called table-top setup are examined and assessed. Various research topics such as laser-plasma wake-field acceleration (LPWA) [20, 21, 22], laser plasma accelerators (LPA) [23, 24], laser dielectric acceleration (LDA) [25] and THz acceleration [26, 27], pursue the development of compact accelerators capable of delivering the desired electron bunches to FEL undulators. Besides such attempts, one promising approach to make a compact undulator is using optical undulators, where the oscillations in an electromagnetic wave realize the wiggling motion of

Table 2.1: Common approximations in modelling free electron laser radiation

code name	approximation					
	steady state approximation	wiggler-average electron motion	slow wave approximation	forward wave	no space-charge	slice
GENESIS 1.3	optional	✓	✓	✓	—	optional
MEDUSA	optional	—	✓	✓	—	✓
TDA3D	✓	✓	✓	✓	—	no time-domain
GINGER	—	✓	✓	✓	—	—
PERSEO	—	—	—	✓	✓	—
CHIMERA	—	—	—	✓	—	—
EURA	—	✓	✓	✓	—	—
FAST	—	✓	✓	—	—	✓
PUFFIN	—	—	—	✓	✓	—

the electrons [28]. Many of the approximations in Table 2.1, which sound reasonable for static undulators are not applicable for studying an optical undulator radiation. In this regime, due to the various involved length-scales and remarkable impact of the parameter tolerances, having access to a rigorous and robust FEL simulation tool is essential.

One of the difficulties in the X-ray FEL simulation stems from the involvement of dramatically multidimensional electromagnetic effects. Some of the nominal numbers in a typical FEL simulation are:

- Size of the bunch: $\sim$ 100 fs or 300 µm
- Undulator period: $\sim$ 1 cm
- Undulator length: $\sim$ 10-500 m
- Radiation wavelength: $\sim$ 1-100 nm

Comparing the typical undulator lengths with radiation wavelengths immediately communicates the extremely large variation space for the values. This in turn predicts very high computation costs to resolve all the physical phenomena, which is not practical even with the existing supercomputer technology. In order to overcome this problem, we exploit Lorentz boosted coordinate system and implement Finite Difference Time Domain (FDTD) [29] method combined with Particle in Cell (PIC) simulation in the electron rest frame. This coordinate transformation makes the bunch size and optical wavelengths longer and shortens the undulator period. Interestingly, these very different length scales transform to values with the same order after the coordinate transformation. Consequently, the length of the computation domain is reduced to slightly more than the bunch length making the full-wave simulation numerically feasible. We comment that the simulation of particle interaction with an electromagnetic wave in a Lorentz boosted framework is not a new concept. The advantage of this technique for the study of relativistic interactions is widely discussed [30, 31]. The method is currently the standard technique for the simulation of plasma-wakefield acceleration [32, 33, 34]. Using Lorentz-boosted equations to solve for FEL physics was previously presented in [35], where the code Warp is adapted to simulate a FEL with static undulator. In [36], the dynamics of a FEL based on optical-lattice undulator is described in the electron rest frame. Here, we are presenting a software dedicated to the analysis of FEL mechanism by solving principal equations in bunch rest frame.

Along with all the benefits offered by numerical simulation in the Lorentz-boosted framework, there exists a disadvantage emanated from treating quantities different from real three-dimensional fields in the laboratory frame. For instance, the field profile along the undulator axis at a certain time does not represent the real radiated field profile, because the fields at various points map to the corresponding values at different time points in the laboratory frame. While this feature introduces difficulties in interpreting and investigating the numerical outputs, the huge gain in computational cost justifies the analysis in the moving frame. In addition, separate modules and functions can be developed to extract the required plots in stationary frame from the computed values. This approach is implemented in the code MITHRA to obtain the radiated power.

The presented manual shows how one can numerically simulate a complete FEL interaction using merely Maxwell equations, equation of motion for a charged particle, and the relativity principles. In chapter 3, the whole computational aspects of the software, including the Finite Difference Time Domain (FDTD), Particle In Cell (PIC), current deposition, Lorentz boosting, quantity initialization, and parallelization, are described in detail. The implementation is explained in a way suitable for a graduate student to start writing the code on his own. Chapter 4 provides a detailed description of the user interface for a software user to get familiar with MITHRA and the required parameters for performing the simulations. Afterwards, in chapter 5, different examples of free electron lasers are analyzed and the results are presented in parallel with some discussions. Finally, chapter 6 presents a general reference card for users of the software. As a new software entering the FEL community, I aim to keep updating this material with new implementations and examples. In this regard, any assistance and help from the users of this software will be highly appreciated.

Chapter 3

Methodology

In this chapter, we present the detailed formalism of Finite Difference Time Domain - Particle In Cell (FDTD/PIC) method in the Lorentz boosted coordinate system. There are many small still very important considerations in order to obtain reliable results, which converge to the real values. For example, the method for electron bunch generation, particle pusher algorithm and computational mesh truncation need particular attention.

3.1 Finite Difference Time Domain (FDTD)

FDTD is perhaps the first choice coming to mind for solving partial differential equations governing the dynamics of a system. Despite its simple formulation and second order accuracy, there are certain features in this method like explicit time update and zero DC fields, which makes this method a superior choice compared to other algorithms [29]. FDTD samples the field in space and time at discrete sampling points and represents the partial derivatives with their discrete counterparts. Subsequently, update equations are derived based on the governing differential equation. Using these updating equations, a time marching algorithm is acquired which evaluates the unknown functions in the whole computational domain throughout the simulation time. In the following, we start with the wave equation which is the governing partial differential equation for our electromagnetic problem.

3.1.1 Wave Equation

The physics of electromagnetic wave and its interaction with charged particles in free space is mathematically formulated through the well-known Maxwell's equations:

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{3.1}$$

$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J} + \mu_0 \varepsilon_0 \frac{\partial \boldsymbol{E}}{\partial t} \tag{3.2}$$

$$\nabla \cdot \boldsymbol{E} = -\frac{\rho}{\varepsilon_0} \tag{3.3}$$

$$\nabla \cdot \boldsymbol{B} = 0 \tag{3.4}$$

These equations in conjunction with the electric current equation $\boldsymbol{J} = \rho \boldsymbol{v}$ ($\boldsymbol{v}$ is the charge velocity) and the Lorentz force equation:

$$\boldsymbol{F} = q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \tag{3.5}$$

are sufficient to describe wave-electron interaction in free space. Moving free electrons introduce electric current which enters into the Maxwell's equations as the source. Electric and magnetic fields derived from these equations are subsequently employed in the Lorentz force equation to determine the forces on the electrons, which in turn determine their motions. As it is evident from the above equations, there are two unknown vectors ($\boldsymbol{E}$ and $\boldsymbol{B}$) to be evaluated, meaning that six unknown components should be extracted from the equations. However, since these two vectors are interrelated and specially because there is no magnetic monopole in the nature ($\nabla \cdot \boldsymbol{B} = 0$), one can recast Maxwell's equations in a wave equation for the magnetic vector potential ($\boldsymbol{A}$) and a wave equation for the scalar electric potential (φ):

$$\boldsymbol{\nabla}^2 \boldsymbol{A} - \frac{1}{c^2} \frac{\partial^2}{\partial t^2} \boldsymbol{A} = -\mu_0 \boldsymbol{J} \tag{3.6}$$

$$\nabla^2\varphi - \frac{1}{c^2}\frac{\partial^2\varphi}{\partial t^2} = -\frac{\rho}{\varepsilon_0}, \tag{3.7}$$

where $c = 1/\sqrt{\mu_0\varepsilon_0}$ is the light velocity in vacuum. In the derivation of above equations, the Lorentz gauge $\nabla \cdot \boldsymbol{A} = -\frac{1}{c^2}\frac{\partial\varphi}{\partial t}$ is used. The original $\boldsymbol{E}$ and $\boldsymbol{B}$ vectors can be obtained from $\boldsymbol{A}$ and φ as:

$$\boldsymbol{B} = \nabla \times \boldsymbol{A} \tag{3.8}$$

$$\boldsymbol{E} = -\frac{\partial \boldsymbol{A}}{\partial t} - \nabla\varphi \tag{3.9}$$

In addition to the above equations, the charge conservation law written as

$$\nabla \cdot \boldsymbol{J} + \frac{\partial\rho}{\partial t} = 0, \tag{3.10}$$

should not be violated in the employed computational algorithm. This is the main motivation for seeking proper current deposition algorithms in the FDTD/PIC methods used for plasma simulations. It is immediately observed that the equations (3.6), (3.7), (3.10) and the Lorentz gauge introduce an overdetermined system of equations. In other words, once a current deposition is implemented that automatically satisfies the charge conservation law, the Lorentz gauge will also hold, provided that the scalar electric potential (φ) is obtained from (3.7). However, due to the space-time discretization and the interpolation of quantities to the grids, a suitable algorithm that holds the charge conservation without violating energy and momentum conservation does not exist. The approach that we follow in MITHRA is using the discretized form of (3.6) and (3.7) with the currents and charges of electrons (i.e. macro-particles) as the source and solving for the vector and scalar potential. It was shown by Umeda et al. [37], that by using similar weighting functions for both current density ($\boldsymbol{J}$) and charge density (ρ), and a proper discretization of current density based on positions of the macro-particles according to a Zigzag scheme, a charge conserving deposition scheme can be obtained. Here, we have implemented the Zigzag scheme to maintain the charge conservation in MITHRA. To obtain the fields $\boldsymbol{E}$ and $\boldsymbol{B}$ at the grid points, we use the momentum conserving interpolation, which will be explained in the upcoming sections.

3.1.2 FDTD for Wave Equation

In cartesian coordinates, a vector wave equation is written in form of three uncoupled scalar wave equations. Therefore, it is sufficient to apply our discretization scheme only on a typical scalar wave equation: $\nabla^2\psi - \frac{1}{c^2}\frac{\partial^2\psi}{\partial t^2} = \zeta$, where ψ stands for A_l ($l \in \{x, y, z\}$); and ζ represents the term $-\mu_0 J_l$. Let us begin with the central-difference discretization scheme for various partial differential terms of the scalar wave equation at the point $(i\Delta x, j\Delta y, k\Delta z, n\Delta t)$. In the following equations, $\psi^n_{i,j,k}$ denotes the value of the quantity ψ at the point $(i\Delta x, j\Delta y, k\Delta z)$ and time $n\Delta t$. The derivatives are written as follows:

$$\frac{\partial^2}{\partial x^2}\psi(x,y,z,t) \simeq \frac{\psi^n_{i+1,j,k} - 2\psi^n_{i,j,k} + \psi^n_{i-1,j,k}}{(\Delta x)^2} \tag{3.11}$$

$$\frac{\partial^2}{\partial y^2}\psi(x,y,z,t) \simeq \frac{\psi^n_{i,j+1,k} - 2\psi^n_{i,j,k} + \psi^n_{i,j-1,k}}{(\Delta y)^2} \tag{3.12}$$

$$\frac{\partial^2}{\partial z^2}\psi(x,y,z,t) \simeq \frac{\psi^n_{i,j,k+1} - 2\psi^n_{i,j,k} + \psi^n_{i,j,k-1}}{(\Delta z)^2} \tag{3.13}$$

$$\frac{\partial^2}{\partial t^2}\psi(x,y,z,t) \simeq \frac{\psi^{n+1}_{i,j,k} - 2\psi^n_{i,j,k} + \psi^{n-1}_{i,j,k}}{(\Delta t)^2}. \tag{3.14}$$

Combining these four equations, one obtains the value of ψ at instant $(n+1)\Delta t$ in terms of its value at $n\Delta t$ and $(n-1)\Delta t$:

$$\psi^{n+1}_{i,j,k} = -\psi^{n-1}_{i,j,k} + \alpha_1\psi^n_{i,j,k} + \alpha_2\psi^n_{i+1,j,k} + \alpha_3\psi^n_{i-1,j,k} + \alpha_4\psi^n_{i,j+1,k} + \alpha_5\psi^n_{i,j-1,k} + \alpha_6\psi^n_{i,j,k+1} + \alpha_7\psi^n_{i,j,k-1} + \alpha_8\zeta^n_{i,j,k}$$

where the coefficients $\alpha_1, \ldots, \alpha_7$ are obtained from:

$$\begin{aligned} &\alpha_1 = 2\left[1 - \left(\frac{c\Delta t}{\Delta x}\right)^2 - \left(\frac{c\Delta t}{\Delta y}\right)^2 - \left(\frac{c\Delta t}{\Delta z}\right)^2\right], \qquad \alpha_8 = (c\Delta t)^2, \\ &\alpha_2 = \alpha_3 = \left(\frac{c\Delta t}{\Delta x}\right)^2, \qquad \alpha_4 = \alpha_5 = \left(\frac{c\Delta t}{\Delta y}\right)^2, \qquad \alpha_6 = \alpha_7 = \left(\frac{c\Delta t}{\Delta z}\right)^2 \end{aligned} \tag{3.15}$$

The term $\zeta^n_{i,j,k}$ is the magnitude of the source term at the time $n\Delta t$, which is calculated from the particle motions. Usually, one needs a finer temporal discretization for updating the equation of motion compared to electromagnetic field equations. If the equation of motion is discretized and updated with $\Delta t_b = \Delta t/N$ time steps, the term $\zeta^n_{i,j,k}$ will be written in terms of the value after each N update:

$$\zeta^n_{i,j,k} = -\mu_0 J_l(n\Delta t) = -\mu_0 \rho(n\Delta t)\frac{\boldsymbol{r}^{n+1/2} - \boldsymbol{r}^{n-1/2}}{\Delta t}. \tag{3.16}$$

As observed in the above equation, the position of particles are sampled at each $n + 1/2$ time step, which later should be considered for updating the scalar potential. This assumption also results in the calculation of charge density at $n+1/2$ time steps, which should be averaged for obtaining $\rho(n\Delta t)$.

3.1.3 Numerical Dispersion in FDTD

It is well-known that the FDTD formulation for discretizing the wave equation suffers from the so-called numerical dispersion. More accurately, the applied discretization leads to the phase velocity of wave propagation calculated different from (lower than) the vacuum speed of light. This may impact the FEL simulation results particularly during the saturation regime, owing to the important role played by the relative phase of electrons with respect to the radiated light. Therefore, careful scrutiny of this effect and minimizing its impact is essential for the goal pursued by MITHRA.

To derive the equation governing such a dispersion, we assume a plane wave function for $\psi(x, y, z, t) = e^{-j(k_x x+k_y y+k_z z-\omega t)}$ in the discretized wave equation. After some mathematical operations, the following equation is obtained for the dispersion properties of central-difference scheme:

$$\frac{\sin^2(k_x\Delta x/2)}{(\Delta x)^2} + \frac{\sin^2(k_y\Delta y/2)}{(\Delta y)^2} + \frac{\sin^2(k_z\Delta z/2)}{(\Delta z)^2} = \frac{\sin^2(\omega\Delta t/2)}{(c\Delta t)^2}. \tag{3.17}$$

This equation is evidently different from the vacuum dispersion relation, which reads as

$$k_x^2 + k_y^2 + k_z^2 = \frac{\omega^2}{c^2}. \tag{3.18}$$

Comparison of the two equations shows that the dispersion characteristics are similar, if and only if $\Delta x \to 0$, $\Delta y \to 0$, $\Delta z \to 0$, and $\Delta t \to 0$. Another output of the dispersion equation is the stability condition, which is referred to as Courant-Friedrichs-Lewy (CFL) condition [29]. The spatial and temporal discretization should be related such that the term ω obtained from equation (3.17) has no imaginary part, i.e. $\sin^2(\omega\Delta t/2) < 1$. This implies that

$$c\Delta t < \frac{1}{\sqrt{\frac{\sin^2(k_x\Delta x/2)}{(\Delta x)^2} + \frac{\sin^2(k_y\Delta y/2)}{(\Delta y)^2} + \frac{\sin^2(k_z\Delta z/2)}{(\Delta z)^2}}}. \tag{3.19}$$

The right hand side of the above equation has its minimum when all the sinus functions are equal to one, which leads to the stability condition for the central-difference scheme:

$$\Delta t < \frac{1}{c\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}}. \tag{3.20}$$

As mentioned above, for the FEL simulation, it is very important to maintain the vacuum speed of light along the z direction (Throughout this document z is the electron beam and undulator direction). More accurately, if $k_x = k_y = 0$, $k_z = \omega/c$ should be the solution of the dispersion equation. However, this solution is obtained if and only if $\Delta t = \Delta z/c$, which violates the stability condition. To resolve this problem, various techniques are developed in the context of compensation of numerical dispersion. Here, we take advantage from the non-standard finite difference (NSFD) scheme to impose the speed of light propagation along z direction [38, 39].

The trick is to consider a weighted average along z for the derivatives with respect to x and y, which is formulated as follows:

$$\frac{\partial^2}{\partial x^2}\psi(x, y, z, t) \simeq \frac{\bar{\psi}^n_{i+1,j,k} - 2\bar{\psi}^n_{i,j,k} + \bar{\psi}^n_{i-1,j,k}}{(\Delta x)^2} \tag{3.21}$$

$$\frac{\partial^2}{\partial y^2}\psi(x, y, z, t) \simeq \frac{\bar{\psi}^n_{i,j+1,k} - 2\bar{\psi}^n_{i,j,k} + \bar{\psi}^n_{i,j-1,k}}{(\Delta y)^2}, \tag{3.22}$$

with

$$\bar{\psi}^n_{i,j,k} = \mathcal{A}\psi^n_{i,j,k-1} + (1-2\mathcal{A})\psi^n_{i,j,k} + \mathcal{A}\psi^n_{i,j,k+1}. \tag{3.23}$$

Such a finite difference scheme leads to the following dispersion equation:

$$\left(1 - 4\mathcal{A}\sin^2(k_z\Delta z/2)\right)\left(\frac{\sin^2(k_x\Delta x/2)}{(\Delta x)^2} + \frac{\sin^2(k_y\Delta y/2)}{(\Delta y)^2}\right) + \frac{\sin^2(k_z\Delta z/2)}{(\Delta z)^2} = \frac{\sin^2(\omega\Delta t/2)}{(c\Delta t)^2}. \tag{3.24}$$

It can be shown that if the NSFD coefficient $\mathcal{A}$ is larger than 0.25, and $\sqrt{(\Delta z/\Delta x)^2 + (\Delta z/\Delta y)^2} < 1$, a real ω satisfies the above dispersion equation for $\Delta t = \Delta z/c$. This time step additionally yields $k_z = \omega/c$, for $k_x = k_y = 0$.

The value we chose for $\mathcal{A}$ in MITHRA is obtained from

$$\mathcal{A} = 0.25\left(1 + \frac{0.02}{(\Delta z/\Delta x)^2 + (\Delta z/\Delta y)^2}\right). \tag{3.25}$$

The update equation can then be written as

$$\begin{aligned}\psi^{n+1}_{i,j,k} = &-\psi^{n-1}_{i,j,k} + \alpha'_1\psi^n_{i,j,k} \\ &+ \alpha'_2(\mathcal{A}\psi^n_{i+1,j,k-1} + (1-2\mathcal{A})\psi^n_{i+1,j,k} + \mathcal{A}\psi^n_{i+1,j,k+1}) + \alpha'_3(\mathcal{A}\psi^n_{i-1,j,k-1} + (1-2\mathcal{A})\psi^n_{i-1,j,k} + \mathcal{A}\psi^n_{i-1,j,k+1}) \\ &+ \alpha'_4(\mathcal{A}\psi^n_{i,j+1,k-1} + (1-2\mathcal{A})\psi^n_{i,j+1,k} + \mathcal{A}\psi^n_{i,j+1,k+1}) + \alpha'_5(\mathcal{A}\psi^n_{i,j-1,k-1} + (1-2\mathcal{A})\psi^n_{i,j-1,k} + \mathcal{A}\psi^n_{i,j-1,k+1}) \\ &+ \alpha'_6\psi^n_{i,j,k+1} + \alpha'_7\psi^n_{i,j,k-1} + \alpha'_8\zeta^n_{i,j,k}.\end{aligned} \tag{3.26}$$

where the coefficients $\alpha'_1, \ldots, \alpha'_7$ are obtained from:

$$\begin{aligned}&\alpha'_1 = 2\left[1 - (1-2\mathcal{A})\left(\frac{c\Delta t}{\Delta x}\right)^2 - (1-2\mathcal{A})\left(\frac{c\Delta t}{\Delta y}\right)^2 - \left(\frac{c\Delta t}{\Delta z}\right)^2\right], \qquad \alpha'_8 = (c\Delta t)^2, \\ &\alpha'_2 = \alpha'_3 = \left(\frac{c\Delta t}{\Delta x}\right)^2, \qquad \alpha'_4 = \alpha'_5 = \left(\frac{c\Delta t}{\Delta y}\right)^2, \qquad \alpha'_6 = \alpha'_7 = \left(\frac{c\Delta t}{\Delta z}\right)^2 - 2\mathcal{A}\left(\frac{c\Delta t}{\Delta x}\right)^2 - 2\mathcal{A}\left(\frac{c\Delta t}{\Delta x}\right)^2.\end{aligned} \tag{3.27}$$

To guarantee a dispersion-less propagation along z direction with the speed of light the update time step is automatically calculated from the given longitudinal discretization (Δz), according to $\Delta t = \Delta z/c$.

3.1.4 FDTD for Scalar Potential

Usually, due to high energy of particles in a FEL process, the FEL simulations neglect the space-charge effects by considering $\varphi \simeq 0$ [17]. However, this is an approximation which we try to avoid in MITHRA. To account for space-charge forces, one needs to solve the Hemholtz equation for scalar potential, i.e. (3.7). For this purpose, the same formulation as used for the vector potential is utilized to update the scalar potential. Nonetheless, since the position of particles are sampled at $t + \Delta t/2$ instants, the obtained value for φ^n corresponds to the scalar potential at $(n + 1/2)\Delta t$. This point should be particularly taken into consideration, when electromagnetic fields $\boldsymbol{E}$ and $\boldsymbol{B}$ are evaluated.

3.1.5 Boundary Truncation

In order to simulate the FEL problem, we consider a cube as our simulation domain. The absorbing boundary condition is also considered for updating the scalar electric potential φ at the boundaries. Therefore, we introduce the parameter ξ, which denotes either ψ or φ. The six boundaries of the cube are supposed to be at: $x = \pm l_x/2$, $y = \pm l_y/2$ and $z = \pm l_z/2$. In the following, the process for implementing Mur absorbing boundary conditions (ABCs) of the first and second order in MITHRA are discussed. We only present the formulation for the boundary conditions at $z = \pm l_z/2$. The process to extract the equations for the other four boundaries will be exactly similar.

First Order ABCs:

The partial differential equations implying first order ABCs at $z = \pm l_z/2$ are:

$$\mp\frac{\partial^2\xi}{\partial z\partial t} - \frac{1}{c}\frac{\partial^2\xi}{\partial t^2} = 0 \tag{3.28}$$

The discretized version for different terms appearing in the above equation reads as:

- At $z = -l_z/2 \;\; (k = 0)$

$$\frac{\partial^2 \xi}{\partial z \partial t} \simeq \frac{1}{2\Delta t}\left(\frac{\xi^{n+1}_{i,j,1} - \xi^{n+1}_{i,j,0}}{\Delta z} - \frac{\xi^{n-1}_{i,j,1} - \xi^{n-1}_{i,j,0}}{\Delta z}\right) \tag{3.29}$$

$$\frac{1}{c}\frac{\partial^2 \xi}{\partial t^2} \simeq \frac{1}{2c}\left(\frac{\xi^{n+1}_{i,j,1} - 2\xi^{n}_{i,j,1} + \xi^{n-1}_{i,j,1}}{\Delta t^2} + \frac{\xi^{n+1}_{i,j,0} - 2\xi^{n}_{i,j,0} + \xi^{n-1}_{i,j,0}}{\Delta t^2}\right) \tag{3.30}$$

- At $z = +l_z/2 \;\; (k = K = l_z/\Delta z)$

$$\frac{\partial^2 \xi}{\partial z \partial t} \simeq \frac{1}{2\Delta t}\left(\frac{\xi^{n+1}_{i,j,K} - \xi^{n+1}_{i,j,K-1}}{\Delta z} - \frac{\xi^{n-1}_{i,j,K} - \xi^{n-1}_{i,j,K-1}}{\Delta z}\right) \tag{3.31}$$

$$\frac{1}{c}\frac{\partial^2 \xi}{\partial t^2} \simeq \frac{1}{2c}\left(\frac{\xi^{n+1}_{i,j,K} - 2\xi^{n}_{i,j,K} + \xi^{n-1}_{i,j,K}}{\Delta t^2} + \frac{\xi^{n+1}_{i,j,K-1} - 2\xi^{n}_{i,j,K-1} + \xi^{n-1}_{i,j,K-1}}{\Delta t^2}\right) \tag{3.32}$$

Combining these equations, one obtains the boundary value of ξ at instant $(n+1)\Delta t$ in terms of its values at $n\Delta t$ and $(n-1)\Delta t$:

- At $z = -l_z/2 \;\; (k = 0)$

$$\xi^{n+1}_{i,j,0} = \beta_0 \xi^{n-1}_{i,j,0} + \beta_1 \xi^{n}_{i,j,0} + \beta_2 \xi^{n-1}_{i,j,1} + \beta_3 \xi^{n}_{i,j,1} + \beta_4 \xi^{n+1}_{i,j,1} \tag{3.33}$$

- At $z = +l_z/2 \;\; (k = K = l_z/\Delta z)$

$$\xi^{n+1}_{i,j,K} = \beta_0 \xi^{n-1}_{i,j,K} + \beta_1 \xi^{n}_{i,j,K} + \beta_2 \xi^{n-1}_{i,j,K-1} + \beta_3 \xi^{n}_{i,j,K-1} + \beta_4 \xi^{n+1}_{i,j,K-1} \tag{3.34}$$

where:

$$\beta_0 = \beta_4 = \frac{c\Delta t - \Delta z}{c\Delta t + \Delta z}, \qquad \beta_1 = \beta_3 = \frac{2\Delta z}{c\Delta t + \Delta z}, \qquad \beta_2 = -1 \tag{3.35}$$

Second Order ABCs:

The partial differential equations implying second order ABCs at $z = \pm l_z/2$ are:

$$\mp \frac{\partial^2 \xi}{\partial z \partial t} - \frac{1}{c}\frac{\partial^2 \xi}{\partial t^2} - \frac{c}{2}\frac{\partial^2 \xi}{\partial x^2} - \frac{c}{2}\frac{\partial^2 \xi}{\partial y^2} = 0 \tag{3.36}$$

The discretized version for different terms appearing in the above equation reads as:

- At $z = -l_z/2 \;\; (k = 0)$

$$\frac{\partial^2 \xi}{\partial z \partial t} \simeq \frac{1}{2\Delta t}\left(\frac{\xi^{n+1}_{i,j,1} - \xi^{n+1}_{i,j,0}}{\Delta z} - \frac{\xi^{n-1}_{i,j,1} - \xi^{n-1}_{i,j,0}}{\Delta z}\right) \tag{3.37}$$

$$\frac{1}{c}\frac{\partial^2 \xi}{\partial t^2} \simeq \frac{1}{2c}\left(\frac{\xi^{n+1}_{i,j,1} - 2\xi^{n}_{i,j,1} + \xi^{n-1}_{i,j,1}}{\Delta t^2} + \frac{\xi^{n+1}_{i,j,0} - 2\xi^{n}_{i,j,0} + \xi^{n-1}_{i,j,0}}{\Delta t^2}\right) \tag{3.38}$$

$$\frac{c}{2}\frac{\partial^2 \xi}{\partial x^2} \simeq \frac{c}{4}\left(\frac{\xi^{n}_{i+1,j,1} - 2\xi^{n}_{i,j,1} + \xi^{n}_{i-1,j,1}}{\Delta x^2} + \frac{\xi^{n}_{i+1,j,0} - 2\xi^{n}_{i,j,0} + \xi^{n}_{i-1,j,0}}{\Delta x^2}\right) \tag{3.39}$$

$$\frac{c}{2}\frac{\partial^2 \xi}{\partial y^2} \simeq \frac{c}{4}\left(\frac{\xi^{n}_{i,j+1,1} - 2\xi^{n}_{i,j,1} + \xi^{n}_{i,j-1,1}}{\Delta y^2} + \frac{\xi^{n}_{i,j+1,0} - 2\xi^{n}_{i,j,0} + \xi^{n}_{i,j-1,0}}{\Delta y^2}\right) \tag{3.40}$$

- At $z = +l_z/2 \;\; (k = K = l_z/\Delta z)$

$$\frac{\partial^2 \xi}{\partial z \partial t} \simeq \frac{1}{2\Delta t}\left(\frac{\xi^{n+1}_{i,j,K} - \xi^{n+1}_{i,j,K-1}}{\Delta z} - \frac{\xi^{n-1}_{i,j,K} - \xi^{n-1}_{i,j,K-1}}{\Delta z}\right) \tag{3.41}$$

$$\frac{1}{c}\frac{\partial^2 \xi}{\partial t^2} \simeq \frac{1}{2c}\left(\frac{\xi^{n+1}_{i,j,K} - 2\xi^{n}_{i,j,K} + \xi^{n-1}_{i,j,K}}{\Delta t^2} + \frac{\xi^{n+1}_{i,j,K-1} - 2\xi^{n}_{i,j,K-1} + \xi^{n-1}_{i,j,K-1}}{\Delta t^2}\right) \tag{3.42}$$

$$\frac{c}{2}\frac{\partial^2 \xi}{\partial x^2} \simeq \frac{c}{4}\left(\frac{\xi^{n}_{i+1,j,K} - 2\xi^{n}_{i,j,K} + \xi^{n}_{i-1,j,K}}{\Delta x^2} + \frac{\xi^{n}_{i+1,j,K-1} - 2\xi^{n}_{i,j,K-1} + \xi^{n}_{i-1,j,K-1}}{\Delta x^2}\right) \tag{3.43}$$

$$\frac{c}{2}\frac{\partial^2 \xi}{\partial y^2} \simeq \frac{c}{4}\left(\frac{\xi^{n}_{i,j+1,K} - 2\xi^{n}_{i,j,K} + \xi^{n}_{i,j-1,K}}{\Delta y^2} + \frac{\xi^{n}_{i,j+1,K-1} - 2\xi^{n}_{i,j,K-1} + \xi^{n}_{i,j-1,K-1}}{\Delta y^2}\right) \tag{3.44}$$

Combining these equations, one obtains the boundary value of ξ at instant $(n+1)\Delta t$ in terms of its values at $n\Delta t$ and $(n-1)\Delta t$:

- At $z = -l_z/2 \;\; (k = 0)$

$$\begin{aligned}\xi^{n+1}_{i,j,0} =& \gamma_0 \xi^{n-1}_{i,j,0} + \gamma_1 \xi^{n}_{i,j,0} + \gamma_2 \xi^{n-1}_{i,j,1} + \gamma_3 \xi^{n}_{i,j,1} + \gamma_4 \xi^{n+1}_{i,j,1} + \\ & \gamma_5 \xi^{n}_{i+1,j,1} + \gamma_6 \xi^{n}_{i-1,j,1} + \gamma_7 \xi^{n}_{i,j+1,1} + \gamma_8 \xi^{n}_{i,j-1,1} + \\ & \gamma_9 \xi^{n}_{i+1,j,0} + \gamma_{10} \xi^{n}_{i-1,j,0} + \gamma_{11} \xi^{n}_{i,j+1,0} + \gamma_{12} \xi^{n}_{i,j-1,0}\end{aligned} \tag{3.45}$$

- At $z = +l_z/2 \;\; (k = K = l_z/\Delta z)$

$$\begin{aligned}\xi^{n+1}_{i,j,K} =& \gamma_0 \xi^{n-1}_{i,j,K} + \gamma_1 \xi^{n}_{i,j,K} + \gamma_2 \xi^{n-1}_{i,j,K-1} + \gamma_3 \xi^{n}_{i,j,K-1} + \gamma_4 \xi^{n+1}_{i,j,K-1} + \\ & \gamma_5 \xi^{n}_{i+1,j,K-1} + \gamma_6 \xi^{n}_{i-1,j,K-1} + \gamma_7 \xi^{n}_{i,j+1,K-1} + \gamma_8 \xi^{n}_{i,j-1,K-1} + \\ & \gamma_9 \xi^{n}_{i+1,j,K} + \gamma_{10} \xi^{n}_{i-1,j,K} + \gamma_{11} \xi^{n}_{i,j+1,K} + \gamma_{12} \xi^{n}_{i,j-1,K}\end{aligned} \tag{3.46}$$

where:

$$\begin{aligned}&\gamma_0 = \gamma_4 = \frac{c\Delta t - \Delta z}{c\Delta t + \Delta z}, \qquad \gamma_1 = \gamma_3 = \frac{\Delta z\left(2 - (c\Delta t/\Delta y)^2 - (c\Delta t/\Delta x)^2\right)}{c\Delta t + \Delta z}, \qquad \gamma_2 = -1 \\ &\gamma_5 = \gamma_6 = \gamma_9 = \gamma_{10} = \frac{(c\Delta t/\Delta x)^2 \Delta z}{2(c\Delta t + \Delta z)}, \qquad \gamma_7 = \gamma_8 = \gamma_{11} = \gamma_{12} = \frac{(c\Delta t/\Delta y)^2 \Delta z}{2(c\Delta t + \Delta z)}\end{aligned} \tag{3.47}$$

Particular attention should be devoted to the implementation of Mur second order absorbing boundary condition at edges and corners. Separate usage of the above equations for second order case encounters problems in the formulation. On one hand, unknown values at grid points outside the computational domain appears in the equations, and on the other a system of overdetermined equations will be obtained. The solution to this problem is to discretize all the involved boundary conditions at the center of the cubes (for corners) or squares (for edges). A simple addition of the obtained equations cancels out the values outside the computational domain and returns the desired value meeting the considered absorbing boundary condition.

3.2 Particle In Cell (PIC)

Particle in cell (PIC) method is the standard algorithm to solve for the motion of particles within an electromagnetic field distribution. The method takes the time domain data of the fields $\boldsymbol{E}$ and $\boldsymbol{B}$ and updates the particle position and momentum according to the Lorentz force equation (3.5). We comment that the electromagnetic fields in the motion equation are the total fields in the computational domain, which in a FEL problem is equivalent to the superposition of undulator field, radiated field and the seeded field in case of a seeded FEL problem. Often considering all the individual particles involved in the problem ($\sim 10^6 - 10^9$ particles) leads to high computation costs and long simulation times. The clever solution to this problem is the macro-particle assumption, through which an ensemble of particles ($\sim 10^2 - 10^4$ particles) are treated as one single entity with charge to mass ratio equal to the particles of interest, which are here electrons. The relativistic equation of motion for electron macro-particles then reads as

$$\frac{\partial}{\partial t}(\gamma m \boldsymbol{v}) = -e(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}), \qquad \text{and} \qquad \frac{\partial \boldsymbol{r}}{\partial t} = \boldsymbol{v}, \tag{3.48}$$

where $\boldsymbol{r}$ and $\boldsymbol{v}$ are the position and velocity vectors of the electron, e is the electron charge and m is its rest mass. γ stands for the Lortenz factor of the moving particle.

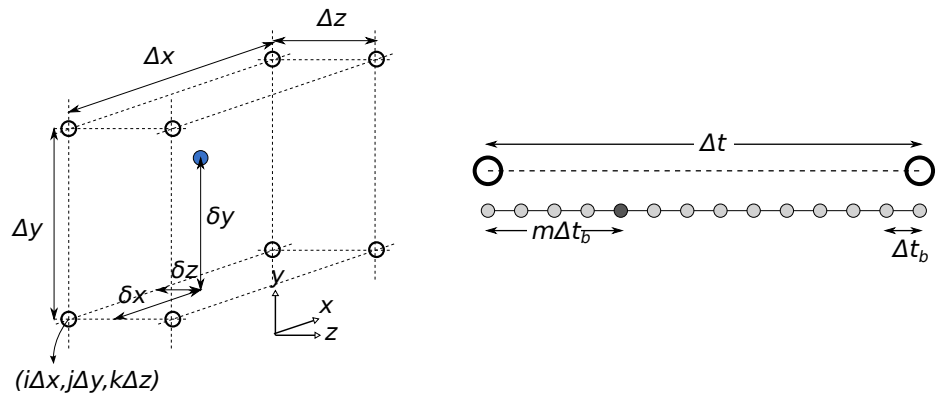


Figure 3.1: Schematic illustration of the parameters used to locate a particle within the computational domain.

3.2.1 Update Algorithm

There are numerous update algorithms proposed for the time domain solution of (3.48), including various Runge-Kutta and finite difference algorithms. Among these methods, Boris scheme has garnered specific attention owing to its interesting peculiarity which is being simplectic. Simplectic update algorithms are update procedures which maintain the conservation of any parameter in the equation which obey a physical conservation law. Since in a FEL problem effect of the magnetic field on a particle motion plays the most important role, using a simplectic algorithm is essential to obtain reliable results. This was the main motivation to choose the Boris scheme for updating the particle motion in MITHRA.

We sample the particle position at times $m\Delta t_b$, which is represented by $\boldsymbol{r}^m$ and the particle normalized momentum at times $(m-\frac{1}{2})\Delta t_b$ which is written as $\gamma\boldsymbol{\beta}^{m-1/2}$. Then, by having $\boldsymbol{r}^m$ and $\gamma\boldsymbol{\beta}^{m-1/2}$ as the known parameters and $\boldsymbol{E}_t^m$ and $\boldsymbol{B}_t^m$ as the *total* field values imposed on the particle at instant $m\Delta t$, the values $\boldsymbol{r}^{m+1}$ and $\gamma\boldsymbol{\beta}^{m+1/2}$ are obtained as follows:

$$\begin{aligned}
\boldsymbol{t}_1 &= \gamma\boldsymbol{\beta}^{m-1/2} - \frac{e\Delta t_b \boldsymbol{E}_t^m}{2mc} \\
\boldsymbol{t}_2 &= \boldsymbol{t}_1 + \alpha \boldsymbol{t}_1 \times \boldsymbol{B}_t^m \\
\boldsymbol{t}_3 &= \boldsymbol{t}_1 + \boldsymbol{t}_2 \times \frac{2\alpha \boldsymbol{B}_t^m}{1+\alpha^2 \boldsymbol{B}_t^m \cdot \boldsymbol{B}_t^m} \\
\gamma\boldsymbol{\beta}^{m+1/2} &= \boldsymbol{t}_3 - \frac{e\Delta t_b \boldsymbol{E}_t^m}{2mc} \\
\boldsymbol{r}^{m+1} &= \boldsymbol{r}^l + \frac{c\Delta t_b \gamma\boldsymbol{\beta}^{m+1/2}}{\sqrt{1+\gamma\boldsymbol{\beta}^{m+1/2}\cdot\gamma\boldsymbol{\beta}^{m+1/2}}}
\end{aligned} \tag{3.49}$$

with $\alpha = -e\Delta t_b/(2m\sqrt{1+\boldsymbol{t}_1\cdot\boldsymbol{t}_1})$. $\boldsymbol{E}_t^m = \boldsymbol{E}_{ext}^m + \boldsymbol{E}^m$ and $\boldsymbol{B}_t^m = \boldsymbol{B}_{ext}^m + \boldsymbol{B}^m$ are total fields imposed on the particle, which are equal to the superposition of the radiated field with the external fields, i.e. the undulator or the seed fields. In order to figure out the derivation of the equations (3.49), the reader is referred to [40, 41]. As seen from the above equations, the electric and magnetic fields at time $m\Delta t_b$ and the position $\boldsymbol{r}$ of the particle are needed to update the motion. In the next section, the equations to extract these values from the computed values of the magnetic and scalar potential are presented. Note that to achieve a certain precision level, the required time step in updating the bunch properties (Δt_b) is usually much smaller than the time step for field update (Δt). In MITHRA, there exists the possibility for setting different time steps for PIC and FDTD algorithms.

3.2.2 Field Evaluation

As described in section 3.1, the propagating fields in the computational domain are evaluated by solving the wave equation for the magnetic vector potential, i.e. (3.6). To update the particle position and momentum, one needs to obtain the field values $\boldsymbol{E}^m$ and $\boldsymbol{B}^m$ from the potentials $\boldsymbol{A}$ and φ. For this purpose, the equations (3.8) and (3.9) need to be discretized in a consistent manner to provide the accelerating field with lowest amount of dispersion and instability error. First, the values of magnetic and scalar potentials at $t+\Delta t/2$ are used to evaluate the electromagnetic fields at the cell vertices. Subsequently, the field values are interpolated to

the particle location for updating the equation of motion. An important consideration at this stage is compatible interpolation of fields from the cell vertices with the interpolations used for current and charge densities. Similar interpolation algorithms should be followed to cancel the effect of self-forces on particle motion.

Using the equation (3.8), the magnetic field $\boldsymbol{B}^n_{i,j,k}$ at cell vertex (i, j, k) is calculated as follows:

$$B_{x}{}^{n}_{i,j,k} = \frac{1}{2}\left(\frac{A_{z}{}^{n}_{i,j+1,k} - A_{z}{}^{n}_{i,j-1,k}}{2\Delta y} - \frac{A_{y}{}^{n}_{i,j,k+1} - A_{y}{}^{n}_{i,j,k-1}}{2\Delta z} + \frac{A_{z}{}^{n+1}_{i,j+1,k} - A_{z}{}^{n+1}_{i,j-1,k}}{2\Delta y} - \frac{A_{y}{}^{n+1}_{i,j,k+1} - A_{y}{}^{n+1}_{i,j,k-1}}{2\Delta z}\right), \tag{3.50}$$

$$B_{y}{}^{n}_{i,j,k} = \frac{1}{2}\left(\frac{A_{x}{}^{n}_{i,j,k+1} - A_{x}{}^{n}_{i,j,k-1}}{2\Delta z} - \frac{A_{z}{}^{n}_{i+1,j,k} - A_{z}{}^{n}_{i-1,j,k}}{2\Delta x} + \frac{A_{x}{}^{n+1}_{i,j,k+1} - A_{x}{}^{n+1}_{i,j,k-1}}{2\Delta z} - \frac{A_{z}{}^{n+1}_{i+1,j,k} - A_{z}{}^{n+1}_{i-1,j,k}}{2\Delta x}\right), \tag{3.51}$$

$$B_{z}{}^{n}_{i,j,k} = \frac{1}{2}\left(\frac{A_{y}{}^{n}_{i+1,j,k} - A_{y}{}^{n}_{i-1,j,k}}{2\Delta x} - \frac{A_{x}{}^{n}_{i,j+1,k} - A_{x}{}^{n}_{i,j-1,k}}{2\Delta y} + \frac{A_{y}{}^{n+1}_{i+1,j,k} - A_{x}{}^{n+1}_{i-1,j,k}}{2\Delta x} - \frac{A_{x}{}^{n+1}_{i,j+1,k} - A_{x}{}^{n+1}_{i,j-1,k}}{2\Delta y}\right). \tag{3.52}$$

Similarly, equation (3.9) is employed to evaluate the electric field at the cell vertices. The electric field $\boldsymbol{E}^n_{i,j,k}$ is obtained from the following equations:

$$E_{x}{}^{n}_{i,j,k} = \left(-\frac{A_{x}{}^{n+1}_{i,j,k} - A_{x}{}^{n}_{i,j,k}}{\Delta t} - \frac{\varphi^n_{i+1,j,k} - \varphi^n_{i-1,j,k}}{2\Delta x}\right), \tag{3.53}$$

$$E_{y}{}^{n}_{i,j,k} = \left(-\frac{A_{y}{}^{n+1}_{i,j,k} - A_{y}{}^{n}_{i,j,k}}{\Delta t} - \frac{\varphi^n_{i,j+1,k} - \varphi^n_{i,j-1,k}}{2\Delta y}\right), \tag{3.54}$$

$$E_{z}{}^{n}_{i,j,k} = \left(-\frac{A_{z}{}^{n+1}_{i,j,k} - A_{z}{}^{n}_{i,j,k}}{\Delta t} - \frac{\varphi^n_{i,j,k+1} - \varphi^n_{i,j,k-1}}{2\Delta z}\right). \tag{3.55}$$

Suppose that a particle resides at the cell ijk with the grid point indices shown in Fig. 3.1. As illustrated in Fig.3.1, the distance to the corner $(i\Delta x, j\Delta y, k\Delta z)$ is assumed to be $(\delta x, \delta y, \delta z)$. We use a linear interpolation of the fields from the vertices to the particle position to calculate the imposed field. If ς denotes for a component of the electric or magnetic field, i.e. $\varsigma \in \{E_x, E_y, E_z, B_x, B_y, B_z\}$, one can write

$$\varsigma^p = \sum_{I,J,K}\left(\frac{1}{2} + (-1)^I\left|\frac{1}{2} - \frac{\delta x}{\Delta x}\right|\right)\left(\frac{1}{2} + (-1)^J\left|\frac{1}{2} - \frac{\delta y}{\Delta y}\right|\right)\left(\frac{1}{2} + (-1)^K\left|\frac{1}{2} - \frac{\delta z}{\Delta z}\right|\right)\varsigma_{i+I,j+J,k+K}, \tag{3.56}$$

where I, J, and K are equal to either 0 or 1, producing the eight indices corresponding to the eight corners of the mesh cell.

3.2.3 Current Deposition

Once the position and momentum of all the particles over the time interval Δt is known, one needs to couple the pertinent currents into the wave equation (3.6). As described before, this coupling over time is implemented through the equation (3.16). The remaining question is how to evaluate the related currents on the grid points, i.e. the method for performing an spatial interpolation. To maintain consistency, we should use a similar interpolation scheme as used for the field evaluation. This assumption leads to the following equation for spatial interpolation.

$${\rho^p}_{i+I,j+J,k+K} = \rho\left(\frac{1}{2} + (-1)^I\left|\frac{1}{2} - \frac{\delta x}{\Delta x}\right|\right)\left(\frac{1}{2} + (-1)^J\left|\frac{1}{2} - \frac{\delta y}{\Delta y}\right|\right)\left(\frac{1}{2} + (-1)^K\left|\frac{1}{2} - \frac{\delta z}{\Delta z}\right|\right) \tag{3.57}$$

where ρ is the charge density attributed to each macro-particle, namely $q/(\Delta x\Delta y\Delta z)$. ${\rho^p}_{i,j,k}$ is the charge density at the grid point (i, j, k) due to the moving particle p in the computational mesh cell (Fig. 3.1a). I, J, and K are equal to either 0 or 1, which produce the eight indices corresponding to the eight corners of the mesh cell. The total charge density $\rho_{i,j,k}$ will be a superposition of all the charge densities due to the moving particles of the bunch. We have removed the superscripts corresponding to the time instant, to avoid the confusion due to different time marching steps Δt and Δt_b. The above interpolation is carried out at each update step of the field values. One can consider the above interpolation equations as a rooftop charge distribution centered at the particle position and expanding in the regions $(-\Delta x < x < \Delta x, -\Delta y < y < \Delta y, -\Delta z < z < \Delta z)$. Eventually, the equation (3.16) is used to calculate the corresponding current densities.

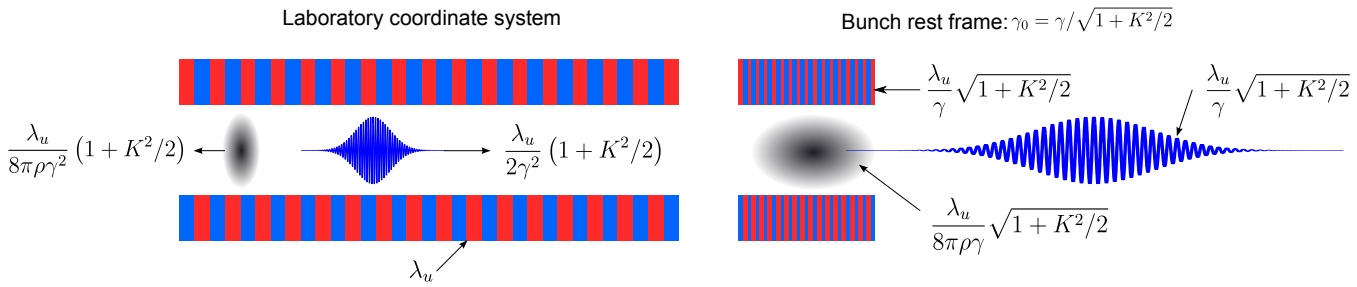


Figure 3.2: Schematic illustration of the Lorentz boosting to transform the problem from the laboratory frame to the bunch rest frame.

The combination of equation (3.16) and (**??**) should maintain the charge conservation law (equation (3.10)) in a discretized space. For this purpose, the projection from position vectors $\boldsymbol{r}$ to the Cartesian components in (3.16) should be done using the so-called ZigZag scheme proposed in [37]. According to this scheme when a particle moves from the point (x_1, y_1, z_1) to (x_2, y_2, z_2), the motion is divided into two separate movements, namely (i) from (x_1, y_1, z_1) to (x_r, y_r, z_r), and (ii) from (x_r, y_r, z_r) to (x_2, y_2, z_2). The coordinates of the relay point (x_r, y_r, z_r) are obtained from the following equation:

$$x_r = \min\left[\min(i_1\Delta x, i_2\Delta x) + \Delta x, \max\left(\max(i_1\Delta x, i_2\Delta x), \frac{x_1+x_2}{2}\right)\right]$$
$$y_r = \min\left[\min(j_1\Delta y, j_2\Delta y) + \Delta y, \max\left(\max(j_1\Delta y, j_2\Delta y), \frac{y_1+y_2}{2}\right)\right]$$
$$z_r = \min\left[\min(k_1\Delta z, k_2\Delta z) + \Delta z, \max\left(\max(k_1\Delta z, k_2\Delta z), \frac{z_1+z_2}{2}\right)\right]$$

where (i, j, k) with indices 1 and 2 represent the cell numbers containing the initial and final points, respectively. Since potential $\boldsymbol{A}$ and φ are obtained from current and charge in exactly similar ways (update equations), if charge and current obey the charge conservation, the gauge condition will be automatically satisfied. In other words, if the initial potentials satisfy the gauge condition, solving equations (3.6), (3.7), and (3.10) results in potential distributions at time t which also satisfy the gauge condition. The only requirement is that both potentials are discretized and updated in the same way.

3.3 Quantity Initialization

The previous two sections on FDTD and PIC algorithms present a suitable and efficient framework for the computation of interaction between charged particles and propagating waves. However, the initial conditions are always required for a complete determination of the problem of interest. For a FEL simulation, the initial conditions corresponding to the FEL input are given to the FDTD/PIC solver. For example, in case of a SASE (Self Amplified Spontaneous Emission) FEL, the initial fields are zero and there is no excitation entering the computational domain, whereas for a seeded FEL, an outside excitation should be considered entering the computational domain. The explanation of how such initializations are implemented in MITHRA is the goal in this section.

One novel feature of the method, followed here, is the solution of Maxwell's equations in the bunch rest frame. It can be shown that a proper coordinate transformation yields the matching of all the major parameters in a FEL simulation, namely bunch length, undulator period, undulator length, and radiation wavelength. Fig. 3.2 schematically describes the advantage of moving into the bunch rest frame. In a typical FEL problem, the FEL parameter ρ_{FEL} is about 10^{-3}. Therefore, simulation of FEL interaction with a bunch equal to the cooperation length of the FEL ($L_c = \lambda_l/(4\pi\rho_{FEL})$, with λ_l being the radiation wavelength) requires a simulation domain only 100 times larger than the wavelength. This becomes completely possible with the today computer technology and constitutes the main goal of MITHRA. In this section, the main basis for Lorentz boosting the simulation coordinate is described first. Afterwards, the relations for evaluating the undulator fields in the Lorentz boosted framework are presented. Finally, the electron bunch initialization in the Lorentz-boosted framework is discussed.

3.3.1 Lorentz Transformation

It is known from the FEL thoery that a bunch with central Lorentz factor equal to γ moves in an undulator with an average Lorentz factor equal to $\gamma_0 = \gamma/\sqrt{(1+K^2/2)}$, where $K = eB\lambda_u/(2\pi mc)$ is the undulator parameter determining the amplitude of the

wiggling motion. Consequently, a frame moving with normalized velocity $\beta_0 = \sqrt{1 - 1/\gamma_0^2}$ is indeed the bunch rest frame, where the volume of the computational domain stays minimal. Transforming into this coordinate system necessitates tailoring the bunch and undulator properties. For this purpose, the Lorentz length contraction, time dilation and relativistic velocity addition need to be employed.

In MITHRA, the input parameters are all taken in the laboratory frame and the required Lorentz transformations are carried out based on the bunch energy. The required transformations for the computational mesh are as the following:

$$\Delta z = \Delta z' \gamma_0, \tag{3.58}$$
$$\Delta t = \Delta t' / \gamma_0, \tag{3.59}$$
$$\Delta t_b = \Delta t'_b / \gamma_0, \tag{3.60}$$

where the prime sign stands for the quantities in the laboratory frame. The quantities without prime are values in the bunch rest frame, which are used in the FDTD/PIC simulation. With the consideration of the above transformations, the length of the total computational domain along the undulator period and the total simulation time is also transformed similarly.

In addition to the data for the computational mesh, the properties of the electron bunch also changes after the Lorentz boosting. This certainly affects the bunch initialization process which is thoroughly explained in the next section. An electron bunch in MITHRA is initialized and characterized by the following parameters:

(i) Mean electron position: $(\bar{x}_b, \bar{y}_b, \bar{z}_b)$,

(ii) Mean electron normalized momentum: $(\overline{\gamma\beta}_x, \overline{\gamma\beta}_y, \overline{\gamma\beta}_z)$,

(iii) RMS value of the electron position distribution: $(\sigma_x, \sigma_y, \sigma_z)$,

(iv) RMS value of the electron normalized momentum distribution: $(\sigma_{\gamma\beta_x}, \sigma_{\gamma\beta_y}, \sigma_{\gamma\beta_z})$.

As mentioned previously, the above parameters are entered by the user in the laboratory frame. The first version of the code was written such that the cumulative parameters of the bunch are first transferred to the moving coordinate system and subsequently the particles are initialized according these parameters. Such a solution works only for simple bunch distributions which are thoroughly determined by their cumulative parameters. A more general approach is to generate the bunch in the laboratory frame and transfer each macro-particle according to the Lorentz transformation into the moving frame. For this purpose, the following equations are used:

$$\begin{array}{lll} x = x', & y = y', & z = \gamma_0 z, \\ \gamma\beta_x = \gamma\beta'_x, & \gamma\beta_y = \gamma\beta'_y, & \gamma\beta_z = \gamma\beta'_z \left(\gamma\gamma_0(\beta_z - \beta_0)\right). \end{array} \tag{3.61}$$

The above equations transfer macro-particles to certain positions at different times. However, it is important that during the simulations particles are captured in the moving frame all at the same time. Therefore, the position of particles need to be changed to correct the time difference implicitly assumed in equation (3.61). This task is done by the adding the following values to the (x, y, z) coordinates of the particles, respectively:

$$\delta x = \gamma\beta_x(z - z_u)\beta_0/\gamma \tag{3.62}$$
$$\delta y = \gamma\beta_y(z - z_u)\beta_0/\gamma \tag{3.63}$$
$$\delta z = \gamma\beta_z(z - z_u)\beta_0/\gamma, \tag{3.64}$$

where z_u is the position of the undulator begin at the bunch initialization instance. The above equations consider that at $z = z_u$ no time shift exists. By using such a transformation, sophisticated bunch formats can be entered into the simulation software using the bunch type `file`, where macro-particles are read from a text file.

3.3.2 Field Initialization

The utilized FDTD/PIC algorithm solves the Maxwell's equation coupled with the motion equation of an ensemble of particles. Therefore, in addition to the field values, particle initial conditions should also be initialized. For a SASE FEL problem, the initial field profile is zero everywhere, whereas for a seeded FEL the initial seed should enter the computational domain through the boundaries. In both cases, the external field which is the undulator field should separately be initialized. In what follows, the equations implemented in the code for initializing the undulator fields and seed fields are explained.

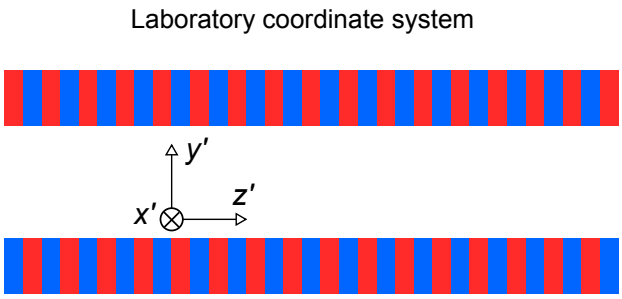


Figure 3.3: Schematic illustration of the undulator in the lab frame and the definition of the coordinates.

Static Undulator Field:

By solving the Laplace equation for the magnetic field, the undulator field in the laboratory frame is found to be as the following (Fig. 3.3) [2]:

$$\begin{aligned} B'_x &= 0, \\ B'_y &= B_0 \cosh(k_u y') \sin(k_u z'), \\ B'_z &= B_0 \sinh(k_u y') \cos(k_u z'), \end{aligned} \tag{3.65}$$

where B_0 is the maximum transverse field of the undulator. Note that the equations here are written for cases where magnetic field is zero along x-axis. As described in chapter 6, there exists a possibility in MITHRA to consider dominant field directed along a vector in the xy-plane. To calculate the undulator field in the bunch rest frame, first the position is transformed to laboratory frame (x', y', z') through the Lorentz boost equations. Afterwards, the field is evaluated using the equation (3.65). Ultimately, these fields are transformed back into the bunch rest frame. The above approach, although adds few mathematical operations for the calculation of undulator fields, it enables straightforward implementation of various realistic effects, like fringing fields of the entrance section and non-gaussian field profiles.

An important consideration in the initialization of undulator field is the entrance region of the undulator. A direct usage of the equation (3.65) with zero field for $z' < 0$ causes an abrupt variation in the particles motion, which results in a spurious coherent radiation. In fact, in a real undulator, there exists fringing fields at the undulator entrance, which remove any abrupt transition in the undulator field and consequently the particle radiations [42]. To the best of our knowledge, the fringing fields are always modeled numerically and there exists no analytical solution for the problem. Here, we approximate the fringing fields by a gradually decreasing magnetic field in form of a Neumann function. The coefficients in the function are set such that the particles do not gain any net transverse momentum and stay in the computational domain as presumed. The undulator field for $z' < 0$ in the laboratory frame is obtained as the following:

$$\begin{aligned} B'_x &= 0, \\ B'_y &= B_0 \cosh(k_u y') k_u z' e^{-(k_u z')^2/2}, \\ B'_z &= B_0 \sinh(k_u y') e^{-(k_u z')^2/2}, \end{aligned} \tag{3.66}$$

Equations (3.65) and (3.66) return the fields in the stationary frame of the undulator, i.e. the laboratory frame. To obtain the fields in the bunch rest frame, MITHRA first transfers the coordinate of input bunch from rest frame to the laboratory frame using Lorentz coordinate transformations:

$$\begin{aligned} x' &= x, \\ y' &= y, \\ z' &= \gamma_0 (z + \beta_0 ct), \end{aligned} \tag{3.67}$$

Then, the undulator field is calculated at point (x', y', z') using (3.65) and (3.66). Afterwards, the calculated field is transferred back to the bunch rest frame using the Lorentz transformation for the electromagnetic fields:

$$\boldsymbol{E} = \gamma_0 (\boldsymbol{E'} + c\boldsymbol{\beta_0} \times \boldsymbol{B'}) - (\gamma_0 - 1) E'_z \hat{\boldsymbol{z}}, \tag{3.68}$$

$$\boldsymbol{B} = \gamma_0 (\boldsymbol{B'} - \frac{\boldsymbol{\beta_0} \times \boldsymbol{E'}}{c^2}) - (\gamma_0 - 1) B'_z \hat{\boldsymbol{z}}, \tag{3.69}$$

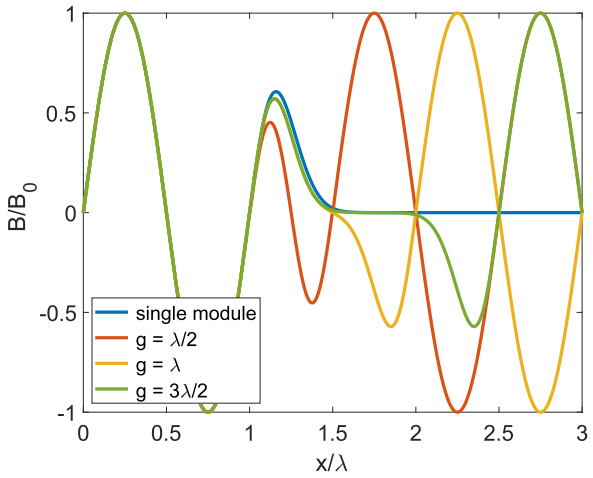


Figure 3.4: Normalized magnetic field at the center of undulator within the undulator and in the fringing field regions. Four cases are visualized and compared: (i) a single undulator module, and two undulator modules with a gap equal to (ii) half the wavelength, (ii) one wavelength, and (iii) one and a half wavelength

where $\boldsymbol{\beta_0} = \beta_0 \hat{\boldsymbol{z}}$. Since the undulator field in the lab frame is purely magnetic, in the above equation $\boldsymbol{E'} = 0$.

Static Undulator Array Field:

Calculating the field of an undulator array is identical to the field of a single module, except for the gap region between the undulator modules. If the equation (3.66) is used for each module, and simply superposed at the gap, the field values close to the two undulator boundaries will be overestimated. To solve this problem, suitable functions should on one side resemble the Gaussian damping of the field and on the other side vanish at the other end of the gap. In MITHRA, the following field variation is assumed for the fringing fields inside the gap:

$$\begin{aligned} B'_x &= 0, \\ B'_y &= B_0 \cosh(k_u y') k_u \delta z e^{-(k_u \delta z)^2/2} f(\delta z, g), \\ B'_z &= B_0 \sinh(k_u y') e^{-(k_u \delta z)^2/2} f(\delta z, g), \end{aligned} \tag{3.70}$$

with

$$f(\delta z, g) = 0.35875 + 0.48829 \cos(\frac{\pi \delta z}{g}) + 0.14128 \cos(\frac{2\pi \delta z}{g}) + 0.01168 \cos(\frac{3\pi \delta z}{g}) \tag{3.71}$$

where δz is the distance to the undulator entrance and g is the gap length between the two undulators. Note that both equations (3.66) and (3.70) are approximations of the field damping at the end of the undulator. An accurate formulation is not possible since there exists no analytical solution for the fringing fields. In order to better figure out the field variations in the gap region, the transverse magnetic field (B'_y) on the undulator axis inside an undulator and an undulator array are compared with each other in Fig. 3.4.

Optical Undulator Field:

The wiggling motion of electrons required for radiation generation can also be instigated by the oscillating fields of an electromagnetic wave. This is the main idea behind another undulator type named as optical undulator. These undulators are typically in form of an electromagnetic beam propagating counter to the electron beam. If the beam is a plane-wave, the fields are obtained as follows:

$$\begin{aligned} E_{\parallel} &= E_0 f(t, t_0, \phi), \\ B_{\perp} &= \frac{E_0}{c} f(t, t_0, \phi), \\ E_{\perp} &= B_{\parallel} = E_l = B_l = 0, \end{aligned} \tag{3.72}$$

where $\perp$ and $\parallel$ indices represent field values perpendicular and parallel to the polarization direction respectively. Subscript l denotes the longitudinal direction along the propagation line, which can be different from the undulator axis z. $f(t, t_0, \phi)$, with t_0 being the time offset and ϕ the carrier-envelope phase (CEP), is the time signature of the incoming pulse. Various signatures are implemented in MITHRA, which are listed in equation (**??**).

A more practical assumption for the counter-propagating beam is a Gaussian beam. The fields of a Gaussian beam is obtained from

$$
\begin{aligned}
E_\parallel &= E_0 \sqrt{\frac{w_{0\parallel} w_{0\perp}}{w_\parallel(l) w_\perp(l)}} \exp\left(-\frac{r_\perp^2}{w_\perp^2} - \frac{r_\parallel^2}{w_\parallel^2}\right) f\left(t, t_0, -\frac{k_0 r_\parallel^2}{2R_\parallel(l)} - \frac{k_0 r_\perp^2}{2R_\perp(l)} - \frac{\pi}{2} + \frac{\tan^{-1}\left(\frac{l}{z_{R\parallel}}\right) + \tan^{-1}\left(\frac{l}{z_{R\perp}}\right)}{2}\right), \\
E_l &= E_0 \frac{r_\parallel w_{0\parallel}}{z_{R\parallel} w_\parallel(l)} \sqrt{\frac{w_{0\parallel} w_{0\perp}}{w_\parallel(l) w_\perp(l)}} \exp\left(-\frac{r_\perp^2}{w_\perp^2} - \frac{r_\parallel^2}{w_\parallel^2}\right) f\left(t, t_0, -\frac{k_0 r_\parallel^2}{2R_\parallel(l)} - \frac{k_0 r_\perp^2}{2R_\perp(l)} + \frac{3\tan^{-1}\left(\frac{l}{z_{R\parallel}}\right) + \tan^{-1}\left(\frac{l}{z_{R\perp}}\right)}{2}\right), \\
B_l &= \frac{E_0 r_\perp w_{0\perp}}{c z_{R\perp} w_\perp(l)} \sqrt{\frac{w_{0\parallel} w_{0\perp}}{w_\parallel(l) w_\perp(l)}} \exp\left(-\frac{r_\perp^2}{w_\perp^2} - \frac{r_\parallel^2}{w_\parallel^2}\right) f\left(t, t_0, -\frac{k_0 r_\parallel^2}{2R_\parallel(l)} - \frac{k_0 r_\perp^2}{2R_\perp(l)} + \frac{\tan^{-1}\left(\frac{l}{z_{R\parallel}}\right) + 3\tan^{-1}\left(\frac{l}{z_{R\perp}}\right)}{2}\right), \\
B_\perp &= \frac{E_\parallel}{c}, \qquad E_\parallel = B_\perp = 0
\end{aligned}
\tag{3.73}
$$

where $\perp$ and $\parallel$ indices represent field components normal to the propagation direction, perpendicular and parallel to the polarization vector, respectively. l, as a subscript for the fields, stands for the component along propagation direction, and as a variable is the position along this direction, i.e. r_l. $w_\parallel = w_{0\parallel}\sqrt{1 + (r_\parallel / z_{R\parallel})^2}$ and $w_\perp = w_{0\perp}\sqrt{1 + (r_\perp / z_{R\perp})^2}$ with $w_{0\parallel}$ being the beam radius along the polarization vector, $w_{0\perp}$ the beam radius normal to the polarization vector, and $z_{R\parallel}$ and $z_{R\perp}$ are the corresponding Rayleigh range values. Parameters $R_\parallel(l) = l(1 + (z_{R\parallel}/l)^2)$ and $R_\perp(l) = l(1 + (z_{R\perp}/l)^2)$ are defined as radius of the curvature of the beam's wavefronts at position l.

Seed Field:

External excitation of free electron laser process using a seed mechanism has proved to be advantageous in terms of output spectrum, photon flux and the required undulator length [2, 43]. Such benefits has propelled the proposal of seeded FEL schemes. To simulate such a mechanism, MITHRA uses the TF/SF (total-field/scattered-field) technique to introduce an external excitation into the computational domain. When seeding is enabled by having a non-zero seed amplitude, the second and third points (after the boundary points) constitute the scattered and total field boundaries, respectively. Therefore, during the time marching process, after each update according to equation (3.26) the excitation terms are added to the fields at TF/SF boundaries. For example for the TF/SF boundaries close to $z = z_{min}$ plane, the field values to be used in the next time steps are obtained as the following:

$$
\begin{aligned}
&\text{SF boundary: } \psi'^{n+1}_{i,j,k} = \psi^{n+1}_{i,j,k} + \mathcal{A}(\alpha'_2 f^n_{i+1,j,k+1} + \alpha'_3 f^n_{i-1,j,k+1} + \alpha'_4 f^n_{i,j+1,k+1} + \alpha'_5 f^n_{i,j-1,k+1}) + \alpha'_6 f^n_{i,j,k+1}, \\
&\text{TF boundary: } \psi'^{n+1}_{i,j,k} = \psi^{n+1}_{i,j,k} - \mathcal{A}(\alpha'_2 f^n_{i+1,j,k-1} + \alpha'_3 f^n_{i-1,j,k-1} + \alpha'_4 f^n_{i,j+1,k-1} + \alpha'_5 f^n_{i,j-1,k-1}) - \alpha'_7 f^n_{i,j,k-1},
\end{aligned}
\tag{3.74}
$$

where $f^n_{i,j,k}$ is the excitation value at time $n\Delta t$ and position $(i\Delta x, j\Delta y, k\Delta z)$. The excitation value is calculated based on the imposed seed fields, which are usually either a plane wave or a Gaussian beam radiation.

3.3.3 Electron Bunch Generation

Position and momentum initialization:

As described previously, the evolution of the electron bunch is always simulated by following the macro-particle approach, where an ensemble of particles are represented by one sample particle. This typically reduces the amount of computation cost for updating the bunch properties by three or four orders of magnitude. Due to the high sensitivity of a FEL problem to the initial conditions, correct and proper initialization of these macro-particles play a critical role in obtaining reliable results. In computational accelerator physics, different approaches are introduced and developed for bunch generation. Some examples are random generation of particles, mirroring macro-particles at different phases to prevent initial average bunching factors, and independent random filling of different coordinates to prevent unrealistic correlations [44]. Among all the different methods, using the sophisticated methods to load the bunch in a "quasi-random" manner seem to be the most appropriate solutions. The Halton or Hammersley sequences, as

generalizations of the bit-reverse techniques, are implemented in MITHRA for particle generation. These sequences compared to random based filling of the phase space avoid the appearance of local clusters in the bunch distribution.

Moreover, such a uniform filling of the phase space prevents initial bunching factor of the generated electron bunch. This aspect is very beneficial in FEL simulations, since it removes any spurious initial radiation. Subsequently, the initial bunching factor or shot noise can be manually added to the particle distribution in a controlled fashion. For details on the nature of Halton sequences, the reader is referred to the specialized documents. Here, we only present the implemented algorithm to generate the required sequence of numbers filling the interval $[0, 1]$. The following C++ function is integrated into MITHRA which produces $N < 20$ uncorrelated sequences including arbitrary number of elements in the interval $[0, 1]$:

```
Double halton (unsigned int i, unsigned int j)
{
  unsigned int prime [20] = {2, 3, 5, 7, 11, 13,
     17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,
     61, 67, 71};
  int p0, p, k, k0, a;
  Double x = 0.0;

  k0 = j;

  p = prime[i];

  p0 = p;

  k  = k0;
  x  = 0.0;
  while (k > 0)
    {
      a   = k % p;
      x  += a / (double) p0;
      k   = int (k/p);
      p0 *= p;
    }

  return 1.0 - x;
}
```

By having the above uniform distributions, the 6D phase space of the initial bunch can be filled according to the desired bunch properties.

In MITHRA, different schemes for the user is implemented to generate the initial electron bunch, which are described in chapter 6. The main requirements for initializing the bunches is to generate 1D and 2D set of numbers with either uniform or Gaussian distributions. Suppose x_1 and x_2 are two uncorrelated number sequences produced by the Halton algorithm. A 1D uniform distribution y_1 with average y_{m1} and total width y_{s1} is found by the following transformation:

$$y_1 = y_{s1}(x_1 - \frac{1}{2}) + y_{m1}. \tag{3.75}$$

Such a distribution is used when a bunch with uniform current profile (z distribution of particles) is to be initialized. On the other hand, a 1D Gaussian distribution is needed when radiation of a bunch with Gaussian current profile is modelled. To generate bunches with Gaussian distribution, we employ Box-muller's theory to extract a sequence of numbers with Gaussian distribution from two uncorrelated uniform distributions. Based on this theory, a 1D Gaussian distribution y_2 with average y_{m2} and deviation width y_{s2} is found by the following transformation:

$$y_2 = y_{s2}\sqrt{-2\ln x_1}\cos(2\pi x_2) + y_{m2}. \tag{3.76}$$

Similar to the undulator fields, an abrupt variation in the bunch profile results in an unrealistic coherent scattering emission (CSE), which happens if the uniform bunch distribution is directly initialized from equation (3.75). CSE is avoided by imposing smooth variations in the particle distribution. For this purpose, we follow the procedure proposed in [45] and [44]. A small Gaussian bunch with the same density as the real bunch and a width equal to an undulator wavelength is produced. The lower half of the bunch (particles with smaller z) is transferred to the tail and the other half is placed at the head of the uniform bunch. Hence, a uniform current profile with smooth variations at its head and tail is created.

The transverse coordinates of the bunches are initialized using 2D distributions. In MITHRA, a 2D Gaussian distribution is assumed for transverse coordinates. To generate such a distribution, two independent sets of numbers x_1 and x_2 are generated based on Halton sequence. The desired 2D Gaussian distribution with average position (y_{m3}, y_{m4}) and total deviation (y_{s3}, y_{s4}) is produced as the following:

$$y_3 = y_{s3}\sqrt{-2\ln x_1}\cos(2\pi x_2) + y_{m3}, \qquad \text{and} \qquad y_4 = y_{s4}\sqrt{-2\ln x_1}\sin(2\pi x_2) + y_{m4}. \tag{3.77}$$

Such algorithms are similarly used to generate the distribution in particle momenta. The only difference is that for initializing a distribution in momentum merely Gaussian profiles are considered in transverse and longitudinal coordinates. The method to introduce these bunch types are described in the next chapter.

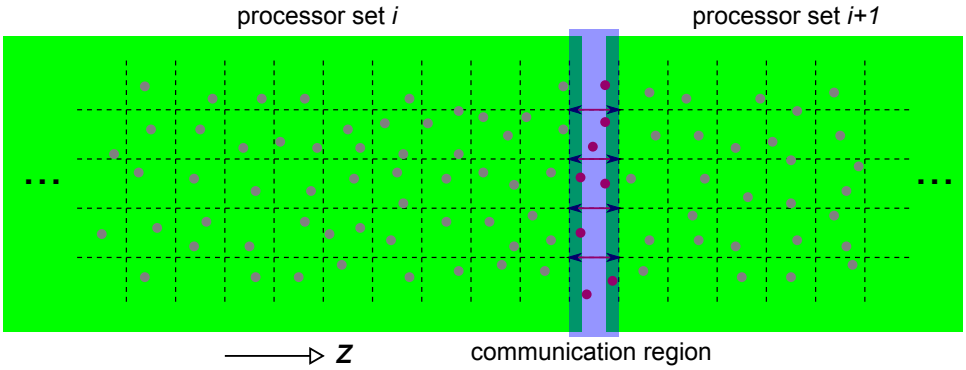


Figure 3.5: Schematic illustration of the domain decomposition used for distributed memory parallelization in MITHRA

Bunching factor:

Free electron laser radiation should start from a nonzero initial radiation. This radiation can be in form of an initial seed field, initial modulation in the bunch, or the radiation from bunch shot noise. The implementation of seeding through an external excitation using TF/SF boundaries was described in 3.3.2. Here, we explain how an initial bunching factor, $B =< e^{jk_u z} >$, is introduced to the electron bunch profile.

For this purpose, the methodology introduced in [46] is followed. A small variation δz is applied to a particle distribution generated using the described formulations. δz for each particle is obtained from

$$\delta z = \xi\gamma_0 b_f/k_u \sin(2\xi\gamma_0 k_u z), \tag{3.78}$$

where b_f is the given bunching factor of the distribution, and $\xi = 1 + \bar{\beta}_z/\beta_0$ accounts for the change in the bunch longitudinal velocity after entering the undulator. The introduced variation to the bunch coordinates, i.e. $z \rightarrow z + \delta z$, yields a bunch with all the given particle and momentum distributions and the desired bunching factor, b_f.

Shot noise:

The number of particles (electrons) in a bunch is limited. As a result, the average of bunching factor magnitudes over the whole bunch ($|B|^2 = 1/N_e$) does not tend to zero, meaning that there exists an initial total radiation in form of a noise. This radiation commonly referred to as *shot noise* can also be a trigger for the free-electron lasing process. Such a mechanism is the basis for Self-Amplified Spontaneous Emission of radiation (SASE) type of FELs. To simulate shot noise, bunch initialization starts with a uniform particle distribution obtained from Halton and Hammersley series. Afterwards, a small variation δz is applied to the particle distribution. δz for a particle residing in slice j is obtained from

$$\delta z = \xi\gamma_0 k_u b_{fj} \sin(2\xi\gamma_0 k_u z + \phi_j), \tag{3.79}$$

where b_{fj} and ϕ_j are the bunching factor value and phase in the slice j. The other parameters are defined in the same way as described in the bunching factor section. The value of b_{fj} for different slices is obtained from a negative exponential distribution according to

$$b_{fj} = \frac{1}{\sqrt{N_e}}\sqrt{-2\ln x_j}, \tag{3.80}$$

where x_j is obtained from a uniform Halton sequence. The value of ϕ_j as the bunching factor phase in various slices is calculated based on a uniform distribution (i.e. Halton sequence) over the interval $[0, \pi]$.tttt

3.4 Parallelization

The large and demanding computation cost needed for the simulation of the FEL process even in the Lorentz boosted coordinate frame necessitates solving the problem on multiple processors to achieve reasonable computation times. Therefore, efficient parallelization techniques should be implemented in the FDTD/PIC algorithm to develop an efficient software. Traditionally, there are two widely used techniques to run a computation in parallel on several processors: (1) *shared* memory, and (2) *distributed*

memory parallelization. In the shared memory parallelization or the so-called multi-threading technique, several processors run a code using the variables saved in one shared memory. This technique is very suitable for PIC algorithms because it avoids the additional costs of communicating the particle position and momenta between the processors. On the other hand, distributed memory technique distributes the involved variables among several processors, solves the problem in each processor independently and communicates the required variables whenever they are called. The distributed memory technique is very suitable for FDTD algorithm due to the ease of problem decomposition beyond various machines. The advantage is fast reading and writing of the data and the possibility to share the computational load between different machines.

Choosing a suitable parallelization scheme for the hybrid FDTD/PIC algorithm depends on both problem size and machine implementations. In MITHRA, we use distributed memory technique for parallelization of the radiation computations. The total computational domain is decomposed to several separate regions, each of them solved by one processor. These sets of processors communicate the required variables based on the technique visualized in Fig. 3.5.

To parallelize the computation among N processors, the whole computational domain is divided into N domains along z (undulator period) axis. In each time update of the field, the field values at the boundaries of each domain are communicated with the corresponding processor. To parallelize the PIC solver, we define a communication domain which as shown in Fig. 3.5, is the region between the boundaries of each processor. After each update of the particles position, it is checked if the particle has entered a communication domain. In case of residing in the communication region, the master processor, which is the processor containing the particle in the previous time step, communicates the new coordinates to the slave processor, which is the processor sharing the communication region with the master one. Through this simple algorithm, the whole computation is distributed among the available processors of the machine.

Chapter 4

User Interface

This chapter, as apparent from its name, is considered as a reference for the MITHRA user interface. The aim here is presenting the functions and variables which can be delivered to the MITHRA software and can be handled for a FEL simulation problem. In what follows in this chapter, the defined language of MITHRA for writing a compatible job file is introduced. This chapter can also be considered as a reference for the current capabilities of MITHRA and with time will be updated with the further improvement of the software capabilities.

Iron Rule: parameters that are used for the solution of a specific electromagnetic problem are delivered to the code at only one single location, *the job file*. This is indeed the only thing that the solver takes as an input parameter.

It should be noted that all the parameters in job file are given in the laboratory frame. The Lorentz boost into the bunch rest frame will be done by the software automatically.

To run a job file using MITHRA, the following command should be written in the linux command line:

- **`mpirun -np`** "number of distributed processors" "MITHRA object file name" "job file name"

The transferred job file to the solver contains five main sections, each one defining an essential part of the electromagnetic problem. These sections include:

1. *`MESH:`* The parameters of the FDTD solver like the computational domain, cell sizes and time steps are set in this section.
2. *`BUNCH:`* The required data to initialize the electron bunch in the computational domain is set in this section. In addition, the desired type of recording the bunch evolution is entered in this section by the user.
3. *`FIELD:`* This section fulfills the same task as the previous section for the electromagnetic fields. The field initialization in case of a seeded FEL and the desired output type for the field evolution is given in this section to the software.
4. *`UNDULATOR:`* This section introduces the different parameters of the undulator.
5. *`EXTERNAL-FIELD:`* This section introduces the fields of some external components to the FEL interaction. It is relatively rare to have external components superimposed on the undulator field. However, such a possibility enables studying novel and advanced FEL cases.
6. *`FEL-OUTPUT:`* The desired data related to the FEL radiation and how to record this data is set in this section.

In the next subsections, we explain each part and the supported parameters, respectively. To write comments in your job file use the sign "#" at the beginning of the comment and the text will be commented to the end of the line.

4.1 `MESH`

As mentioned above, this part is dedicated to the determination of the FDTD/PIC parameters. In Fig 4.1, a typical computation domain assumed in MITHRA is depicted. The mesh and update parameters of the solver are defined through the following ten parameters:

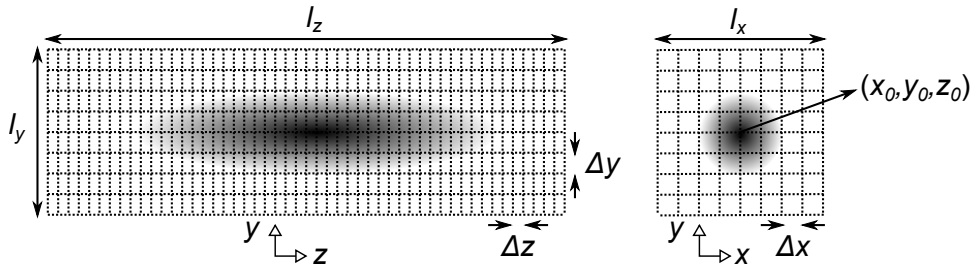


Figure 4.1: The definition of the spatial mesh parameters in MITHRA

- `length-scale` is the scaling of the length and all the spatial parameters in the job file. The capability to play with length scales is crucial to avoid working with very large or very small numbers.
- `time-scale` is the scaling of the time and all the temporal parameters in the job file. Similar to above, through this capability working with very large or very small numbers is avoided.
- `mesh-lengths` is a three dimensional vector equal to the lengths (l_x, l_y, l_z) of the computational domain (4.1) along the three Cartesian axes.
- `mesh-resolution` defines the length of one single grid cell or in other words the spatial discretization resolution of the FDTD mesh in the laboratory coordinate system $(\Delta x', \Delta y', \Delta z')$.
- `mesh-center` is the position of the central point of the computational rectangle, i.e. (x'_0, y'_0, z'_0) in Fig. 4.1.
- `total-time` is the total computation time in the scale given by the time scale. This is indeed the time it takes for the electron bunch to travel through the considered undulator length.
- `total-distance` is the total traveled distance by the bunch. Once this parameter is set, the given `total-time` will be ignored and the computation will be continued as long as the last particle in the bunch passes through a point that resides on the given distance from the coordinate origin.
- `bunch-time-step` is the time step for updating the macro-particles' coordinates in the PIC solver. The default value is the value calculated from the mesh using the stability condition.
- `mesh-truncation-order` is the truncation order of the absorbing boundary condition in the computational domain. This parameter can be either 1 or 2, representing the first order and second order absorbing boundary condition.
- `space-charge` is a boolean flag determining if the space-charge effect should be considered or not. If this flag is false, the scalar potential ϕ is zero throughout the calculation. Otherwise, the scalar potential is calculated using the corresponding Helmholtz equation.
- `solver` determines if the non-standard finite-difference (NSFD) algorithm should be used to remove the effects of numerical dispersion or the simulation should be done with a simple finite-difference (FD) algorithm. Default is the non-standard finite-difference.
- `optimize-bunch-position` is a boolean flag that tells the solver to automatically shift the bunch so that it resides in the middle of the computational domain after passing through the undulator entrance. This parameter is by default set to false.
- `initial-time-back-shift` is a real positive value that tells the solver to start the simulation from a time before the standard initial condition of solver. We comment that the solver automatically places the bunch head in a given distance from the undulator entrance.
- `lorentz-factor` is a the Lorentz factor of the moving mesh. If this value is not given the solver automatically computes an optimal value. It is generally recommended to leave this option so that the solver does the automatic computation. However, in some rare cases setting manually the Lorentz factor of the mesh is needed.

The format of the `MESH` group is:

```
MESH
{
    length-scale                = < real | METER | DECIMETER | CENTIMETER | MILLIMETER |
                                    MICROMETER | NANOMETER | ANGSTROM >
    time-scale                  = < real | SECOND | MILLISECOND | MICROSECOND | NANOSECOND |
                                    PICOSECOND | FEMTOSECOND |  ATTOSECOND >
    mesh-lengths                = < ( real, real, real ) >
    mesh-resolution             = < ( real, real, real ) >
    mesh-center                 = < ( real, real, real ) >
    total-time                  = < real >
    total-distance              = < real >
    bunch-time-step             = < real >
    mesh-truncation-order       = < 1 | 2 >
    space-charge                = < true | false >
    solver                      = < NSFD | FD >
    optimize-bunch-position     = < true | false >
    initial-time-back-shift     = < real >
    lorentz-factor              = < real >
}
```

An example of the computational mesh definition looks as the following:

```
MESH
{
    length-scale                = MICROMETER
    time-scale                  = PICOSECOND
    mesh-lengths                = ( 3200,  3200.0,    280.0)
    mesh-resolution             = ( 50.0,    50.0,      0.1)
    mesh-center                 = ( 0.0,      0.0,      0.0)
    total-time                  = 30000
    bunch-time-step             = 1.6
    mesh-truncation-order       = 2
    space-charge                = false
    solver                      = NSFD
    optimize-bunch-position     = false
    initial-time-back-shift     = 0.0
}
```

Note that there are some conditions, which should be fulfilled for the numerical integrator to obtain reliable dispersion-less results. The software checks for these conditions before starting to solve the problem, if the conditions are violated the closest value to the given number meeting the violated conditions will be used. Regarding the above parameters. the software checks for the stability condition $\sqrt{(\Delta z/\Delta x)^2 + (\Delta z/\Delta y)^2} < 1$, adapts the values of Δx and Δy accordingly, and finally sets the time step for field update equal to $\Delta z/c$. In addition, the bunch update time step should be an integer fraction of the field time step to avoid redundant dispersion in the calculated values. Therefore, the closest value to the given bunch time step, which satisfies the above criterion, will be chosen.

4.2 BUNCH

The section *BUNCH* is the main part of the job file to establish the required data for the bunch input and output framework. This section consists of four groups: (1) *bunch-initialization*, (2) *bunch-sampling*, (3) *bunch-visualization*, and (4) *bunch-profile*. As apparent from the name the first group determines the set of parameters to initialize the bunch and the other three groups are dedicated to reporting the bunch evolution in different formats. In what follows, the parameters in each group are introduced:

1. *bunch-initialization:* This group mainly determines the parameters whose values are needed for initializing a bunch of electrons with different types. If several bunches are present in a simulation, this group should simply be repeated in the *BUNCH* section. The set of values accepted in this group include:

- `type` is the type of the bunch to be initialized in the computational domain. There are four bunch types supported by MITHRA:
  (a) `manual` initializes charges at the points specified by the position vector. At each appearance of this type of bunch only one single macro-particle will be initialized. Therefore, to have multiple manual initialization, the `bunch-initialization` group should be repeated. Using the `file` type is a better solution for high number of manual inputs. Alternatively, one can repeat the `position` parameter to manually inject several particles.
  (b) `ellipsoid` initializes charges with a given distribution over an ellipsoid defined by the `sigma-position` parameter.
  (c) `3D-crystal` initializes multiple bunches on the points of a 3D crystal centered at the coordinate specified by the position vector and extends over the space by the number vector and the considered lattice constant. Each single bunch has a ellipsoid Gaussian property with the values read from the deviation parameters.
  (d) `file` reads a list of 6D position and momentum coordinates from a file and initializes the macro-particles correspondingly in the solver. The format of the file that is read by MITHRA is a text (`.txt`) file. In this file, each line presents the properties of one macro-particle that should be initialized in the code. In each line, six values corresponding to position of the macro-particle (x, y, z) and its normalized momentum $(\gamma\beta_x, \gamma\beta_y, \gamma\beta_z)$ are written. This simple format is also the general format of all the bunch profiles produced by the MITHRA code.
- `distribution` determines if the initialized particle distribution should have a uniform or Gaussian current profile. In MITHRA, the transverse distributions are always Gaussian, unless the bunch is given manually. This parameter merely affects the distribution along the traveling path, i.e. z.
- `file-name` returns the name of the file where the bunch data is written for a bunch of type `file`. If the file does not reside in the current directory file-name should contain the file address.
- `number-of-particles` is the total number of particles (or macro-particles) considered in the bunch. The value should be a multiple of 4. Otherwise, the solver automatically changes the given value to the closest multiple of 4.
- `charge` is the total charge of the bunch in one electron charge unit. In other words, it stands for the total number of electrons in the bunch.
- `gamma` is the initial mean Lorentz factor of the bunch.
- `beta` is the initial mean normalized velocity of the particles, if it is not determined here the value will be calculated from the `gamma` parameter, otherwise the same `beta` will be used.
- `direction` is the average momentum direction of the bunch, i.e. $(\beta_x, \beta_y, \beta_z)/\beta$. In a typical FEL example, this parameter should be $(0, 0, 1)$.
- `position` is the central position of the bunch. This parameter can be repeated to initialize multiple bunches with similar profiles at different positions.
- `sigma-position` is the RMS deviation in position of the bunch, i.e. $(\sigma_x, \sigma_y, \sigma_z)$ for Gaussian distributions. σ_z is half the bunch length for the uniform distribution.
- `sigma-momentum` is the RMS deviation in energy of the bunch, i.e. $(\sigma_{\gamma\beta_x}, \sigma_{\gamma\beta_y}, \sigma_{\gamma\beta_z})$.
- `numbers` is a parameter read only when the bunch type is a `3d-crystal` type. It is the number of bunch replication in the three directions.
- `lattice-constants` is a parameter read only when the bunch type is a `3d-crystal` type. It is the length of lattice constants of the crystal in the three directions.
- `transverse-truncation` determines a limit to transversely truncate the bunches. This factor brings the possibility to control particle initialization and prevents them from escaping out of the computational domain. The bunch initializer truncates the bunch at the given distance from the bunch center.
- `longitudinal-truncation` determines a limit to longitudinally truncate the bunches. This factor brings the possibility to control particle initialization and prevents them from escaping out of the computational domain. The bunch initializer truncates the bunch at the given distance from the bunch center.
- `bunching-factor` is a value larger than zero and less than one, which determines the bunching factor, i.e. $< e^{jk_u z} >$, of the initialized bunch.
- `bunching-factor-phase` is the initial phase of the bunching factor that is read in the parameter `bunching-factor`.
- `shot-noise` is a boolean parameter that determines if the shot-noise should be introduced to the bunch initialization or not. To model, SASE FEL, this parameter should be set to `true`.

2. *bunch-sampling:* This group defines the required parameters for saving the bunch properties with time. The bunch mean position, mean momentum, position spread, and momentum spread along the three Cartesian coordinates are saved respectively with time. There are different parameters required for this definition which include:

   - *sample* is a boolean value determining if the bunch sampling should be activated.
   - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
   - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
   - *rhythm* is a real value returning the rhythm of bunch sampling, i.e. the time interval between two consecutive sampling times.

3. *bunch-visualization:* This group defines the required parameters for visualizing the charge distribution in the whole computational domain. The output will be a set of *.vtu* files at each time for each processor, which are connected with a set of *.pvtu* files. They can be very nicely visualized using the open source ParaView package. There are different parameters required for this definition which include:

   - *sample* is a boolean value determining if the charge visualization should be activated.
   - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
   - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
   - *rhythm* is the rhythm of charge visualization, i.e. the time interval between two consecutive visualization times.

4. *bunch-profile:* This group defines the required parameters for saving a histogram of the charges. It means that at a specific time instant the charge values, positions and momenta of all the particles (or macro-particles) will be written and saved in a file. The parameters entered by the user for saving the histogram include:

   - *sample* is a boolean value determining if the writing of the histogram during the PIC simulations should be activated.
   - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
   - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
   - *time* is the time instant for saving the histogram. If this needs to be done in several time instants, simply this line should be repeated with different time values.
   - *rhythm* is the rhythm of writing the bunch profile, i.e. the time interval between two consecutive profiling times. If this value is nonzero, the sequence of times will be considered in addition to the specific time points given by the time variable.

The format of the *BUNCH* group is (The repeatable variables are shown in red):

```
BUNCH
{
    bunch-initialization
    {
        type                    = < manual | ellipsoid | 3D-crystal | file >
        distribution            = < uniform | gaussian >
        file-name               = < string >
        charge                  = < real >
        number-of-particles     = < int >
        gamma                   = < real >
        beta                    = < real >
        direction               = < ( real, real, real ) >
        position                = < ( real, real, real ) >
        sigma-position          = < ( real, real, real ) >
```

```
        sigma-momentum            = < ( real, real, real ) >
        numbers                   = < ( int, int, int ) >
        lattice-constants         = < ( real, real, real ) >
        transverse-truncation     = < real >
        longitudinal-truncation   = < real >
        bunching-factor           = < real between 0 and 1 >
        bunching-factor-phase     = < real >
        shot-noise                = < true | false >
    }

    bunch-sampling
    {
        sample                    = < true | false >
        directory                 = < /path/to/location >
        base-name                 = < string >
        rhythm                    = < real >
    }

    bunch-visualization
    {
        sample                    = < true | false >
        directory                 = < /path/to/location >
        base-name                 = < string >
        rhythm                    = < real >
    }

    bunch-profile
    {
        sample                    = < true | false >
        directory                 = < /path/to/location >
        base-name                 = < string >
        time                      = < real >
        rhythm                    = < real >
    }
}
```

An example of the bunch category definition looks as the following:

```
BUNCH
{
    bunch-initialization
    {
        type                      = ellipsoid
        distribution              = uniform
        charge                    = 1.846e8
        number-of-particles       = 131072
        gamma                     = 100.41
        direction                 = ( 0.0, 0.0, 1.0)
        position                  = ( 0.0, 0.0, 0.0)
        sigma-position            = ( 260.0, 260.0, 50.25)
        sigma-momentum            = ( 1.0e-8, 1.0e-8, 100.41e-4)
        transverse-truncation     = 1040.0
        longitudinal-truncation   = 90.0
        bunching-factor           = 0.01
        bunching-factor-phase     = 0.0
        shot-noise                = false
    }

    bunch-sampling
```

```
    {
        sample                    = false
        directory                 = ./
        base-name                 = bunch-sampling/bunch
        rhythm                    = 3.2
    }

    bunch-visualization
    {
         sample                   = true
         directory                = ./
         base-name                = bunch-visualization/bunch
         rhythm                   = 32
    }

    bunch-profile
    {
         sample                   = false
         directory                = ./
         base-name                = bunch-profile/bunch
         time                     = 5000
         time                     = 10000
         time                     = 15000
         time                     = 20000
         time                     = 25000
         time                     = 30000
    }
}
```

4.3 FIELD

In section *FIELD*, the required data for the input and output framework of the field in the FDTD algorithm is produced. This section consists of four groups: (1) *field-initialization*, (2) *field-sampling*, (3) *field-visualization*, and (4) *field-profile*. As apparent from the name, the first group determines the set of parameters to initialize the field and the other three groups are dedicated to reporting the field propagation in different formats. In what follows, the parameters in each group are introduced:

1. *field-initialization:* This group mainly determines the parameters whose values are needed for initializing a field excitation entering the computational domain. The excitation may have different types. This group is where a seed can be added to the simulations to simulate a seeded-FEL problem. The set of values accepted in this group include:

    - *type* is the type of the excitation or the seed field. The accepted excitation types in MITHRA include plane wave, truncated plane-wave, Gaussian beam, and super-Gaussian beam. A truncated plane-wave is a plane-wave that introduces fields to the particle only over an ellipse determined by beam radii.
    - *position* is the reference position of the excitation. It is the reference position of the plane wave propagation in the plane-wave excitation and the focusing point in the Gaussian beam excitation. The coordinate system for this position vector is the same as for bunch position vector. Typically, the focal point or the reference position of the seed is given with respect to the undulator begin. Therefore, special care should be exercised with this position vector; If the reference position with respect to undulator begin is z_0, then the value for seed reference position should be given as $z_0+l_z+10\lambda_X$, where l_z is the *longitudinal truncation* value and λ_X is the radiation wavelength.
    - *direction* is the propagation direction of the excitation in the plane-wave and Gaussian beam types.
    - *polarization* is the polarization of the incoming excitation and is used by both plane-wave and Gaussian-beam types.
    - *radius-parallel* is the Rayleigh radius (beam waist) of the Gaussian beam in the direction parallel to the polarization. For the truncated plane-wave it is the radius of the ellipse along the polarization direction confining the plane wave.

- `radius-perpendicular` is the Rayleigh radius (beam waist) of the Gaussian beam in the direction perpendicular to the polarization. For the truncated plane-wave it is the radius of the ellipse perpendicular to the polarization direction confining the plane-wave.
- `order-parallel` is the order of the super Gaussian beam along the field polarization.
- `order-perpendicular` is the order of the super Gaussian beam perpendicular to the field polarization.
- `signal-type` determines the time signature of the signal exciting the fields according to the particular type. The accepted signal types in MITHRA include modulated Neumann, modulated Gaussian, modulated secant hyperbolic, sinusoidal (flat-top), and the inverse-Gaussian pulse. The equations representing the time domain variation of each pulse are as follows:

  (a) `neumann:`

$$f(t) = -A_0 4 \ln 2 \ \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) \frac{t-t_0}{\tau^2} e^{-2\ln 2\,(t-t_0)^2/\tau^2} \tag{4.1}$$

  (b) `gaussian:`

$$f(t) = A_0 \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) e^{-2\ln 2\,(t-t_0)^2/\tau^2} \tag{4.2}$$

  (c) `secant-hyperbolic:`

$$f(t) = A_0 \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) \frac{1}{\cosh((t-t_0)/\tau)} \tag{4.3}$$

  (d) `flat-top:`

$$f(t) = A_0 \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) \begin{cases} e^{-(t-t_0-\tau/2)^2/(N/f)^2} & \|t-t_0\| \geq \tau/2 \\ 1.0 & \|t-t_0\| \leq \tau/2 \end{cases} \tag{4.4}$$

  (e) `inverse-gaussian:`

$$f(t) = A_0 \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) \sqrt[4]{\left(1+\frac{t^2}{\sigma_1^2}\right)\left(1+\frac{t^2}{\sigma_2^2}\right)} \begin{cases} e^{-(t-t_0-\tau/2)^2/(N/f)^2} & \|t-t_0\| \geq \tau/2 \\ 1.0 & \|t-t_0\| \leq \tau/2 \end{cases} \tag{4.5}$$

- `strength-parameter` is the normalized amplitude $a_0 = eA_0/m_e c$ of the beam.
- `offset` is the distance offset of the signal ct_0 with respect to the reference position.
- `pulse-length` is the pulse duration of the signal in length units $c\tau$. The pulse duration is defined as the interval in which the pulse intensity is larger than half the maximum intensity of the pulse, i.e. FWHM of the intensity.
- `wavelength` is the modulation wavelength λ_0 of the modulated signal.
- `rising-cycles` is the number of cycles that it takes for the flat-top and inverse-Gaussian pulses to reach to the signal value in the corresponding time interval, i.e. N in (4.4) and (**??**). Default value is two cycles.
- `CEP` is the carrier envelope phase ϕ_{CEP} of the modulated signal in degrees.
- `sigma-inverse-gaussian` is a vector of two real numbers containing the values of parameters σ_1 and σ_2 for the inverse-Gaussian pulse in (**??**).

2. `field-sampling:` This group defines the required parameters for saving the field value at specific points with time. There are different parameters required for this definition which include:

- `sample` is a boolean value determining if the field sampling should be activated.
- `type` determines if the field should be sampled at the given points (`at-point`) or the field should be sampled at the points over a line (`over-line`).
- `field` determines which electromagnetic field is to be sampled. The available options are the electric field, magnetic field, magnetic vector potential, scalar electric potential, charge and current. This item can be repeated to assign several fields for the sampling. In the text file, the fields appear in columns with the same order as given in this group.
- `base-name` is the file name (no suffix) with the required address of the file to save the output data.
- `directory` is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.

- *rhythm* is a real value determining the rhythm of field sampling, i.e. the time interval between two consecutive sampling times.
- *position* is the coordinate of the points where the fields should be sampled. By repeating this line any number of points can be added to the set of sampling locations. This option is merely used when the sampling type is set to *at-point*.
- *line-begin* defines the position of the line begin over which the fields should be sampled and is used when the sampling type is set to *over-line*.
- *line-end* defines the position of the line end over which the fields should be sampled and is used when the sampling type is set to *over-line*.
- *number-of-points* is the number of points between line-begin and line-end for field-sampling. This value is used when the sampling type is set to *over-line*.

3. *field-visualization:* This group defines the required parameters for visualizing the fields in the whole computational domain. The output will be a set of `.vtu` files at each time for each processor which are connected with a set of `.pvtu` files. They can be very nicely visualized using the open source ParaView package. To enable various visualizations, this group can be repeated in the job file. There are different parameters required for this definition which include:
   - *sample* is a boolean value determining if the field visualization should be activated.
   - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
   - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
   - *type* is the type of visualization and mainly decides if the visualization is 2D (*in-plane*) or 3D (*all-domain*).
   - *plane* is the plane of field-visualization if the visualization is to be done over a 2D plane. This parameter is read only if *type* is set to *in-plane*.
   - *rhythm* is the rhythm of field visualization, i.e. the time interval between two consecutive visualization instants.
   - *field* determines which electromagnetic field is to be visualized. The available options are the electric field, magnetic field, magnetic vector potential, scalar electric potential, charge and current. This item can be repeated to assign several fields for the sampling. In the vtk file, the fields appear with the same order. To make the output compatible with the visualizer ParaView, it is most suitable if three consistent parameters are given here.

4. *field-profile:* This group defines the required parameters for saving a histogram of the field over the whole computational domain. It means that at a specific time instant the field values and the corresponding positions at all the grid points will be written and saved in a text file. The parameters entered by the user for saving the histogram include:
   - *sample* is a boolean value determining if the writing of the histogram during the FDTD simulations should be activated.
   - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
   - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
   - *time* is the time instant for saving the histogram. If this needs to be done in several time instants, simply this line should be repeated with different time values.
   - *rhythm* is the rhythm of field profiling, i.e. the time interval between two consecutive profiling times. Both rhythmic profiling and saving the fields at specific times can be given to the software.
   - *field* determines which electromagnetic field is to be profiled. The available options are the electric field, magnetic field, magnetic vector potential, scalar electric potential, charge and current. This item can be repeated to assign several fields for the sampling. In the text file, the fields appear with the same order.

The format of the *FIELD* group is (The repeatable variables are shown in red):

```
FIELD
{
    field-initialization
    {
        type                        = < plane-wave | truncated-plane-wave | gaussian-beam |
                                        super-gaussian-beam >
        position                    = < ( real, real, real ) >
        direction                   = < ( real, real, real ) >
        polarization                = < ( real, real, real ) >
        radius-parallel             = < real >
        radius-perpendicular        = < real >
        order-parallel              = < int >
        order-perpendicular         = < int >
        signal-type                 = < neumann | gaussian | secant-hyperbolic | flat-top >
        strength-parameter          = < real >
        offset                      = < real >
        pulse-length                = < real >
        wavelength                  = < real >
        rising-cycles               = < int >
        CEP                         = < real >
        sigma-inverse-gaussian      = < ( real, real ) >
    }

    field-sampling
    {
        sample                      = < true | false >
        type                        = < over-line | at-point >
        field                       = < Ex | Ey | Ez | Bx | By | Bz | Ax | Ay | Az | F >
        directory                   = < /path/to/location >
        base-name                   = < string >
        rhythm                      = < real >
        position                    = < ( real, real, real ) >
        line-begin                  = < ( real, real, real ) >
        line-end                    = < ( real, real, real ) >
        number-of-points            = < int >
    }

    field-visualization
    {
        sample                      = < true | false >
        type                        = < in-plane | all-domain >
        plane                       = < xy | yz | xz >
        position                    = < ( real, real, real ) >
        field                       = < Ex | Ey | Ez | Bx | By | Bz | Ax | Ay | Az | F >
        directory                   = < /path/to/location >
        base-name                   = < string >
        rhythm                      = < real >
    }

    field-profile
    {
        sample                      = < true | false >
        field                       = < Ex | Ey | Ez | Bx | By | Bz | Ax | Ay | Az | F >
        directory                   = < /path/to/location >
        base-name                   = < string >
        rhythm                      = < real >
        time                        = < real >
    }
```

}

An example of the field category definition looks as the following:

```
FIELD
{
    field-initialization
    {
        type                        = gaussian-beam
        position                    = ( 0.0, 0.0, -2500.0)
        direction                   = ( 0.0, 0.0, 1.0)
        polarization                = ( 0.0, 1.0, 0.0)
        radius-parallel             = 0.5
        radius-perpendicular        = 0.5
        strength-parameter          = 0.0
        signal-type                 = gaussian
        offset                      = 0.00
        pulse-length                = 1.00
        wavelength                  = 0.0
        rising-cycles               = 2
        CEP                         = 0.0
    }

    field-sampling
    {
        sample                      = true
        type                        = at-point
        field                       = Ex
        field                       = Ey
        field                       = Ez
        directory                   = ./
        base-name                   = field-sampling/field
        rhythm                      = 3.2
        position                    = (0.0, 0.0, 110.0)
    }

    field-visualization
    {
        sample                      = true
        field                       = Ex
        field                       = Ey
        field                       = Ez
        field                       = Q
        directory                   = ./
        base-name                   = field-visualization/field
        rhythm                      = 32
    }

    field-profile
    {
        sample                      = false
        field                       = Ex
        field                       = Ey
        field                       = Ez
        directory                   = ./
        base-name                   = field-profile/field
        rhythm                      = 80
    }
}
```

4.4 UNDULATOR

In section UNDULATOR, the properties of the undulator considered in the FEL problem are introduced. The parameters for establishing undulator fields are obtained in various groups. These groups contain the already implemented undulator types and get updated with time. By adding additional groups, the fileds of these undulators are superposed. Note that the reference undulator for initializing bunches or setting the electron rest frame is the first undulator given in the list.

1. *static-undulator:* This group mainly determines the parameters for defining a static undulator. The set of values accepted in this group include:
   - *undulator-parameter* is the undulator parameter of the undulator, i.e. the so-called K parameter.
   - *period* is the period of the undulator in the given length-scale determined in the mesh class.
   - *length* is an integer returning the total length of the undulator in one period scale. In other words, it determines the number of undulator periods in the module.
   - *polarization-angle* is the angle between the magnetic field polarization and the x-axis in degrees.
   - *offset* determines the point where the beginning of undulator resides. For the first undulator, it is automatically set to zero.
   - *distance-to-bunch-head* determines the distance between the ***first*** undulator entrance and the bunch head at the initialization time. This distance is needed to avoid particles experiencing a sudden change in the undulator field. The value of this parameter is by default two undulator periods. In MITHRA, the radiation of the particles are calculated after they pass through this point. Therefore, this value has a different effect compared to the *initial-time-back-shift* parameter that only shifts the particle in time.
2. *static-undulator-array:* This group determines the required parameters for defining an array of static undulators. The set of values accepted in this group include:
   - *undulator-parameter* is the undulator parameter of the undulators, i.e. the so-called K parameter. For a non-zero tapering parameter, this value corresponds to the K-parameter of the first undulator.
   - *period* is the period of the undulators in the given length-scale determined in the mesh class.
   - *length* is an integer returning the total length of the undulator in one period scale. In other words, it determines the number of undulator periods in the module.
   - *polarization-angle* is the angle between the magnetic field polarization and the x-axis in degrees.
   - *distance-to-bunch-head* is the initial distance between the head of the bunch and the beginning of the undulator. By default, a distance of two undulator periods is considered in MITHRA.
   - *gap* determines the gap between the adjacent undulators.
   - *number* is the total number of undulator modules in the array.
   - *tapering-parameter* is the tapering parameter of the undulator array, i.e. δK in $K_i = K_0 + i\delta K$ giving the K parameters of the i'th undulator module.
   - *distance-to-bunch-head* determines the distance between the ***first*** undulator entrance and the bunch head at the initialization time. This distance is needed to avoid particles experiencing a sudden change in the undulator field. The value of this parameter is by default two undulator periods. In MITHRA, the radiation of the particles are calculated after they pass through this point. Therefore, this value has a different effect compared to the *initial-time-back-shift* parameter that only shifts the particle in time.
3. *optical-undulator:* This group mainly determines the parameters for defining an optical undulator. The set of values accepted in this group include:
   - *beam-type* is the type of the pulse for an optical undulator. The accepted excitation types in MITHRA include plane wave, truncated plane-wave, Gaussian beam, and super-Gaussian beam. A truncated plane-wave is a plane-wave that introduces fields to the particle only over an ellipse determined by beam radii. In addition, for each beam type an equivalent standing wave type can be defined, which represents superposition of two beams with same properties propagating counter to each other.

- *position* is the reference position of the undulator. It is the reference position of the plane-wave propagation in the plane-wave undulator and the focusing point in the Gaussian beam undulator.
- *direction* is the propagation direction of the optical undulator in the plane-wave and Gaussian beam types.
- *polarization* is the polarization of the undulator and is used by both plane-wave and Gaussian beam types.
- *radius-parallel* is the Rayleigh radius (beam waist) of the Gaussian beam in the direction parallel to the polarization. For the truncated plane-wave it is the radius of the ellipse along the polarization direction confining the plane-wave.
- *radius-perpendicular* is the Rayleigh radius (beam waist) of the Gaussian beam in the direction perpendicular to the polarization. For the truncated plane-wave it is the radius of the ellipse perpendicular to the polarization direction confining the plane-wave.
- *order-parallel* is the order of the super Gaussian beam along the field polarization.
- *order-perpendicular* is the order of the super Gaussian beam perpendicular to the field polarization.
- *signal-type* determines the time signature of the undulator. The accepted signal types in MITHRA are listed in the field section. Here, the same set of signal can be given as an undulator envelope.
- *strength-parameter* is the normalized amplitude $a_0 = eA_0/m_e c$ of the undulator, which is equivalent to the undulator-parameter in the static case.
- *offset* is the distance offset of the signal ct_0. The user can play with this parameter and also the reference position to control the arrival time of the pulse on the bunch. By default, MITHRA considers a distance equal to 10 undulator period between the pulse begin and the head of the bunch.
- *pulse-length* is the pulse duration of the signal in length units $c\tau$. The pulse duration is defined as the interval in which the pulse intensity is larger than half the maximum intensity of the pulse, i.e. FWHM of the intensity.
- *wavelength* is the modulation wavelength λ_0 of the modulated signal. Default value is two cycles.
- *rising-cycles* is the number of cycles that it takes for the flat-top and inverse-Gaussian pulses to reach to the signal value in the corresponding time interval, i.e. N in (4.4) and (**??**). Default value is two cycles.
- *CEP* is the carrier envelope phase ϕ_{CEP} of the modulated signal.
- *sigma-inverse-gaussian* is a vector of two real numbers containing the values of parameters σ_1 and σ_2 for the inverse-Gaussian pulse in (**??**).
- *distance-to-bunch-head* determines the distance between the reference position of the optical undulator and the bunch head at the initialization time. This distance is needed to avoid particles experiencing a sudden change in the undulator field. The value of this parameter is by default five time the undulator periods for optical undulators. In MITHRA, the radiation of the particles are calculated after they pass through this point. Therefore, this value has a different effect compared to the *initial-time-back-shift* parameter that only shifts the particle in time.

The format of the UNDULATOR group is (the read groups are repeatable groups):

```
UNDULATOR
{
    static-undulator
    {
        undulator-parameter       = < real >
        period                    = < real >
        length                    = < int >
        polarization-angle        = < real >
        offset                    = < real >
        distance-to-bunch-head    = < real >
    }

    static-undulator-array
    {
        undulator-parameter       = < real >
        period                    = < real >
        length                    = < int >
        polarization-angle        = < real >
```

```
        gap                         = < real >
        number                      = < int >
        tapering-parameter          = < real >
        distance-to-bunch-head      = < real >
    }

    optical-undulator
    {
        beam-type                   = < plane-wave | standing-plane-wave |
                                        truncated-plane-wave | standing-truncated-plane-wave |
                                        gaussian-beam | standing-gaussian-beam |
                                        super-gaussian-beam | standing-super-gaussian-beam >
        position                    = < ( real, real, real ) >
        direction                   = < ( real, real, real ) >
        polarization                = < ( real, real, real ) >
        radius-parallel             = < real >
        radius-perpendicular        = < real >
        order-parallel              = < int >
        order-perpendicular         = < int >
        signal-type                 = < neumann | gaussian | secant-hyperbolic | flat-top >
        strength-parameter          = < real >
        offset                      = < real >
        pulse-length                = < real >
        wavelength                  = < real >
        rising-cylces               = < int >
        CEP                         = < real >
        sigma-inverse-gaussian      = < ( real, real ) >
        distance-to-bunch-head      = < real >
    }
}
```

As explained before, MITHRA always initializes the bunch outside the undulator. It may be already noticed that there exists no option to determine the beginning of the first static undulator with respect to the bunch, because MITHRA ignores this parameter for the first undulator. This value is automatically set by the solver, to avoid particle initialization inside the undulator. For the optical undulator type, the user should control this effect through the parameter offset. An example of the undulator category definition looks as the following:

```
UNDULATOR
{
    static-undulator
    {
        undulator-parameter         = 1.417
        period                      = 3.0e4
        length                      = 300
        polarization-angle          = 0.0
        offset                      = 0.0
    }
}
```

An instance of optical undulator definition reads as follows:

```
UNDULATOR
{
    optical-undulator
    {
        beam-type                   = plane-wave
        position                    = ( 0.0, 0.0, 0.0 )
        direction                   = ( 0.0, 0.0,-1.0 )
```

```
        polarization              = ( 0.0, 1.0, 0.0 )
        strength-parameter        = 0.5
        signal-type               = flat-top
        wavelength                = 1.0e3
        pulse-length              = 1200.0e3
        offset                    = 600.e3
        rising-cycles             = 2
        CEP                       = 0.0
  }
}
```

4.5 EXTERNAL-FIELD

The parameters for defining fields of additional components during the wiggling process are given to the solver in this section. This part of the solver will be updated depending on the projects where MITHRA is used for modelling the interaction. Put differently, the components used in the project over the undulator section in each project will be implemented in this section. The current version of MITHRA accepts the following set of external fields:

1. *electromagnetic-wave:* This group mainly determines the parameters for superposing the field of an electromagnetic beam over the undulator section. This external field in principle fulfills the same thing as a general seed in the FEL interaction. However, if the seed is defined as an external plane-wave, the output radiation does not include the field of seeded beam. In other words, it starts from a zero initial radiation. This group can be repeated to superpose a number of plane waves over each other during the interaction. The set of values accepted in this group include:

   - *beam-type* is the type of the beam. The accepted excitation types in MITHRA include plane wave, truncated plane-wave, Gaussian beam, and super-Gaussian beam. A truncated plane-wave is a plane-wave that introduces fields to the particle only over an ellipse determined by beam radii. In addition, for each beam type an equivalent standing wave type can be defined, which represents superposition of two beams with same properties propagating counter to each other.
   - *position* is the reference position of the excitation. It is the reference position of the plane-wave propagation in the plane-wave excitation and the focusing point in the Gaussian beam excitation.
   - *direction* is the propagation direction of the excitation in the plane-wave and Gaussian beam types.
   - *polarization* is the polarization of the incoming excitation and is used by both plane-wave and Gaussian beam types.
   - *radius-parallel* is the Rayleigh radius (beam waist) of the Gaussian beam in the direction parallel to the polarization. For the confined plane-wave, it is the radius of the ellipse along the polarization direction confining the plane-wave.
   - *radius-perpendicular* is the Rayleigh radius (beam waist) of the Gaussian beam in the direction perpendicular to the polarization. For the confined plane-wave, it is the radius of the ellipse perpendicular to the polarization direction confining the plane-wave.
   - *order-parallel* is the order of the super Gaussian beam along the field polarization.
   - *order-perpendicular* is the order of the super Gaussian beam perpendicular to the field polarization.
   - *signal-type* determines the time signature of the signal exciting the fields according to the particular type. The accepted signal types in MITHRA include modulated Neumann, modulated Gaussian, modulated secant hyperbolic, sinusoidal (flat-top), and the inverse-Gaussian pulse. The equations representing the time domain variation of each pulse are as follows:

     (a) *neumann:*
     $$f(t) = -A_0 4 \ln 2 \ \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) \frac{t-t_0}{\tau^2} e^{-2\ln 2\,(t-t_0)^2/\tau^2} \tag{4.6}$$

     (b) *gaussian:*
     $$f(t) = A_0 \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) e^{-2\ln 2\,(t-t_0)^2/\tau^2} \tag{4.7}$$

     (c) *secant-hyperbolic:*
     $$f(t) = A_0 \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) \frac{1}{\cosh((t-t_0)/\tau)} \tag{4.8}$$

(d) *flat-top:*

$$f(t) = A_0 \cos(2\pi f(t-t_0) + \phi_{\mathrm{CEP}}) \begin{cases} e^{-(t-t_0-\tau/2)^2/(N/f)^2} & \|t-t_0\| \geq \tau/2 \\ 1.0 & \|t-t_0\| \leq \tau/2 \end{cases} \tag{4.9}$$

(e) *inverse-gaussian:*

$$f(t) = A_0 \cos(2\pi f(t-t_0)+\phi_{\mathrm{CEP}}) \sqrt[4]{\left(1+\frac{t^2}{\sigma_1^2}\right)\left(1+\frac{t^2}{\sigma_2^2}\right)} \begin{cases} e^{-(t-t_0-\tau/2)^2/(N/f)^2} & \|t-t_0\| \geq \tau/2 \\ 1.0 & \|t-t_0\| \leq \tau/2 \end{cases} \tag{4.10}$$

- *strength-parameter* is the normalized amplitude $a_0 = eA_0/m_e c$ of the beam.
- *offset* is the distance offset of the signal ct_0 with respect to the reference position.
- *pulse-length* is the pulse duration of the signal in length units $c\tau$. The pulse duration is defined as the interval in which the pulse intensity is larger than half the maximum intensity of the pulse, i.e. FWHM of the intensity.
- *wavelength* is the modulation wavelength λ_0 of the modulated signal.
- *rising-cycles* is the number of cycles that it takes for the flat-top and inverse-Gaussian pulses to reach to the signal value in the corresponding time interval, i.e. N in (4.9) and (4.10). Default value is two cycles.
- *CEP* is the carrier envelope phase ϕ_{CEP} of the modulated signal in degrees.
- *sigma-inverse-gaussian* is a vector of two real numbers containing the values of parameters σ_1 and σ_2 for the inverse-Gaussian pulse in (4.10).

The format of the FIELD group is:

```
EXTERNAL-FIELD
{
    electromagnetic-wave
    {
        beam-type                = < plane-wave | standing-plane-wave |
                                     truncated-plane-wave | standing-truncated-plane-wave |
                                     gaussian-beam | standing-gaussian-beam |
                                     super-gaussian-beam | standing-super-gaussian-beam >
        position                 = < ( real, real, real ) >
        direction                = < ( real, real, real ) >
        polarization             = < ( real, real, real ) >
        radius-parallel          = < real >
        radius-perpendicular     = < real >
        order-parallel           = < int >
        order-perpendicular      = < int >
        signal-type              = < neumann | gaussian | secant-hyperbolic | flat-top >
        strength-parameter       = < real >
        offset                   = < real >
        pulse-length             = < real >
        wavelength               = < real >
        rising-cycles            = < int >
        CEP                      = < real >
        sigma-inverse-gaussian   = < ( real, real ) >
    }
}
```

An example of the external field definition looks as the following:

```
EXTERNAL-FIELD
{
    electromagnetic-wave
    {
```

```
        beam-type                = plane-wave
        position                 = ( 0.0, 0.0, 0.0)
        direction                = ( 0.0, 1.0, 0.0)
        polarization             = ( 0.0, 0.0, 1.0)
        strength-parameter       = 1.0
        signal-type              = flat-top
        offset                   = 0.00
        pulse-length             = 1.00
        wavelength               = 0.0
        rising-cycles            = 2
        CEP                      = 0.0
    }

    electromagnetic-wave
    {
        beam-type                = plane-wave
        position                 = ( 0.0, 0.0, 0.0)
        direction                = ( 0.0,-1.0, 0.0)
        polarization             = ( 0.0, 0.0, 1.0)
        strength-parameter       = 1.0
        signal-type              = flat-top
        offset                   = 0.00
        pulse-length             = 1.00
        wavelength               = 0.0
        rising-cycles            = 2
        CEP                      = 0.0
    }
}
```

4.6 FEL-OUTPUT

The typical parameters for a free electron laser instrument are calculated from the radiated fields using the parameter definitions at this section. Currently, there are three groups implemented in MITHRA, for computation of radiated power, visualization of this power on a detector in front of the bunch, and visualization of the bunch profile in the lab frame by recording the particles that are intercepted by screens placed along the z-axis. In what follows, the parameters in each group are introduced:

1. *radiation-power:* This group mainly determines the parameters whose values are needed for calculating the radiation power from the field distribution. The output is a .txt file with the first column the time point and the second column the radiated power at the given point. For multiple points the column is repeated. If multiple frequencies are given, there will be several rows with similar time value listing the radiated power with different wavelengths. This group can be repeated to obtain various files for different output definitions. The set of values accepted in this group include:

    - *sample* is a boolean parameter which activates the computation of the total radiated power at each instant.
    - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
    - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
    - *type* determines if the radiated power should be sampled at given distances from the bunch (*at-point*) or should be sampled at the points over a line (*over-line*).
    - *plane-position* gives the total distance from the bunch where a sampling plate to capture the whole radiated power will be placed. The sampling plate should exist inside the computational domain. This may change after far-field transformation is implemented in the code. One can enter several sampling positions by repeating this line. This option is only considered if the type variable is set to *at-point*.

- *line-begin* defines the distance from the bunch center for the line begin over which the fields should be sampled and is used when the sampling type is set to *over-line* option. In case of multiple bunches, the center of the first bunch is considered as the reference position.
- *line-end* defines the distance from the bunch center for the line end over which the fields should be sampled and is used when the sampling type is set to *over-line* option. In case of multiple bunches, the center of the first bunch is considered as the reference position.
- *number-of-points* is the number of points between line-begin and line-end for the computation of radiation. This value is used when the type variable is set to *over-line*.
- *normalized-frequency* is the central frequency normalized to the radiation frequency of the radiation spectrum.
- *minimum-normalized-frequency* is the minimum frequency normalized to the radiation frequency. This parameters and the next two parameters are used to sweep over the normalized frequency and save the spectrum of the total radiated power.
- *maximum-normalized-frequency* is the maximum frequency normalized to the radiation frequency.
- *number-of-frequency-points* is the number of frequency points between the minimum and maximum normalized frequencies for the computation of radiation spectrum.

2. *power-visualization:* This group is designed to visualize the radiated power on a detector in front of the bunch. This detector needs to reside insider the computational grid, since there is no far-field transformation implemented in the code. The output in this group will be a set of 2D `.vtk` files that can be read by a visualization software like Paraview. The set of values accepted in this group include:

   - *sample* is a boolean parameter which activates the visualization of power at a detector-plane in front of the bunch.
   - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
   - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
   - *plane-position* gives the total distance from the bunch where a visualization plane for the local radiated power will be placed. The visualization plane should exist inside the computational domain. This may change after far-field transformation is implemented in the code.
   - *normalized-frequency* is the central frequency normalized to the radiation frequency of the radiation spectrum.
   - *rhythm* is a real value determining the intervals for power-visualization. Note that because of the requirements on the Fourier transform, the power computation is accomplished at each time step. However, the saving of the visualization files will be done according to the set value for this parameter.

3. *bunch-profile-lab-frame:* This group determines the z-coordinates where the diagnostics screens are placed. Particles that are intercepted by these screens will have their momentum, transverse position, and time of intersection stored in a .txt file. Each line of the file is one particle, and the columns are $q, x, y, t, p_x, p_y, p_z$ respectively. The set of values accepted in this group include:

   - *sample* is a boolean parameter which activates the computation of the total radiated power at each instant.
   - *base-name* is the file name (no suffix) with the required address of the file to save the output data.
   - *directory* is the address where the above file should be saved. The file name with the address can also be given in the base-name section. The software eventually considers the combination of directory and base-name as the final complete file name.
   - *position* gives the distance from the undulator begin at which to place the diagnostics screen. A negative position will mean that the screen is placed before the undulator.
   - *rhythm* is the rhythm in position with respect to the undulator begin for initializing the screens. The sequence of screens starts at the undulator begin.

The format of the `FEL-OUTPUT` group is (the red groups and parameters are repeatable parts):

```
FEL-OUTPUT
{
    radiation-power
    {
        sample                      = < false | true >
        type                        = < at-point | over-line >
        directory                   = < /path/to/location >
        base-name                   = < string >
        plane-position              = < real >
        line-begin                  = < real >
        line-end                    = < real >
        number-of-points            = < int >
        normalized-frequency        = < real >
        minimum-normalized-frequency      = < real >
        maximum-normalized-frequency      = < real >
        number-of-frequency-points        = < int >
    }

    power-visualization
    {
        sample                      = < false | true >
        directory                   = < /path/to/location >
        base-name                   = < string >
        plane-position              = < real >
        normalized-frequency        = < real >
        rhythm                      = < real >
    }

    bunch-profile-lab-frame
    {
        sample                      = < false | true >
        directory                   = < /path/to/location >
        base-name                   = < string >
        position                    = < real >
        rhythm                      = < real >
    }
}
```

An example of the FEL output category definition looks as the following:

```
FEL-OUTPUT
{
    radiation-power
    {
        sample                      = true
        type                        = at-point
        directory                   = ./
        base-name                   = power-sampling/power
        plane-position              = 110.0
        normalized-frequency        = 1.00
    }

    power-visualization
    {
        sample                      = true
        directory                   = ./
        base-name                   = power-visualization/power
        plane-position              = 110.0
```

```
        normalized-frequency      = 1.00
        rhythm                    = 32.0
    }

    bunch-profile-lab-frame
    {
        sample                    = true
        directory                 = ./
        base-name                 = bunch-profile-lab-frame/profile
        position                  = -0.1e6
        position                  = 2.5e6
        position                  = 9.0e6
    }
}
```

Chapter 5

Examples

The goal in this chapter is to present several examples for the MITHRA users to more easily get familiar with the interface of the software. In addition, through the presented examples the pros and cons of using the developed FDTD/PIC algorithm are more accurately evaluated. For example, the computation time, numerical stability and numerical convergence and more importantly the reliability of results are studied based on some standard examples. The software developers aim to update this chapter with the most recent examples where MITHRA is used for the FEL simulation. The job files needed by the MITHRA code for the examples provided in this chapter are all available in the github repository under the link https://github.com/aryafallahi/mithra. Additionally, in the appendix A of this manual, some of the main files are also presented for an interested reader.

5.1 Example 1: Infrared FEL

5.1.1 Problem Definition

As the first example, we consider an infrared FEL with the parameters tabulated in table 5.1.1, which is inspired by the numerical analysis presented in [10]. The bunch distribution is assumed to be uniform in order to compare the results with one-dimensional FEL theory. For the same purpose, the transverse energy spread is considered to be zero and a minimal longitudinal energy spread is assumed. In this first example, saturation of the FEL gain is obtained after a small number of microbunches compared to a typical x-ray FEL, which leads to a short simulation time. As a result, we use this problem to assess the simulation results, verify the convergence and reliability of the algorithm, and finally compare the output with well-established softwares in the community.

To simulate the considered FEL configuration, a job file is written and given to the software to analyze the interaction and produce the results shown in Fig. 5.1 [1]. As observed in the mesh definition, the transverse size of the computational domain is almost 10 times larger than the bunch transverse size. In the contrary, the longitudinal size of the mesh is only three times larger than the bunch length. This needs to be considered due to the failure of absorbing boundary conditions for the oblique incidence of the field. During the simulation, the code adds tapering sections to both bunch and undulator to avoid abrupt transitions which produce coherent scattering emission (CSE). To consider the additional tapering sections, the undulator begin is initialized at least ten radiation wavelengths apart from the bunch head to reduce the CSE. This also introduces corresponding limitations on the mesh size, meaning that the minimum distance from the bunch tail and the mesh boundary should be at least ten radiation wavelengths. In the illustrated job file, some of the output formats are turned off which can always be activated to obtain the required data.

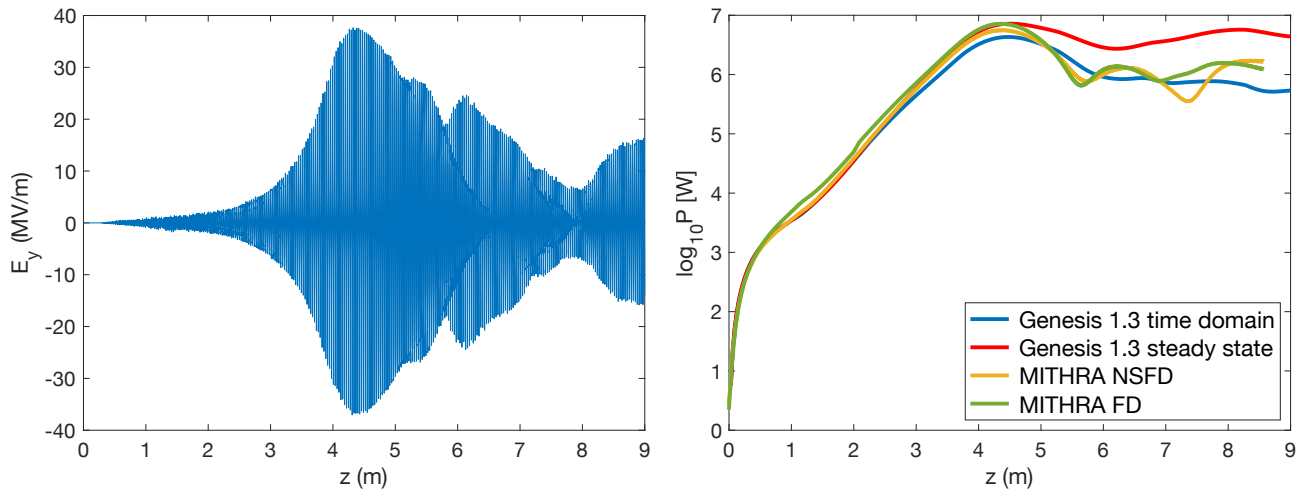
5.1.2 Simulation Results

In the beginning, we neglect the space-charge effect only to achieve a good assessment of MITHRA simulation results. The investigation of space-charge effect will be performed in the second step. Fig. 5.1a shows the transverse electric field sampled at 110 µm in front of the bunch center. The logarithmic plot of the radiated power for different propagation lengths (z) is also depicted in Fig. 5.1b. We comment that the full-wave analysis offered by MITHRA obtains the total radiated field as a superposition of forward, backward and near-field radiation components. In an FEL simulation, one is often interested in the forward radiation

[1] It should be emphasized here that Genesis and MITHRA start the simulation of FEL at different instances. The former considers bunch within the undulator at the start of the simulation, whereas the latter starts the simulation when the bunch is outside the undulator. In the plots presented in this manual, the curves are shifted to have similar gain regimes thereby achieving a valid comparison between the results.

Table 5.1: Parameters of the Infrared FEL configuration considered as the first example.

FEL parameter	Value
Current profile	Uniform
Bunch size	$(260\times260\times100.5)\,\mu$m
Bunch charge	29.5 pC
Bunch energy	51.4 MeV
Bunch current	88.5 A
Longitudinal momentum spread	0.01%
Normalized emittance	0.0
Undulator period	3.0 cm
Magnetic field	0.5 T
Undulator parameter	1.4
Undulator length	5 m
Radiation wavelength	3 μm
Electron density	$2.72\times10^{13}1/\mathrm{cm}^3$
Gain length (1D)	38.8 cm
FEL parameter	0.006
Cooperation length	39.7 μm
Initial bunching factor	0.01



(a) (b)

Figure 5.1: (a) The transverse field E_y at 110 μm distance from the bunch center and (b) the total radiated power calculated at 110 μm distance from the bunch center in terms of the traveled undulator length.

component, which can only be extracted at a distance in front of the radiation source, namely the electron bunch. This is the main reason for illustrating the radiated power and field at 110 µm in front of the bunch center.

According to the 1D FEL theory the gain length of the considered SASE FEL configuration is $L_G = 22.4\,\mathrm{cm}$. The gain length calculated from the slope of the power curve is $L_G = 22\,\mathrm{cm}$. There exists also a good agreement in the computed saturation power. The beam energy according to the data in table 5.1.1 is 1.52 mJ which for the bunch length of 100 µm corresponds to $P_{beam} = 4.55\,\mathrm{GW}$ beam power. The estimated saturation power according to the 1D theory is equal to $P_{sat} = \rho P_{beam} = 27\,\mathrm{MW}$. The saturation power computed by MITHRA is $26\,\mathrm{MW}$.

We have also performed a comparison study between the obtained results from MITHRA and the code Genesis 1.3, which is presented in Fig. 5.1b. As observed, both codes produce similar results in the initial state and the gain regime. Nonetheless, there exists a considerable discrepancy between the calculated radiated power in the saturation regime. The illustrated results in Fig. 5.1b show that the steady state and time domain analyses using Genesis do not produce similar results. This shows that the bunch is not long enough to justify the steady state approximation, and dictates a time domain analysis for accurate simulation. However, the results obtained by MITHRA at saturation do not match with the Genesis results even in the time domain.

The origin of such a discrepancy is described as follows: As explained in chapter 2, Genesis 1.3 and all the existing softwares for FEL simulation neglect the backward radiation of the electrons. Such an approximation is motivated by the inherent interest in forward radiation throughout the FEL process. The backward radiation although is seldom used due to its long wavelength, it influences the motion of electrons, the charge distribution and in turn the FEL output. The influence of low-frequency backward radiation on the performance of free electron lasers has been already studied in a 1D regime [47]. The effect becomes stronger in the saturation regime, where the electron bunch is modulated and the FEL radiation is a strong function of the particles distribution.

Furthermore, in Fig. 5.1b, we compare the results obtained using the NSFD implemented in MITHRA and standard FD scheme. As observed, formulation based on FD predicts slightly higher radiation power compared to NSFD. This effect happens due to the smaller phase velocity of light when wave propagation follows dispersion equation (3.17). The result is slower phase slippage of electron bunch over the radiation and consequently later saturation of the radiation.

As a 3D electromagnetic simulation, it is always beneficial to investigate the electromagnetic field profile in the computational domain. Using the field visualization capability in MITHRA, snapshots of the field profile at different instants and from various view points are provided. In Fig. 5.2, snapshots of the radiated field profile, beam power and bunch profile at different time instants are illustrated. The emergence of lasing radiation at the end of the undulator motion is clearly observed in the field profile. Furthermore, snapshots of the bunch profile are also presented beside the field profile. The main FEL principle which is the lasing due to micro-bunching of the electron bunch is observed from the field and bunch profiles. The first two snapshots evidence a considerable change in the bunch length, which occurs due to the entrance in the undulator. The bunch outside of the undulator with Lorentz factor γ travels faster than the bunch inside the undulator with Lorentz factor $\gamma/\sqrt{1+K^2/2}$. Therefore, after the entrance to the undulator the bunch length becomes shorter. This effect may not be easily observed in real laboratory frame, but is significant in electron rest frame.

5.1.3 Convergence Analysis

The convergence rate of the results is the main factor used to assess a numerical algorithm. In our FEL analysis, there are several parameters introduced by the numerical method which may affect the final result. These parameters include (1) number of macro-particles (n), (2) time step for updating equation of motion (Δt_b), (3) longitudinal mesh size (l_z), (4) transverse mesh size ($l_x = l_y$), (5) longitudinal discretization (Δz) and (6) transverse discretization ($\Delta x = \Delta y$). Studying the convergence of the results is crucial to acquire an estimate for the uncertainty in the reported values. Here, this task is accomplished by sweeping over the above parameters and plotting the error function defined as the following:

$$\mathrm{error} = \frac{\int_{z_i}^{z_f} |P(z) - P_0(z)|\mathrm{dz}}{\int_{z_i}^{z_f} P_0(z)\mathrm{dz}}, \tag{5.1}$$

where z_i and z_f are the beginning and end of the undulator, respectively and P_0 is the reference simulation result which is chosen as the results with the highest resolution.

In Fig. 5.3 the convergence study is shown for the aforementioned parameters. Generally, accuracy of less than 3% is achieved by using the initially suggested values.

5.1.4 Space-charge effect

A promising benefit offered by MITHRA is the assessment of various approximations used in the previously developed FEL codes. As an example, the algorithm used in the TDA method to evaluate the space-charge effect can be examined and verified using this

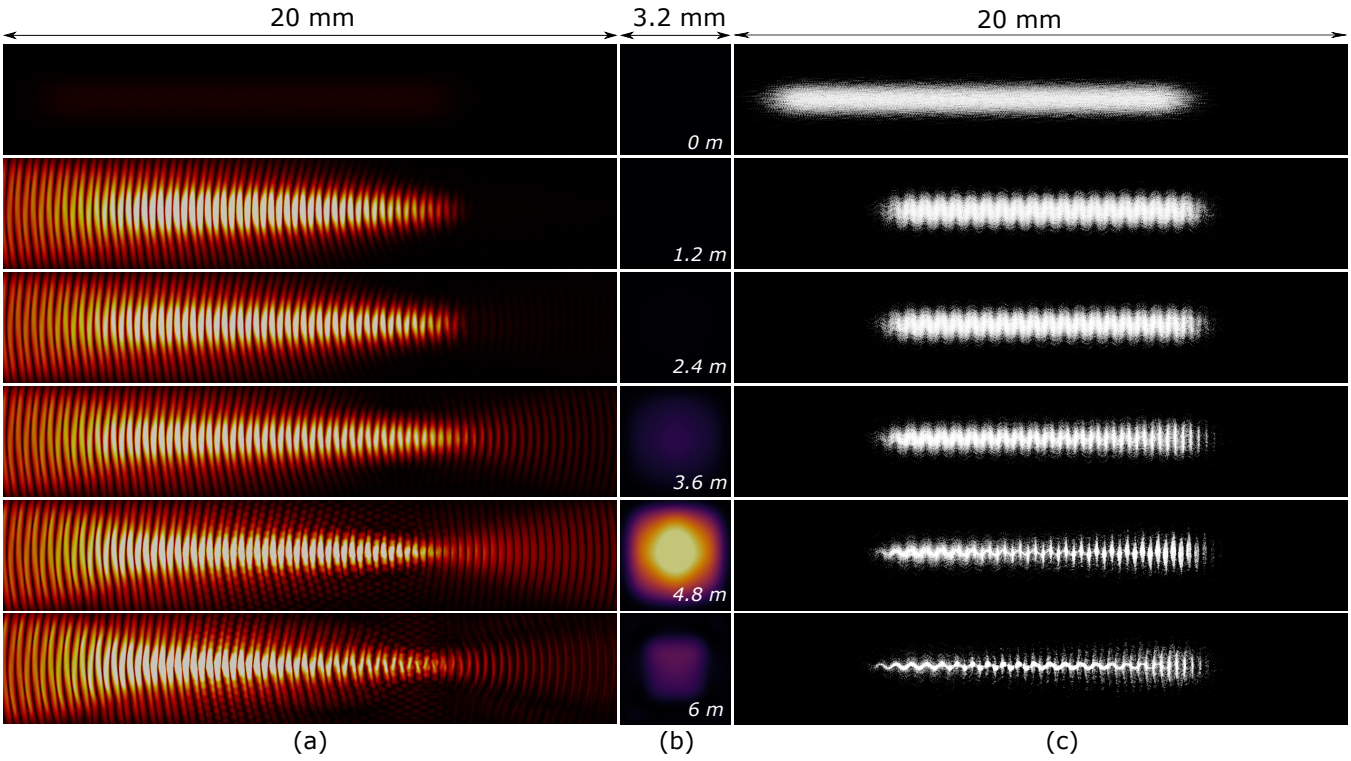


Figure 5.2: (a) Snapshots of the radiated field profile taken at $x = 0$, (b) snapshots of the beam power at $z = 60\,\mu$m plane, and (c) the bunch profile viewed from the x axis.

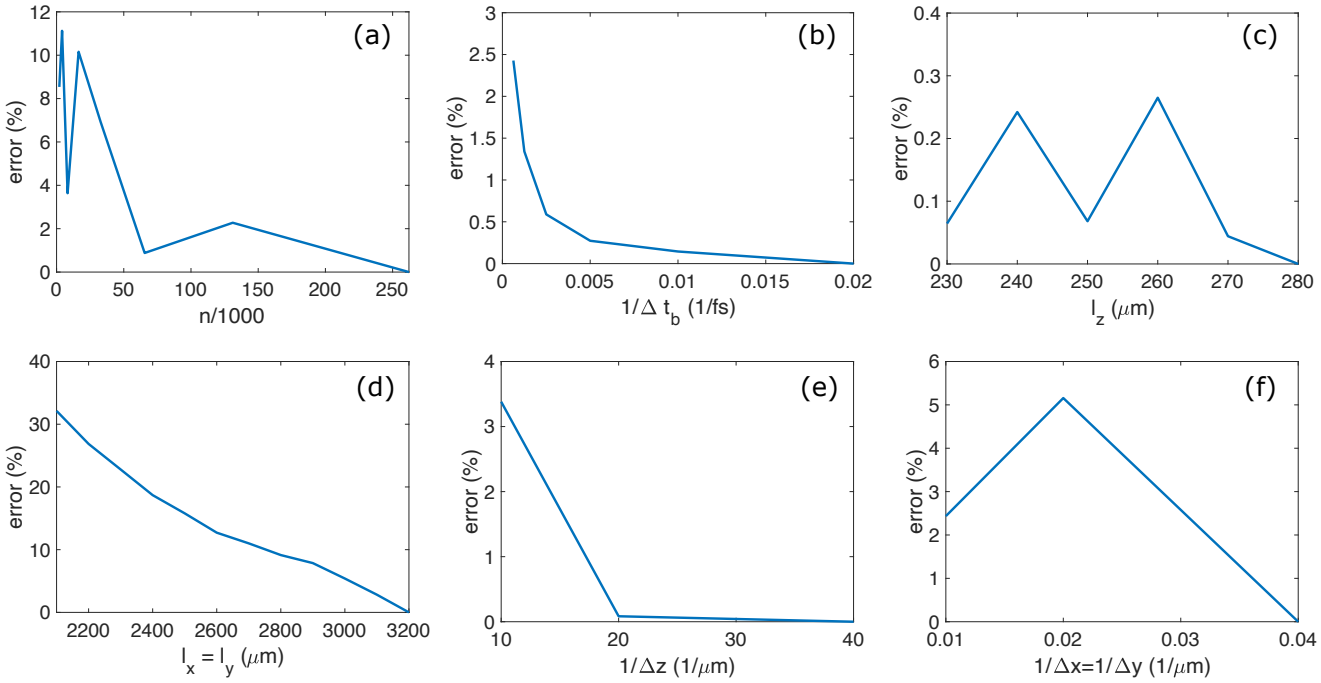


Figure 5.3: Convergence study for the different involved parameters in the considered FEL simulation: (a) n, (b) Δt_b, (c) l_z, (d) $l_x = l_y$, (e) Δz and (f) $\Delta x = \Delta y$

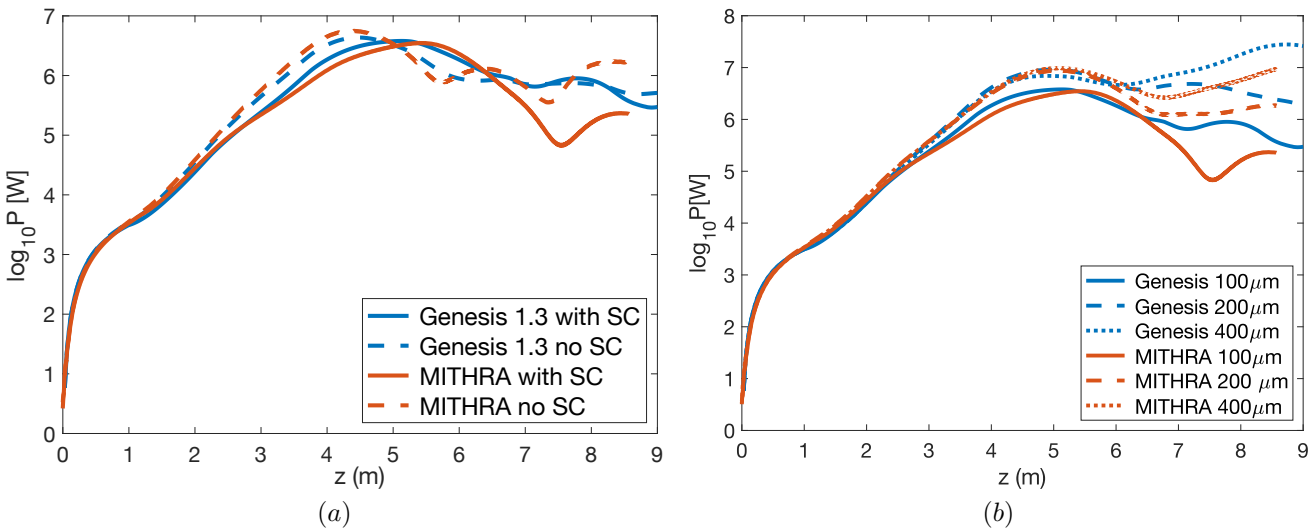


Figure 5.4: The total radiated power calculated at 110 µm distance from the bunch center in terms of the traveled undulator length (a) with and without space-charge consideration and (b) various lengths of the bunch with space-charge assumption.

code. The TDA method implemented in Genesis 1.3 software considers a periodic variation of space-charge force throughout the electron bunch [44, 48]. This assumption is implicitly made, when electric potential equation is solved in a discrete Fourier space. However, a simple investigation of bunch profiles shown in Fig. 5.2c shows that a periodic assumption for the electron distribution may be a crude approximation. In addition, this assumption is favored by the FEL gain process and potentially decreases any detrimental influence of the space-charge fields on the FEL radiation. On the other hand, the algorithm in TDA method considers longitudinal space-charge forces and neglects transverse forces, which is merely valid in high energy electron regimes.

In Fig. 5.4a, we are comparing the solution of the FEL problem using MITHRA and Genesis 1.3 with and without considering the space-charge effect. As observed in the results, the effect of space-charge on the radiation gain predicted by MITHRA is much stronger than the same effect predicted by Genesis. This is attributed to the assumption of periodic variations in the space-charge force made in TDA algorithm. If such a hypothesis is correct, the observed discrepancy should reduce once the radiation from a longer bunch is simulated, because the accuracy of periodicity assumption increases for longer bunches. Indeed, this is observed after repeating the simulation for longer electron bunches with similar charge and current densities. The results of such a study is illustrated in Fig. 5.4b.

5.1.5 Computation performance

A potential user of the code is usually interested in the total computation resources required for a specific FEL simulation. To clarify such features, the study on the computation performance for MPI parallelized code is presented in Fig. 5.5, where the total computation time is depicted in terms of the number of processors. The simulation with 131072 macro-particles, a grid with 11'468'800 cells and 37'500 time steps is taken into account. The code is run on euler cluster of the scientific computing facility at ETH Zürich. It is observed that running on 48 CPUs is optimal for this problem. This number increases for larger and more demanding examples. In case of the run on 48 CPUs, field update on the computational grid, motion update of the bunch macro-particles and the computation of the total radiated field together with the required Fourier transform take 44%, 28%, and 28% of the total computational time, respectively.

5.2 Example 2: Seeded UV FEL

5.2.1 Problem Definition

As the second example, we consider a seeded FEL in the UV regime to verify the implemented features for simulating a seeded mechanism. The parameters of the considered case are taken from [17], which are tabulated in table 5.2.1. The bunch distribution is again assumed to be uniform with a long current profile ($\sim$1000 times the radiation wavelength) in order to compare the results with the steady state simulations. For the same reason, the seed pulse length is considered to be infinitely long, i.e. a continuous wave

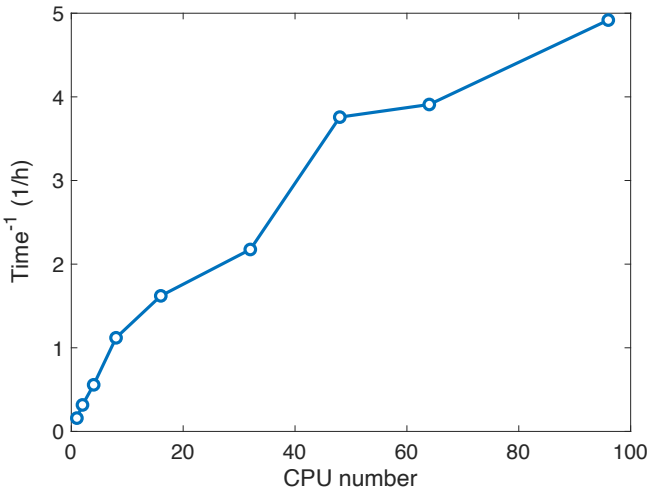


Figure 5.5: Reverse of total computation time versus the total number of processors.

Table 5.2: Parameters of the UV seeded FEL configuration considered as the second example.

FEL parameter	Value
Current profile	Uniform
Bunch size	(95.3×95.3×150) µm
Bunch charge	54.9 pC
Bunch energy	200 MeV
Bunch current	110 A
Longitudinal momentum spread	0.01%
Normalized emittance	0.97 µm-rad
Undulator period	2.8 cm
Magnetic field	0.7 T
Undulator parameter	1.95
Undulator length	15 m
Radiation wavelength	0.265 µm
Electron density	$2.52 \times 10^{14} 1/\mathrm{cm}^3$
Gain length (1D)	66.8 cm
FEL parameter	0.0033
Cooperation length	3.65 µm
Initial bunching factor	0.0
Seed type	Gaussian beam
Seed focal point	70 cm
Seed beam radius	183.74 µm
Seed pulse length	infinite
Seed power	10 kW

pulse. The transverse energy spread is calculated for a bunch with normalized transverse emittance equal to 1 mm-mrad. Because of the very long bunch compared to the previous example, the number of required macro-particles to obtain convergent results is around 40 times larger. Furthermore, the stronger undulator parameter dictates a smaller time step for the simulation of bunch dynamics. Note that MITHRA, takes the bunch step value as an initial guess, it automatically adjusts the value based on the calculated time step for mesh update. To simulate the considered FEL configuration, the job file presented in A.2 is written and given to the software to analyze the interaction.

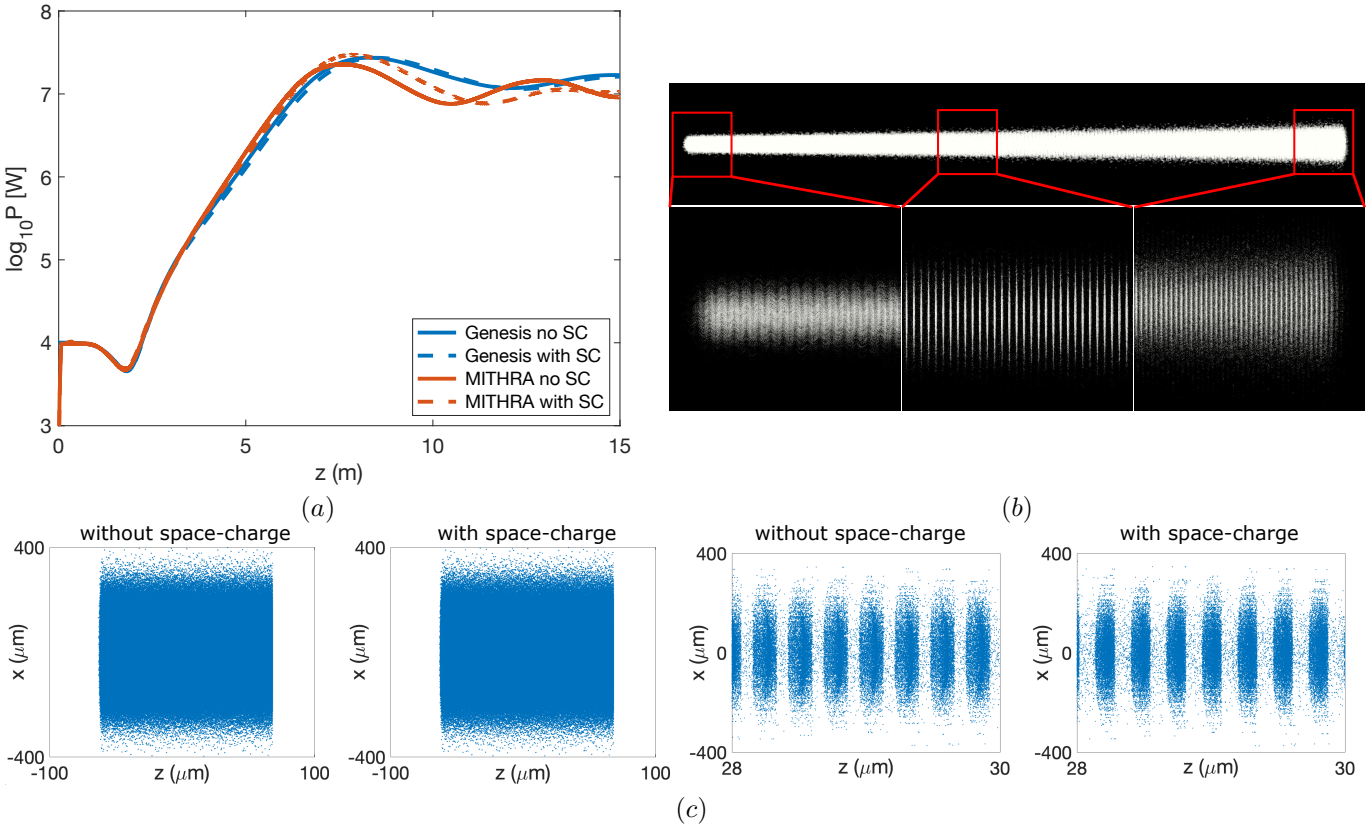


Figure 5.6: (a) The total radiated power measured at 80 µm distance from the bunch center in terms of the traveled undulator length and (b) the bunch profile in the rest frame at 12 m from the undulator begin. (c) Bunch profile and microbunch profiles of the electron beam with and without space-charge considerations are compared.

5.2.2 Simulation Results

Fig. 5.6a shows the radiated power in terms of travelled undulator distance computed using MITHRA and Genesis. As observed again in this example, the results agree very well in the seeded and gain regime, with notable discrepancies in the saturation regime. In Fig. 5.6b, the bunch profile after 12 m propagation in the undulator is also depicted. The micro-bunching of the large bunch is only visible once a zoom into a part of the bunch is considered. The investigation of the results with and without considering space-charge effect shows that in the seeded and gain intervals, space charge plays a negligible role. However, in the saturation regime the effect of space-charge predicted by MITHRA is stronger than the effect predicted by Genesis. By visualizing the bunch in the laboratory frame, one can explore the origin of the small change due to space-charge effect. Fig. 5.6c and 5.6d illustrate this comparison. As observed from these figures, the total bunch profile in both cases are similar, whereas the microbunches in the simulation with space-charge are slightly denser than the case without space-charge consideration.

5.3 Example 3: Optical Undulator

5.3.1 Problem Definition

As explained in the introduction of this manual, one of the milestones considered for the development of MITHRA is full-wave simulation of inverse Compton scattering (ICS) or the so-called optical undulator. The possibility of lasing or the so-called micro-bunching in an electron beam due to an interaction with a counter-propagating laser beam has been under debate for several years. A full-wave analysis of such an interaction definitely gives valuable physical insight to this process. Note that the classical treatment of this interaction within MITHRA does not allow for any consideration of quantum mechanical effects. It is known that the radiation of photons results in a backward force on electrons which leads to a change in their momenta. In the spontaneous radiation regime, the ratio $\rho_1 = \hbar\omega/\gamma mc^2$, representing the amount of quantum recoil due to each photon emission, quantifies this effect. In the FEL gain regime, $\rho_2 = (\hbar\omega/2\rho_{FEL}\gamma mc^2)^2$, with ρ_{FEL} being the FEL parameter, estimates the level of quantum recoil influence on the gain process [49, 50]. The use of classical formulation for optical undulators is only valid if $\rho_1 \ll 1$ and $\rho_2 \ll 1$.

Before embarking on the analysis and interpretation of results for a typical ICS experiment, a benchmark to validate the analysis of optical undulators using FDTD/PIC is presented. It is known that electron trajectories in a static undulator with undulator parameter K and periodicity λ_u are similar to the trajectories in an electromagnetic undulator setup with normalized vector potential $a_0 = K$ and wavelength $\lambda_l = 2\lambda_u$ [51]. We take the first SASE FEL example in table 5.1.1 into account and analyze the same configuration but with an equivalent optical undulator. For this purpose, the undulator definition of example 1 is entered as an electromagnetic undulator with the wavelength and strength parameter obtained from aforementioned relations (see A.3). Fig. 5.7 illustrates a comparison between the radiated infrared light for the static and optical undulator cases. The very close agreement between the two results validates the implementation of optical undulators in MITHRA. The small discrepancies observed are mainly due to the different fringe fields implemented for static undulator and optical undulator with flat-top temporal signature for the signal.

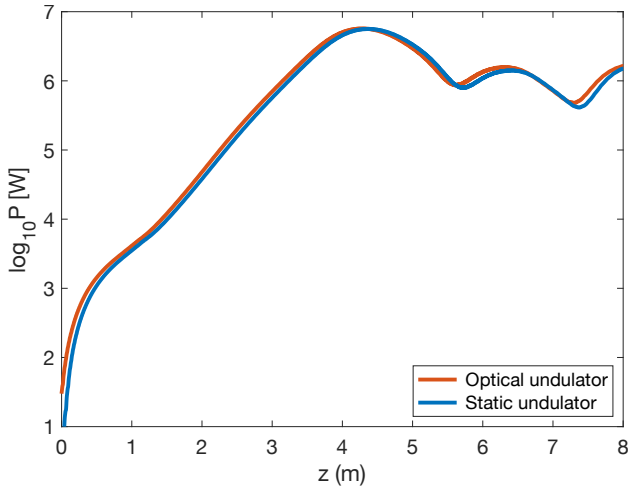


Figure 5.7: The total radiated power calculated at 110 μm distance from the bunch center in terms of the traveled undulator length compared for two cases of an optical and static undulator.

The parameters of FEL interaction in an optical undulator, considered as the third example, are tabulated in table 5.3.1. Since we observe drastic deviation from the predictions of one-dimensional FEL theory in our simulations, we have not listed the FEL parameters calculated using the 1D theory. We believe the discrepancies are originated from the small number of electrons in each 3D wave bucket, i.e. only 2 electrons. This strongly intensifies the 3D effects, dramatically reduces the transverse coherence of the radiation, and indeed makes analysis using 1D FEL theory completely invalid. We comment that for the listed parameters $\rho_1 = 2 \times 10^{-4}$ and $\rho_2 = 0.003$, which are much smaller than errors caused by space-time discretization. In addition, the energy spread and normalized emittance of the electron beam is assumed to be very low to remove the effects of beam divergence on the interaction, thereby easing the interpretation of the simulation outcomes. To simulate the considered FEL configuration, the job file in A.4 is written and given to the software to analyze the interaction.

Table 5.3: Parameters of the FEL configuration with optical undulator considered as the third example.

FEL parameter	Value
Current profile	Uniform
Bunch size	$(60 \times 60 \times 144)$ nm
Bunch charge	0.45 fC
Bunch energy	15 MeV
Bunch current	0.93 A
Longitudinal momentum spread	0.003%
Normalized emittance	0.06 nm-rad
Laser wavelength	1 μm
Laser strength parameter	1.0
Pulse duration	8 ps
Laser pulse type	flat-top
Radiation wavelength	0.41 nm
Electron density	$5.4 \times 10^{18} 1/\text{cm}^3$
Initial bunching factor	0.0

5.3.2 Simulation Results

Fig. 5.8a illustrates the radiation field 82 nm away from the bunch center with and without space-charge. In addition, Fig. 5.8b shows the radiated power in terms of travelled undulator distance computed using MITHRA, illustrating the effect of space charge. It is observed that the gain obtained in this regime is very small compared with typical static undulators, i.e. a factor of ~ 10 when space-charge is neglected and a factor of ~ 7 for a simulation including space-charge effects. To explore the reason for such observation, the micro-bunching of the electron beam is studied. To show that the micro-bunching effect takes place in this regime, the bunching factor of the electron beam in the moving frame is depicted in Fig. 5.8d [2]. The bunching of the electrons due to the ICS interaction is clearly observed in the plot of bunching factor. Here, a question rises; why despite the micro-bunching process no gain in the radiation is observed?

The reason for this effect is the very large shot noise in the bunch because of the low number of particles in each micro-bunch. The strong shot noise causes a strong initial incoherent radiation, which reaches close to the expected saturation power even at the beginning of the interaction. As a matter of fact, the micro-bunching process here increases the coherence of the output radiation rather than power amplification. The investigation of bunching factor throughout the interaction shows that micro-bunching takes place. Nonetheless, the low number of particles in each micro-bunch results in enhancement of micro-bunching only with a factor of ~ 3. According to the depicted power and pulse shape, total number of emitted photons is approximately equal to 4.2×10^3.

To demonstrate the presented hypothesis related to the micro-bunching of bunches with low number of electrons per wavelength bucket, we perform an *unreal* simulation, where each electron is presented by 1000 particles. The thousand particles are distributed evenly throughout each wavelength bucket in order to drastically reduce the shot noise level. In this case, each particle represents a charge 1000 times smaller than the charge of one electron. In addition, we assume an initial bunching factor equal to 0.001 for the input bunch to trigger the FEL gain. In Fig. 5.9, the radiation of such a charge configuration is depicted. The results clearly reveal the radiation start from much lower powers, possibility of achieving the FEL gain and saturating in the same power level as observed with *real* number of particles, thereby confirming the above theory for radiation of low density electron bunches. Consequently, the presented simulation by MITHRA agrees with the already developed FEL principle, according to which low number of electrons per coherence volume prevents achieving the radiation gain, even if the electron bunch is micro-bunched.

Another aspect in this regime of interaction is the generation of strong higher order harmonics, which are depicted up to the third harmonic in Fig. 5.8c. Note that the accuracy of the results decreases for higher harmonics due to the required resolution in the computational mesh.

[2] Currently, bunching factor calculation is not implemented in MITHRA. The user should save the bunch profile using the `bunch-profile` group and subsequently extract the bunching factor from the saved distribution.

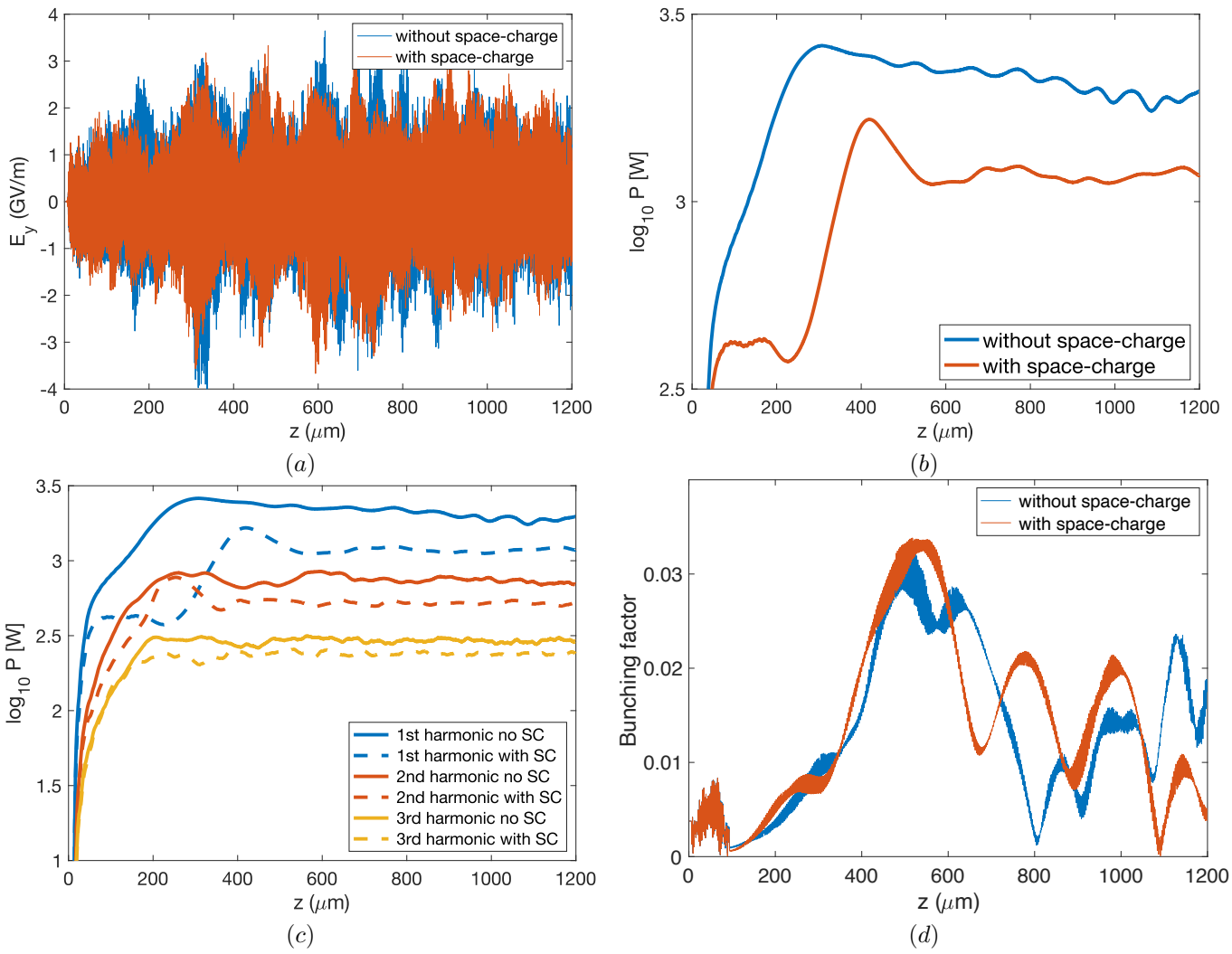


Figure 5.8: (a) Electric field of the generated radiation in front of the bunch, (b) the total radiated power measured at 82 nm distance from the bunch center in terms of the traveled distance, (c) the same radiation power for various harmonic orders, and (d) bunching factor of the considered bunch in the moving frame during the ICS interaction.

5.4 Example 4: Free Propagation

5.4.1 Problem Definition

The fourth example aims at verifying the implementation of space-charge forces in MITHRA. For this purpose, we tackle the problem of free-space propagation for an electron bunch and study the bunch phase-space variations due to space-charge effect. This problem can also be solved using well-established simulation tools in accelerator physics like ASTRA [52]. We take the bunch of the first example, but with Gaussian distribution along the propagation path. The computational domain needs to be slightly larger to account for the Gaussian distribution, and additionally no undulator parameter needs to be parsed to the solver. Transverse emittance of the bunch is assumed to be very small so that the bunch transverse expansion occurs only due to the space-charge effect. The bunch sampling option in MITHRA is activated to save the statistical phase-space data during the propagation. The job file to perform the above simulation in MITHRA is presented in A.5.

5.4.2 Simulation Results

In Fig. 5.10, we show the results for the evolution of transverse bunch size as well as the divergence angle of the beam in root-mean-square (RMS). As observed the bunch size expands with propagation along the undulator due to space-charge forces. This is

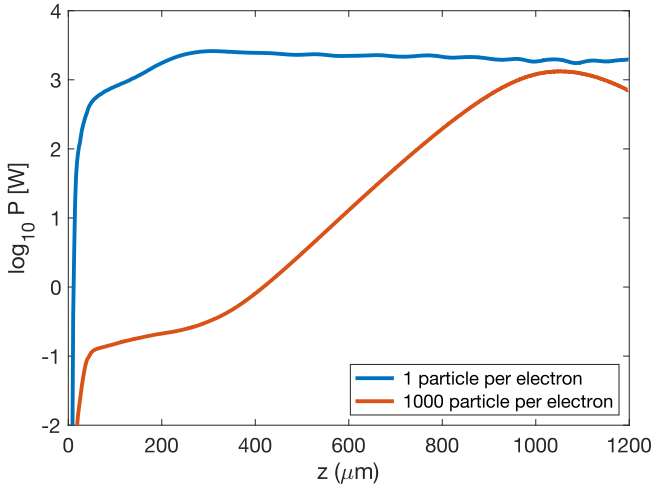


Figure 5.9: The total radiated power measured at 82 nm distance from the bunch center in terms of the traveled distance for an imaginary bunch where each electron is represented by a cloud of 1000 particles.

a confirmation for the considerable space-charge effect encountered in the first example. The results obtained using both MITHRA and ASTRA are depicted and compared against each other. The agreement between the results evidences the reliability of the space-charge implementation in MITHRA.

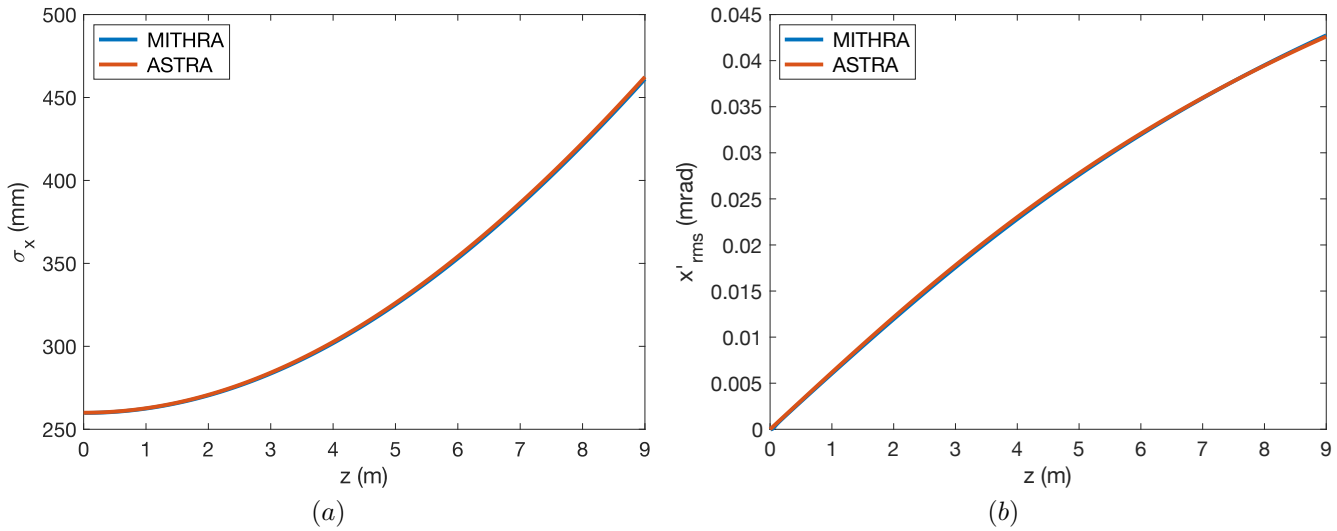


Figure 5.10: (a) Transverse size and (b) rms divergence angle of the electron beam expanding due to space-charge forces after free propagation.

5.5 Example 5: Short Pulse Hard X-ray Source

5.5.1 Problem Definition

In the fifth example, simulation of a problem with parameter sets corresponding to the short pulse regime of the hard X-ray FEL source in the LCLS facility is pursued. The parameters considered in this example are tabulated in table 5.5.1. To simulate the described FEL, the job file of A.6 needs to be parsed in MITHRA.

Table 5.4: Parameters of the hard X-ray FEL configuration considered as the fifth example.

FEL parameter	Value
Current profile	Uniform
Bunch size	$(30.0 \times 30.0 \times 0.8)\ \mu$m
Bunch charge	20.0 pC
Bunch energy	6.7 GeV
Bunch current	7.5 kA
Longitudinal momentum spread	0.1%
Normalized emittance	0.2 μm-rad
Undulator period	3.0 cm
Undulator parameter	3.5
Undulator length	75 m
Radiation wavelength	0.62 nm
Gain length (1D)	1.59 m
FEL parameter	0.0015
Cooperation length	19.3 nm
Shot-noise	true

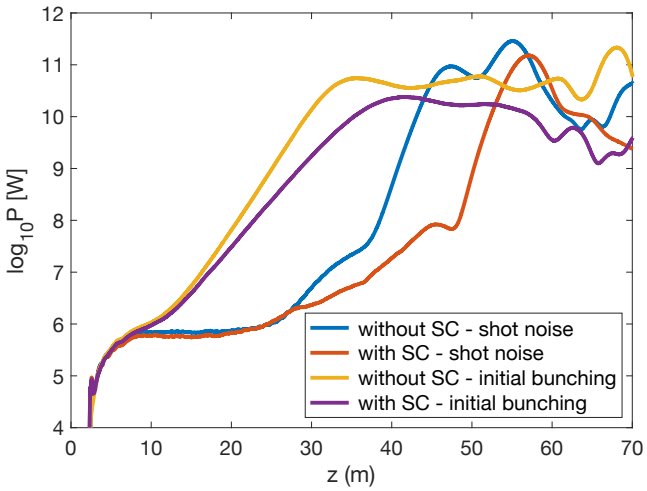


Figure 5.11: Total radiated power measured at 450 µm distance from the bunch center in terms of the traveled undulator length for the hard X-ray FEL source as the third example.

5.5.2 Simulation Results

Fig. 5.11 shows the computed radiated power in terms of traveled undulator distance with and without consideration of space-charge effects. In this figure, the two cases including start of radiation from shot-noise and an initial bunching factor of 0.001 are compared against each other. It is seen that the initial bunching factor leads to a faster saturation of the radiation. According to the 1D FEL theory, the FEL gain length for this example is around 0.92 m, which predicts saturation after around 18 m of undulator length. However, due to 3D effects this saturation length is longer than the predictions of 1D FEL theory. Here, saturation length of about 32 m is observed for a space-charge free simulation. In addition, the space-charge effect seems to be considerable after 10 m of undulator propagation, which contradicts with the typical assumptions that such effects are negligible for multi-GeV beams. This large space-charge effect, not observed in the previous examples, is occurring due to the very short bunch length, which intensifies the Coulomb repulsion forces at the head and tail of the bunch. A rough estimate of the Coulomb field leads to 1 V/m electric field, which in 10 meters of free propagation adds a displacement about 8 nm to the relativistic electrons. This value being ten times larger than the radiation wavelength confirms the strong effect of space-charge forces.

Chapter 6

Reference Card

In the following, a general format for the input file of MITHRA is presented. The red icons or groups can be repeated in the text. *int* stands for an integer number, *real* represents a real value, and *string* denotes a string of characters. The reference directory in the path locations is the path where the simulation is started. In other words, "./" points to the location where the project is called.

```
MESH
{
  length-scale              = < real |
                                METER |
                                DECIMETER |
                                CENTIMETER |
                                MILLIMETER |
                                MICROMETER |
                                NANOMETER |
                                ANGSTROM >
  time-scale                = < real |
                                SECOND |
                                MILLISECOND |
                                MICROSECOND |
                                NANOSECOND |
                                PICOSECOND |
                                FEMTOSECOND |
                                ATTOSECOND >
  mesh-lengths              = < ( real, real, real ) >
  mesh-resolution           = < ( real, real, real ) >
  mesh-center               = < ( real, real, real ) >
  total-time                = < real >
  total-distance            = < real >
  bunch-time-step           = < real >
  mesh-truncation-order     = < 1 | 2 >
  space-charge              = < true | false >
  solver                    = < NSFD | FD >
  optimize-bunch-position   = < true | false >
  initial-time-back-shift   = < real >
  lorentz-factor            = < real >
}

BUNCH
{
  bunch-initialization
  {
    type                    = < manual |
                                ellipsoid |
                                3D-crystal |
                                file >
    distribution            = < uniform | gaussian >
    file-name               = < string >
    charge                  = < real >
    number-of-particles     = < int >
    gamma                   = < real >
    beta                    = < real >
    direction               = < ( real, real, real ) >
    position                = < ( real, real, real ) >
    sigma-position          = < ( real, real, real ) >
    sigma-momentum          = < ( real, real, real ) >
    numbers                 = < ( int, int, int ) >
    lattice-constants       = < ( real, real, real ) >
    transverse-truncation   = < real >
    longitudinal-truncation = < real >
    bunching-factor         = < real between 0 and 1 >
    bunching-factor-phase   = < real >
    shot-noise              = < true | false >
  }

  bunch-sampling
  {
    sample                  = < true | false >
    directory               = < /path/to/location >
    base-name               = < string >
    rhythm                  = < real >
  }

  bunch-visualization
  {
    sample                  = < true | false >
    directory               = < /path/to/location >
    base-name               = < string >
    rhythm                  = < real >
  }

  bunch-profile
  {
    sample                  = < true | false >
    directory               = < /path/to/location >
    base-name               = < string >
    time                    = < real >
    rhythm                  = < real >
  }
}

FIELD
{
  field-initialization
  {
    type                    = < plane-wave |
```

```
                                confined-plane-wave |
                                gaussian-beam >
    position                = < ( real, real, real ) >
    direction               = < ( real, real, real ) >
    polarization            = < ( real, real, real ) >
    radius-parallel         = < real >
    radius-perpendicular    = < real >
    signal-type             = < neumann | gaussian |
                                secant-hyperbolic |
                                inverse-gaussian |
                                flat-top >
    strength-parameter      = < real >
    offset                  = < real >
    pulse-length            = < real >
    wavelength              = < real >
    rising-cycles           = < int >
    CEP                     = < real >
    sigma-inverse-gaussian  = < ( real, real ) >
  }

field-sampling
  {
    sample                  = < true | false >
    type                    = < over-line | at-point >
    field                   = < Ex | Ey | Ez |
                                Bx | By | Bz |
                                Ax | Ay | Az | F >
    directory               = < /path/to/location >
    base-name               = < string >
    rhythm                  = < real >
    position                = < ( real, real, real ) >
    line-begin              = < ( real, real, real ) >
    line-end                = < ( real, real, real ) >
    number-of-points        = < int >
  }

field-visualization
  {
    sample                  = < true | false >
    type                    = < in-plane |
                                all-domain >
    plane                   = < xy | yz | xz >
    position                = < ( real, real, real ) >
    field                   = < Ex | Ey | Ez |
                                Bx | By | Bz |
                                Ax | Ay | Az | F >
    directory               = < /path/to/location >
    base-name               = < string >
    rhythm                  = < real >
  }

field-profile
  {
    sample                  = < true | false >
    field                   = < Ex | Ey | Ez |
                                Bx | By | Bz |
                                Ax | Ay | Az | F >
    directory               = < /path/to/location >
    base-name               = < string >
    rhythm                  = < real >
    time                    = < real >
  }
}

UNDULATOR
{
  static-undulator
  {
    undulator-parameter     = < real >
    period                  = < real >
    length                  = < int >
    polarization-angle      = < real >
    offset                  = < real >
    distance-to-bunch-head  = < real >
  }

static-undulator-array
  {
    undulator-parameter     = < real >
    period                  = < real >
    length                  = < int >
    polarization-angle      = < real >
    gap                     = < real >
    number                  = < int >
    tapering-parameter      = < real >
    distance-to-bunch-head  = < real >
  }

  optical-undulator
  {
    beam-type         = < plane-wave |
                          confined-plane-wave |
                          gaussian-beam |
                          standing-plane-wave |
                          standing-confined-plane-wave |
                          standing-gaussian-beam >
    position                = < ( real, real, real ) >
    direction               = < ( real, real, real ) >
    polarization            = < ( real, real, real ) >
    radius-parallel         = < real >
    radius-perpendicular    = < real >
    signal-type             = < neumann | gaussian |
                                secant-hyperbolic |
                                inverse-gaussian |
                                flat-top >
    strength-parameter      = < real >
    offset                  = < real >
    pulse-length            = < real >
    wavelength              = < real >
    rising-cycles           = < int >
    CEP                     = < real >
    sigma-inverse-gaussian  = < ( real, real ) >
    distance-to-bunch-head  = < real >
  }
}

EXTERNAL-FIELD
{
  electromagnetic-wave
  {
    beam-type         = < plane-wave |
                          confined-plane-wave |
                          gaussian-beam |
                          standing-plane-wave |
                          standing-confined-plane-wave |
                          standing-gaussian-beam >
    position                = < ( real, real, real ) >
    direction               = < ( real, real, real ) >
    polarization            = < ( real, real, real ) >
    radius-parallel         = < real >
    radius-perpendicular    = < real >
    signal-type             = < neumann | gaussian |
                                secant-hyperbolic |
                                inverse-gaussian |
                                flat-top >
    strength-parameter      = < real >
```

```
    offset                    = < real >
    pulse-length              = < real >
    wavelength                = < real >
    rising-cycles             = < int >
    CEP                       = < real >
    sigma-inverse-gaussian    = < ( real, real ) >
  }
}

FEL-OUTPUT
{
  radiation-power
  {
    sample                    = < false | true >
    type                      = < at-point | over-line >
    directory                 = < /path/to/location >
    base-name                 = < string >
    plane-position            = < real >
    line-begin                = < real >
    line-end                  = < real >
    number-of-points          = < int >
    normalized-frequency      = < real >
    minimum-normalized-frequency    = < real >
    maximum-normalized-frequency    = < real >
    number-of-frequency-points      = < int >
  }

  power-visualization
  {
    sample                    = < false | true >
    directory                 = < /path/to/location >
    base-name                 = < string >
    plane-position            = < real >
    normalized-frequency      = < real >
    rhythm                    = < real >
  }

  bunch-profile-lab-frame
  {
    sample                    = < false | true >
    directory                 = < /path/to/location >
    base-name                 = < string >
    position                  = < real >
    rhythm                    = < real >
  }
}
```

Appendices

Appendix A

Job files

A.1 Example 1: Infrared FEL

```
MESH
{
    length-scale                    = MICROMETER
    time-scale                      = PICOSECOND
    mesh-lengths                    = ( 3200,  3200.0,    280.0)
    mesh-resolution                 = ( 50.0,    50.0,      0.1)
    mesh-center                     = ( 0.0,      0.0,      0.0)
    total-time                      = 30000
    bunch-time-step                 = 1.6
    mesh-truncation-order           = 2
    space-charge                    = false
    solver                          = NSFD
}

BUNCH
{
    bunch-initialization
    {
        type                        = ellipsoid
        distribution                = uniform
        charge                      = 1.846e8
        number-of-particles         = 131072
        gamma                       = 100.41
        direction                   = (    0.0,     0.0,       1.0)
        position                    = (    0.0,     0.0,       0.0)
        sigma-position              = (  260.0,   260.0,     50.25)
        sigma-momentum              = ( 1.0e-8,  1.0e-8, 100.41e-4)
        transverse-truncation       = 1040.0
        longitudinal-truncation     = 90.0
        bunching-factor             = 0.01
    }
}

FIELD
{
    field-sampling
    {
        sample                      = true
        type                        = at-point
        field                       = Ex
        field                       = Ey
        field                       = Ez
        directory                   = ./
        base-name                   = field-sampling/field
        rhythm                      = 3.2
        position                    = (0.0, 0.0, 110.0)
```

```
    }
}

UNDULATOR
{
    static-undulator
    {
        undulator-parameter         = 1.417
        period                      = 3.0e4
        length                      = 300
        polarization-angle          = 0.0
    }
}

FEL-OUTPUT
{
    radiation-power
    {
        sample                      = true
        type                        = at-point
        directory                   = ./
        base-name                   = power-sampling/power
        plane-position              = 110.0
        normalized-frequency        = 1.00
    }
}
```

A.2 Example 2: Seeded UV FEL

```
MESH
{
    length-scale                    = MICROMETER
    time-scale                      = PICOSECOND
    mesh-lengths                    = ( 2500.0, 2500.0, 165.0  )
    mesh-resolution                 = (   30.0,   30.0,   0.02 )
    mesh-center                     = (    0.0,    0.0,   0.0  )
    total-time                      = 50000
    bunch-time-step                 = 1.6
    mesh-truncation-order           = 2
    space-charge                    = false
    solver                          = NSFD
}

BUNCH
{
    bunch-initialization
    {
        type                        = ellipsoid
        distribution                = uniform
        charge                      = 3.4332e8
        number-of-particles         = 4194304
        gamma                       = 391.36
        direction                   = ( 0.0,  0.0,  1.0 )
        position                    = ( 0.0,  0.0,  0.0 )
        sigma-position              = ( 95.3, 95.3, 75.0)
        sigma-momentum              = ( 0.0105, 0.0105, 391.36e-4)
        transverse-truncation       = 400.0
        longitudinal-truncation     = 78.0
        bunching-factor             = 0.0
    }

    bunch-visualization
    {
        sample                      = true
        directory                   = /cluster/scratch/afallahi/
```

```
        base-name                   = bunch-visualization-seeded/bunch
        rhythm                      = 500
    }
}

FIELD
{
    field-initialization
    {
        type                        = gaussian-beam
        position                    = ( 0.0, 0.0, 700000)
        direction                   = ( 0.0, 0.0, 1.0)
        polarization                = ( 0.0, 1.0, 0.0)
        radius-parallel             = 183.74
        radius-perpendicular        = 183.74
        strength-parameter          = 9.857e-7
        signal-type                 = gaussian
        offset                      = 700000.0 #not really needed
        pulse-length                = 1.0e12
        wavelength                  = 0.265187
        CEP                         = 0.0
    }
}

UNDULATOR
{
    static-undulator
    {
        undulator-parameter         = 1.95
        period                      = 2.8e4
        length                      = 535
        polarization-angle          = 0.0
        offset                      = 0.0
    }
}

FEL-OUTPUT
{
    radiation-power
    {
        sample                      = true
        type                        = at-point
        directory                   = ./
        base-name                   = power-sampling/power
        plane-position              = 78.0
        normalized-frequency        = 1.00
    }

    bunch-profile-lab-frame
    {
        sample                      = true
        directory                   = ./
        base-name                   = bunch-profile-lab-frame/profile
        position                    =  0.0e6
        position                    =  2.0e6
        position                    =  4.0e6
        position                    =  6.0e6
        position                    =  8.0e6
        position                    = 10.0e6
        position                    = 12.0e6
    }
}
```

A.3 Example 3: Infrared FEL with Optical Undulator

```
MESH
{
    length-scale                    = MICROMETER
    time-scale                      = PICOSECOND
    mesh-lengths                    = ( 3200,  3200.0,    280.0)
    mesh-resolution                 = ( 50.0,    50.0,      0.1)
    mesh-center                     = ( 0.0,      0.0,      0.0)
    total-time                      = 30000
    bunch-time-step                 = 1.6
    mesh-truncation-order           = 2
    space-charge                    = false
    solver                          = NSFD
}

BUNCH
{
    bunch-initialization
    {
        type                        = ellipsoid
        distribution                = uniform
        charge                      = 1.846e8
        number-of-particles         = 131072
        gamma                       = 100.41
        direction                   = (    0.0,     0.0,       1.0)
        position                    = (    0.0,     0.0,       0.0)
        sigma-position              = (  260.0,   260.0,     50.25)
        sigma-momentum              = ( 1.0e-8,  1.0e-8, 100.41e-4)
        transverse-truncation       = 1040.0
        longitudinal-truncation     = 90.0
        bunching-factor             = 0.01
    }
}

UNDULATOR
{
    optical-undulator
    {
        beam-type                   = plane-wave
        position                    = ( 0.0, 0.0, 0.0 )
        direction                   = ( 0.0, 0.0,-1.0 )
        polarization                = ( 0.0, 1.0, 0.0 )
        strength-parameter          = 1.417
        signal-type                 = flat-top
        wavelength                  = 6.0e4
        pulse-length                = 18.0e6
        offset                      =  9.0e6
        CEP                         = 0.0
    }
}

FEL-OUTPUT
{
    radiation-power
    {
        sample                      = true
        type                        = at-point
        directory                   = ./
        base-name                   = power-sampling/power
        plane-position              = 110.0
        normalized-frequency        = 1.00
    }
}
```

A.4 Example 3: Inverse Compton Scattering

```
MESH
{
    length-scale                    = NANOMETER
    time-scale                      = ATTOSECOND
    mesh-lengths                    = ( 2000.0, 2000.0, 165.0 )
    mesh-resolution                 = (    5.0,    5.0,   0.05)
    mesh-center                     = (    0.0,    0.0,   0.0 )
    total-time                      = 4000000
    bunch-time-step                 = 100.0
    mesh-truncation-order           = 2
    space-charge                    = false
    solver                          = NSFD
}

BUNCH
{
    bunch-initialization
    {
        type                        = ellipsoid
        distribution                = uniform
        charge                      = 2800
        number-of-particles         = 2800
        gamma                       = 30.0
        direction                   = (  0.0,   0.0,   1.0)
        position                    = (  0.0,   0.0,   0.0)
        sigma-position              = ( 60.0,  60.0,  72.0)
        sigma-momentum              = ( 0.001,  0.001,  0.001)
        transverse-truncation       = 240.0
        longitudinal-truncation     = 77.0
        bunching-factor             = 0.0
        shot-noise                  = true
    }

    bunch-profile
    {
        sample                      = true
        directory                   = ./
        base-name                   = bunch-profile/bunch
        rhythm                      = 2000
    }
}

FIELD
{
    field-sampling
    {
        sample                      = true
        type                        = at-point
        field                       = Ex
        field                       = Ey
        field                       = Ez
        directory                   = ./
        base-name                   = field-sampling/field
        rhythm                      = 3.2
        position                    = (0.0, 0.0, 80.0)
    }
}

UNDULATOR
{
    optical-undulator
    {
        beam-type                   = plane-wave
        position                    = ( 0.0, 0.0, 0.0 )
        direction                   = ( 0.0, 0.0,-1.0 )
```

```
        polarization                = ( 0.0, 1.0, 0.0 )
        strength-parameter          = 0.5
        signal-type                 = flat-top
        wavelength                  = 1.0e3
        pulse-length                = 2.4e6
        offset                      = 1.2e6
        CEP                         = 0.0
    }
}

FEL-OUTPUT
{
    radiation-power
    {
        sample                      = true
        type                        = at-point
        directory                   = ./
        base-name                   = power-sampling/power
        plane-position              = 82
        normalized-frequency        = 1.0
        normalized-frequency        = 2.0
        normalized-frequency        = 3.0
    }
}
```

A.5 Example 4: Free-space Propagation

```
MESH
{
    length-scale                = MICROMETER
    time-scale                  = PICOSECOND
    mesh-lengths                = ( 3200,  3200.0,    500.0)
    mesh-resolution             = ( 50.0,    50.0,      0.1)
    mesh-center                 = ( 0.0,      0.0,      0.0)
    total-time                  = 30000
    bunch-time-step             = 1.6
    mesh-truncation-order       = 2
    space-charge                = true
    solver                      = NSFD
}

BUNCH
{
    bunch-initialization
    {
        type                    = ellipsoid
        distribution            = gaussian
        charge                  = 1.846e8
        number-of-particles     = 262144
        gamma                   = 100.41
        direction               = (    0.0,     0.0,       1.0)
        position                = (    0.0,     0.0,       0.0)
        sigma-position          = (  260.0,   260.0,     50.25)
        sigma-momentum          = ( 1.0e-8,  1.0e-8, 100.41e-4)
        transverse-truncation   = 1040.0
        longitudinal-truncation = 200.0
        bunching-factor         = 0.00
    }

    bunch-sampling
    {
        sample                  = true
        directory               = ./
        base-name               = bunch-sampling/bunchSC
        rhythm                  = 8
```

```
    }
}
```

A.6 Example 5: Short Pulse Hard X-ray Source

```
MESH
{
    length-scale                    = MICROMETER
    time-scale                      = PICOSECOND
    mesh-lengths                    = ( 400.0,  400.0,  1.5)
    mesh-resolution                 = ( 4.0,  4.0,  3.0e-5)
    mesh-center                     = ( 0.0,      0.0,      0.0)
    total-time                      = 300000
    bunch-time-step                 = 1.6
    mesh-truncation-order           = 2
    space-charge                    = false
    solver                          = NSFD
}

BUNCH
{
    bunch-initialization
    {
        type                        = ellipsoid
        distribution                = uniform
        charge                      = 1.25e8
        number-of-particles         = 8388608
        gamma                       = 13089
        direction                   = (    0.0,     0.0,       1.0)
        position                    = (    0.0,     0.0,       0.0)
        sigma-position              = (  30.0,   30.0,     0.4)
        sigma-momentum              = ( 0.007,  0.007, 13089e-3)
        transverse-truncation       = 180.0
        longitudinal-truncation     = 0.43
        bunching-factor             = 0.0
        shot-noise                  = true
    }
}

UNDULATOR
{
    static-undulator
    {
        undulator-parameter         = 3.5
        period                      = 3.0e4
        length                      = 2500
        polarization-angle          = 0.0
    }
}

FEL-OUTPUT
{
    radiation-power
    {
        sample                      = true
        type                        = at-point
        directory                   = ./
        base-name                   = power-sampling/power
        plane-position              = 0.45
        normalized-frequency        = 1.00
    }
}
```

Bibliography

[1] Arya Fallahi, Alireza Yahaghi, and Franz X Kärtner. Mithra 1.0: A full-wave simulation tool for free electron lasers. *Computer Physics Communications*, 228:192–208, 2018.

[2] Peter Schmüser, Martin Dohlus, and Jörg Rossbach. *Ultraviolet and soft X-ray free-electron lasers: introduction to physical principles, experimental results, technological challenges*, volume 229. Springer, 2008.

[3] Evgeny L Saldin, E Evgeny A Schneidmiller, and Mikhail V Yurkov. *The physics of free electron lasers*. Springer, 2000.

[4] Henry P Freund. *Principles of free-electron lasers*. Springer Science & Business Media, 2012.

[5] A.E. Siegman. *Lasers*. University Science Books, 1986. ISBN 9780935702118. URL `https://books.google.de/books?id=1BZVwUZLTkAC`.

[6] AS Gilmour Jr. Microwave tubes. *Dedham, MA, Artech House, 1986, 502 p.*, 1, 1986.

[7] Eberhard Jaeschke, Shaukat Khan, Jochen R Schneider, and Jerome B Hastings. *Synchrotron Light Sources and Free-Electron Lasers*. Springer, 2015.

[8] S Reiche. Genesis 1.3: a fully 3d time-dependent fel simulation code. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 429(1):243–248, 1999.

[9] Sandra G Biedron, Henry P Freund, and Stephen V Milton. 3d fel code for the simulation of a high-gain harmonic generation experiment. In *Optoelectronics' 99-Integrated Optoelectronic Devices*, pages 96–108. International Society for Optics and Photonics, 1999.

[10] Trach Minh Tran and JS Wurtele. TDA - a three-dimensional axisymmetric code for free-electron-laser (fel) simulation. *Computer Physics Communications*, 54(2):263–272, 1989.

[11] B Faatz, W Fawley, P Pierini, S Reiche, G Travish, D Whittum, and J Wurtele. Tda3d: Updates and improvements to the widely used three-dimensional free electron laser simulation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 393(1):277–279, 1997.

[12] William M Fawley. A user manual for ginger and its post-processor xplotgin. *Lawrence Berkeley National Laboratory*, 2002.

[13] L Giannessi. Overview of perseo, a system for simulating fel dynamics in mathcad. In *Proceedings of the Free-Electron Laser Conference*, 2006.

[14] A Bacci, C Maroli, V Petrillo, AR Rossi, L Serafini, and P Tomassini. Compact x-ray free-electron laser based on an optical undulator. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 587(2):388–397, 2008.

[15] Roger J Dejus, Oleg A Shevchenko, and Nikolai A Vinokurov. An integral equation based computer code for high-gain free-electron lasers. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 429(1):225–228, 1999.

[16] EL Saldin, EA Schneidmiller, and MV Yurkov. Fast: a three-dimensional time-dependent fel simulation code. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 429(1):233–237, 1999.

[17] Igor A Andriyash, Rémi Lehe, and Victor Malka. A spectral unaveraged algorithm for free electron laser simulations. *Journal of Computational Physics*, 282:397–409, 2015.

[18] LT Campbell and BWJ McNeil. Puffin: A three dimensional, unaveraged free electron laser simulation code. *Physics of Plasmas (1994-present)*, 19(9):093119, 2012.

[19] SG Biedron, YC Chae, Roger J Dejus, B Faatz, HP Freund, SV Milton, H-D Nuhn, and S Reiche. Multi-dimensional free-electron laser simulation codes: a comparison study. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 445(1):110–115, 2000.

[20] SPD Mangles, CD Murphy, Z Najmudin, AGR Thomas, JL Collier, AE Dangor, EJ Divall, PS Foster, JG Gallacher, CJ Hooker, et al. Monoenergetic beams of relativistic electrons from intense laser–plasma interactions. *Nature*, 431(7008):535–538, 2004.

[21] Jérôme Faure, Yannick Glinec, A Pukhov, S Kiselev, S Gordienko, E Lefebvre, J-P Rousseau, F Burgy, and Victor Malka. A laser–plasma accelerator producing monoenergetic electron beams. *Nature*, 431(7008):541–544, 2004.

[22] CGR Geddes, Cs Toth, J Van Tilborg, E Esarey, CB Schroeder, D Bruhwiler, C Nieter, J Cary, and WP Leemans. High-quality electron beams from a laser wakefield accelerator using plasma-channel guiding. *Nature*, 431(7008):538–541, 2004.

[23] T Tajima and JM Dawson. Laser electron accelerator. *Physical Review Letters*, 43(4):267, 1979.

[24] O Lundh, J Lim, C Rechatin, L Ammoura, A Ben-Ismail, X Davoine, Guilhem Gallot, Jean-Philippe Goddet, E Lefebvre, Victor Malka, et al. Few femtosecond, few kiloampere electron bunch produced by a laser-plasma accelerator. *Nature Physics*, 7(3):219–222, 2011.

[25] R Joel England, Robert J Noble, Karl Bane, David H Dowell, Cho-Kuen Ng, James E Spencer, Sami Tantawi, Ziran Wu, Robert L Byer, Edgar Peralta, et al. Dielectric laser accelerators. *Reviews of Modern Physics*, 86(4):1337, 2014.

[26] Emilio A Nanni, Wenqian R Huang, Kyung-Han Hong, Koustuban Ravi, Arya Fallahi, Gustavo Moriena, RJ Dwayne Miller, and Franz X Kärtner. Terahertz-driven linear electron acceleration. *Nature communications*, 6, 2015.

[27] Arya Fallahi, Moein Fakhari, Alireza Yahaghi, Miguel Arrieta, et al. Short electron bunch generation using single-cycle ultrafast electron guns. *arXiv preprint arXiv:1606.02153*, 2016.

[28] FX Kärtner, F Ahr, A-L Calendron, H Çankaya, S Carbajo, G Chang, G Cirmi, K Dörner, U Dorda, A Fallahi, et al. Axsis: Exploring the frontiers in attosecond x-ray science, imaging and spectroscopy. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 2016.

[29] Allen Taflove and Susan C Hagness. *Computational electrodynamics*. Artech house publishers, 2000.

[30] J.-L. Vay. Noninvariance of space- and time-scale ranges under a lorentz transformation and the implications for the study of relativistic interactions. *Phys. Rev. Lett.*, 98:130405, Mar 2007.

[31] P Sprangle and AT Drobot. Stimulated backscattering from relativistic unmagnetized electron beams. *Journal of Applied Physics*, 50(4):2652–2661, 1979.

[32] Peicheng Yu, Xinlu Xu, Viktor K Decyk, Weiming An, Jorge Vieira, Frank S Tsung, Ricardo A Fonseca, Wei Lu, Luis O Silva, and Warren B Mori. Modeling of laser wakefield acceleration in lorentz boosted frame using em-pic code with spectral solver. *Journal of Computational Physics*, 266:124–138, 2014.

[33] Jean-Luc Vay, Irving Haber, and Brendan B Godfrey. A domain decomposition method for pseudo-spectral electromagnetic simulations of plasmas. *Journal of Computational Physics*, 243:260–268, 2013.

[34] Jean-Luc Vay, DP Grote, RH Cohen, and Alex Friedman. Novel methods in the particle-in-cell accelerator code-framework warp. *Computational Science & Discovery*, 5(1):014019, 2012.

[35] WM Fawley and J-L Vay. Use of the lorentz-boosted frame transformation to simulate free-electron laser amplifier physics. In *AIP Conference Proceedings*, volume 1086, pages 346–350. AIP, 2009.

[36] IA Andriyash, Emmanuel d'Humières, VT Tikhonchuk, and Ph Balcou. X-ray amplification from a raman free-electron laser. *Physical review letters*, 109(24):244802, 2012.

[37] Takayuki Umeda, Yoshiharu Omura, T Tominaga, and Hiroshi Matsumoto. A new charge conservation method in electromagnetic particle-in-cell simulations. *Computer Physics Communications*, 156(1):73–85, 2003.

[38] Kurt L Shlager and John B Schneider. Comparison of the dispersion properties of several low-dispersion finite-difference time-domain algorithms. *IEEE Transactions on Antennas and Propagation*, 51(3):642–653, 2003.

[39] Bezalel Finkelstein and Raphael Kastner. Finite difference time domain dispersion reduction schemes. *Journal of Computational Physics*, 221(1):422–438, 2007.

[40] Jay P Boris. Acceleration calculation from a scalar potential. Technical report, Princeton Univ., NJ Plasma Physics Lab., 1970.

[41] JP Boris. Relativistic plasma simulation-optimization of a hybrid code. In *Proc. Fourth Conf. Num. Sim. Plasmas, Naval Res. Lab, Wash. DC*, pages 3–67, 1970.

[42] D Sagan, JA Crittenden, D Rubin, and E Forest. A magnetic field model for wigglers and undulators. In *Particle Accelerator Conference, 2003. PAC 2003. Proceedings of the*, volume 2, pages 1023–1025. IEEE, 2003.

[43] C Pellegrini, A Marinelli, and S Reiche. The physics of x-ray free-electron lasers. *Reviews of Modern Physics*, 88(1):015006, 2016.

[44] Sven Reiche. Numerical studies for a single pass high gain free-electron laser. Technical report, DESY, 2000.

[45] C. Penman and B.W.J. McNeil. Simulation of input electron noise in the free-electron laser. *Optics Communications*, 90(1):82 – 84, 1992. ISSN 0030-4018.

[46] C Penman and BWJ McNeil. Simulation of input electron noise in the free-electron laser. *Optics communications*, 90(1-3): 82–84, 1992.

[47] C Maroli and V Petrillo. Effects of the low-frequency backward wave in high-gain free-electron lasers. *Optics communications*, 183(1):139–147, 2000.

[48] Trach-Minh Tran and JS Wurtele. Review of free-electron-laser (fel) simulation techniques. Technical report, Ecole Polytechnique Federale, Lausanne (Switzerland), 1990.

[49] R Bonifacio, N Piovella, GRM Robb, and Angelo Schiavi. Quantum regime of free electron lasers starting from noise. *Physical Review Special Topics-Accelerators and Beams*, 9(9):090701, 2006.

[50] R Bonifacio, N Piovella, and GRM Robb. Quantum theory of sase fel. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 543(2):645–652, 2005.

[51] Eric Esarey, Sally K Ride, and Phillip Sprangle. Nonlinear thomson scattering of intense laser pulses from beams and plasmas. *Physical Review E*, 48(4):3003, 1993.

[52] Klaus Flöttmann et al. Astra: A space charge tracking algorithm. *Manual, Version*, 3:2014, 2011.