SIMLA: Simulating laser particle interactions

Manual for version 1.0

C. N. Harvey and D. G. Green

2014

Copyright © 2014 by Christopher Harvey and Dermot Green.
All rights reserved.

Contents

Chapter 1

Introduction

SIMLA is a software package for calculating the trajectories and emission spectra of charged particles in arbitrary electromagnetic background fields, including laser fields and ultraintense[1] laser fields.

The SIMLA program can operate in both classical or quantum-electrodynamics (QED) modes. Importantly, the program takes account of the effects of emission of radiation from the charge as it is accelerated in the fields. These effects, which include modification of the particle trajectory and reduction in the particle energy, can be significant in strong fields in which the electron experiences large acceleration. This so called '*radiation reaction*' is included in the classical mode of operation via the Landau-Lifshitz equation, or alternatively in the QED mode, via the simulation of stochastic photon emission events determined by strong-field quantum-electrodynamics amplitudes and implemented using Monte-Carlo type routines. Multiple laser fields can be included in the simulation and the propagation direction, beam shape (plane wave, focussed paraxial, constant crossed, or constant magnetic), and time envelope of each can be independently specified. It is possible to create movies showing the evolution of the fields.

SIMLA is *not* a particle-in-cell program. It is written primarily for the simulation of high-energy collisional experiments where collective effects can be neglected. An advantage of this is that it can run on an ordinary computer, rather than requiring large scale supercomputing resources. Moreover, an important feature of SIMLA is its adaptive grid, which further significantly increases

[1] For example, lasers with focussed intensities of $\sim 10^{18}$–10^{24} Wcm^{-2}, which will be found at upcoming large-scale international facilities, such as the 'ELI', the European Light Infrastructure. It is usual to characterise the field experienced by a particle via the *dimensionless laser amplitude*, defined in the lab frame as $a_0 \equiv qE_{\rm rms}/\omega mc$, where $E_{\rm rms}$ is the rms electric field strength, ω is the laser frequency, and q and m are the charge and rest mass of the particle respectively. By 'intense', or 'strong' fields, we typically mean those with $a_0 \sim$1–10^2 (for an electron or positron). Note that for a Lorentz- and gauge-invariant definition of the amplitude is given in Eqn. (14) of [1].

efficiency: a particle track through a realistic field can be calculated on an ordinary desktop computer in just a few seconds. Additionally, it allows for precision calculations of particle trajectories and emission spectra, enabling comparisons to be made with analytical calculations. Thus it is hoped it will facilitate work on the improvement of existing numerical techniques.

This document outlines the background physics and basic functionality of the SIMLA program. The main program is written in Fortran 90, with a separate suite of tools written in MATLAB for post-processing of the output data. Instructions on how to download, compile and run the program are given, with several tutorial style examples detailed.

1.0.1 Authorship and copyright

The driving program (Fortran 90) was initially written by Christopher Harvey, and was developed by Harvey and by Dermot Green, who wrote the QED modules (Fortran 90). The authors can be contacted via email at:
cnharvey@physics.org
dermot.green@balliol.oxon.org.

All publications resulting from the use of this program should include a reference to Green and Harvey, *Comp. Phys. Commun.* XX XXX (20XX).

Chapter 2

Overview

2.1 Installation

The source files can be downloaded from `http://cnharvey.github.io/SIMLA/`. In order to compile and run the files you must have a fortran compiler installed on your computer. Consult your compiler's manual for further information. Once extracted the following files should be present:

makefile the makefile for the program, which can be used on a Unix (or Mac) system to compile the software with either GNU gfortran or ifortran.

simla.f90 the source code of the main program.

simla_subroutines.f90 the source code of the various subroutines (except for those involved in the calculation of the QED processes).

simla_qedroutines.f90 the source code of the subroutines used to calculate the QED processes.

simlaplot.m a MATLAB file used for reading in (and plotting) *all* of the particle trajectories created during a run.

simlasingle.m a MATLAB file used for reading in (and plotting) a single, specified, particle trajectory created during a run.

photon.m a MATLAB file used for reading in (and plotting) all the emitted photon data if the program has been run in QED mode.

simlafields.m a MATLAB file used for reading in (and plotting) the field intensity data if outputted by the program.

simlafields_addtraj.m a MATLAB file used for reading in and plotting a specific particle trajectory on top of the field intensity plot created by *simlafields.m*.

spectrum.m a MATLAB file used to calculate the classical emission spectrum of a particle using the trajectory data read in by *simlaplot.m* or *simla_single.m*.

input_setup.txt an example file showing the structure of the input file specifying the setup of the simulation.

input_fields.txt an example file showing the structure of the input file specifying the setup of the background fields.

particle_input.csv an example file showing the structure of the input file specifying properties of the particles to be used in the run.

Once these files have been moved to an appropriate folder the program may be compiled on a Unix or Mac as follows

1. Open a terminal window
2. Navigate to the directory where the files are stored, e.g. `cd ~/home/simla`
3. Remove any previous executables by typing `make clean`
4. Compile the code by typing `make simla`
5. The program can then be executed by typing `./simla`

(Note that in step 5 typing '`./simla`' is all that is needed: the program reads the named input files automatically, i.e., there is no need to feed the input files to the executable in this line.) It is not so straightforward to use the makefile if using a Windows machine. In this case, the program can be compiled and linked manually by following the instructions given for the specific compiler. For example, in the case of gfortran, the user should replace steps 3–5 with

3. Compile all the files (except the main code) by typing
   `gfortran -c simla_subroutines.f90 simla_qedroutines.f90`

4. Compile and link the resulting object files by typing `gfortran simla.f90 simla_subroutines.o simla_qedroutines.o -o simla.exe`

5. The program can then be executed by typing `simla.exe`

For further information consult your compiler manual.

Note that, except in the case of calculating the classical emission spectrum, the MATLAB files are not essential for running the program, but have been provided for the users convenience. If the user wishes, the output data can, of course, be read using the software of their choice.

2.2 The Basics

2.2.1 Specifying the Simulation Parameters

The various parameters determining the setup of a simulation should be specified by the user via the set of three input files: *input_setup.txt*, which governs the main running of the program; *input_fields.txt*, which defines the background fields, and; *particle_input.csv*, which contains information about what particles are in the simulation and which equations govern their motion. Detailed descriptions of the contents of these input files, and how one can use them to set up a calculation, are discussed in Sec. 3 and Sec. 4.

2.2.2 Essential Information about the SIMLA program

The code reads through the *particle_input.csv* file conducting a simulation of the particle passing through the background fields for each particle in turn. Each simulation will run while the particle is in a four-dimensional region of spacetime called the *simulation box*. The size (and duration) of this box is specified by the user via the input files, as will be discussed later. Once the particle leaves this box, the code will write the final data about the particle to the file *finaldata.dat* and then terminate the run, moving to the next particle in the list. As well as a *simulation box* there is also a *write box*. The write box should be a subspace of the simulation box (although this isn't strictly necessary). While the particle is in the write box, the various data, e.g. trajectory, momentum, etc., will be written to the output file *trajectoryXXXXX.dat*, where *XXXXX* is the run/particle number. Unlike in the case of the simulation box, for which when the particle leaves the simulation will terminate, it is possible for the particle to enter and leave the write box multiple times during the simulation: the particle data is, however, only be written to file during those times when the particle is inside the write box. An illustration of the simulation and write boxes is shown in Figure 2.1.

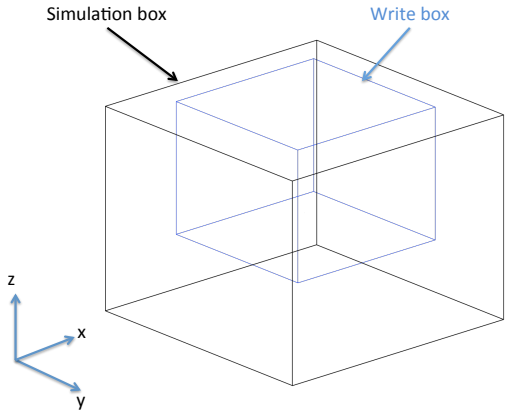

Figure 2.1: The simulation box and the write box.

The user may at first wonder why there is a need for two separate boxes. There are several reasons for this. Firstly, the user may only be interested in seeing the trajectory in the most intense region of a focussed pulse, but may additionally wish to know the energy and direction of the particle at the far-field limit. In that case the write box would be sized to fit around the pulse focus, while the simulation box would be much larger so that the final data would be recorded when the particle is a long way from the field. Secondly, in the case of long, strongly focussed fields, the particle may have to travel through a long region of weak 'pre-pulse' field before it reaches the vicinity of the focus. During the journey through this weak-field region the particle trajectory can be slowly altered and so it is necessary to include it in the simulation. However, the user may not wish for all of the data during this transit to be written to file.

One important feature of the program is that it uses an adjustable time step. This is one reason why, in cases where collective effects can be neglected, the program is superior to particle-in-cell (PIC) programs, where such a feature is not possible. The user specifies in the input files the maximum permissible error in the norm of the spatial coordinates that can be introduced at each time step. The program then propagates the particle through the simulation, making continuous adjustments to the time step such that the error condition is maintained. Thus the grid points are not uniform: there will be very few time steps in regions where the background field is weak, while there will be many in more intense/complex regions of the field. We have found that the implementation of the adjustable grid typically decreases the computation time by around two orders of magnitude compared to a fixed size grid.

Rather than outlining the format of the input files here, we will demonstrate their construction via a series of guided examples in the next chapter.

2.3 The Physics

2.3.1 Classical Motion

In the classical case where we neglect radiation damping effects, the motion of a charged particle in an electromagnetic background field is described by the Lorentz-force equation,

$$\frac{d\mathbf{p}}{dt} = q[\mathbf{E} + \mathbf{v} \times \mathbf{B}], \tag{2.1}$$

where $\mathbf{p} = \gamma m \mathbf{v}$ is the relativistic momentum expressed in terms of the γ-factor, velocity $\mathbf{v}$, charge of the particle q and the particle mass m. The electric and magnetic field components $\mathbf{E}$ and $\mathbf{B}$ can be arbitrary functions of space and time.

2.3.2 Radiation Reaction

Both classical and quantum electrodynamics dictate that a particle that is accelerated by a background field will radiate. If the acceleration is strong, this radiation can modify the dynamics, e.g., significantly reducing the energy and momentum of the particle. There will be times when the user will wish to include the effect of this *radiation reaction* on the particle motion, especially in the case of high-intensity fields (as quantified below).

In the classical case, the spectrum of radiation is continuous (the quantum case is discussed below), and the effect of this radiation reaction on the particle dynamics can be included by means of a correctional term to the Lorentz-force equation (Eq. (1)). Determining the 'exact' form of this correction is, however, non-trivial. It has been the subject of study for over 100 years, and remains one of the most fundamental problems in electrodynamics. The most common starting point to describe the radiation reaction is the Lorentz-Abraham-Dirac equation, which is obtained by solving the coupled Lorentz and Maxwell's equations for the particle and the fields [2–4]. However, this equation is notorious due to defects such as pre-acceleration and (unphysical) runaway solutions. In the SIMLA program we adopt one of the most common ways of avoiding these problems, namely the perturbative approximation of Landau and Lifshitz [5]. In this approach the equation of motion becomes

$$\frac{d\mathbf{p}}{dt} = \mathbf{f}_{\mathrm{L}} + \mathbf{f}_{\mathrm{R}}, \tag{2.2}$$

where $\mathbf{f}_{\mathrm{L}}$ is the Lorentz force (2.1) and the radiative correction term

$$\begin{aligned}\mathbf{f}_{\mathrm{R}} &= -\left(\frac{2}{3}\frac{q^3}{4\pi m}\right)\gamma\left[\left(\frac{\partial}{\partial t}+\mathbf{v}\cdot\nabla\right)\mathbf{E}+\mathbf{v}\times\left(\frac{\partial}{\partial t}+\mathbf{v}\cdot\nabla\right)\mathbf{B}\right] \\ &\quad+\left(\frac{2}{3}\frac{q^4}{4\pi m^2}\right)[(\mathbf{E}+\mathbf{v}\times\mathbf{B})\times\mathbf{B}+(\mathbf{v}\cdot\mathbf{E})\mathbf{E}] \\ &\quad-\left(\frac{2}{3}\frac{q^4}{4\pi m^2}\right)\gamma^2[(\mathbf{E}+\mathbf{v}\times\mathbf{B})^2-(\mathbf{v}\cdot\mathbf{E})^2]\mathbf{v}. \qquad (2.3)\end{aligned}$$

Equation (2.3) is valid when the radiative-reaction force is much less than the Lorentz force in the instantaneous rest frame of the particle. We note that there are numerous alternative equations in the literature (see, for example, [6, 7]) and it is still an open problem as to which is the correct formulation. However, the Landau-Lifshitz equation has, along with some others, recently been shown to be consistent with quantum electrodynamics to the order of the fine structure constant α [8, 9].

It should be noted that the first term (derivative term) of eq. (2.3) is significantly smaller than the remaining terms, owing to the fact that it only linear in the field strength whereas the other terms are quadratic. It is found that in almost all cases the contribution from this term is negligible and so, to increase computational speed, it is not evaluated by the program.

We end by stressing that the modular nature of the code allows a user with some programming knowledge to quickly and easily write an alternative particle pusher module if implementation of a different equation of motion is desired.

2.3.3 Quantum electrodynamics and Photon Emission

In addition to classical calculations, the program can calculate dynamics of electrons in fields via simulation of photon emission events determined by strong-field quantum-electrodynamics amplitudes and implemented using Monte-Carlo type routines.

To understand the strong-field QED we begin by introducing the dimensionless and invariant 'quantum efficiency' parameter for the particle

$$\chi_e \equiv \sqrt{(F^{\mu}{}_{\nu}p^{\nu})^2}/m^2 \sim \gamma E/E_{\mathrm{cr}} \qquad (2.4)$$

where $E_{\mathrm{cr}} = 1.3\times10^{16}\,\mathrm{Vcm}^{-1}$ is the QED 'critical' field ('Sauter-Schwinger' field) [10–12]. This can be interpreted as the work done on the particle by the laser field over the distance of a Compton wavelength. It is thus a measure of the importance of quantum effects for a given set of parameters. In the limit $a_0 \gg 1$ the size of the radiation formation region is of the order $\lambda/a_0 \ll \lambda$, where

$\lambda = 2\pi/\omega$ is the laser wavelength [13]. Thus the laser varies on a scale much larger than the formation region and so can be approximated as locally constant and crossed, allowing us to determine the probability of photon emission using the differential rate [13]

$$d\Gamma = \frac{\alpha m}{\sqrt{3}\pi\gamma\chi_e}\left[\left(1-\eta+\frac{1}{1-\eta}\right)K_{2/3}(\tilde{\chi}) - \int_{\tilde{\chi}}^{\infty} dx K_{1/3}(x)\right] d\chi_\gamma, \quad (2.5)$$

where K_ν is the modified Bessel function of order ν, $\eta \equiv \chi_\gamma/\chi_e$, the parameter $\tilde{\chi} \equiv 2\eta/\left[3\chi_e\left(1-\eta\right)\right]$, and we have introduced the analogous invariant parameter $\chi_\gamma \equiv \sqrt{(F^\mu{}_\nu \kappa^\nu)^2}/m^2$ for the emitted photon with momentum κ^ν.

Then the QED routines operate in the following manner. The program implements its classical particle pusher to propagate the particle through the field via the Lorentz equation. After every time step a uniform random number $r \in [0,1]$ is generated, and emission deemed to occur if the condition $r \leq \Gamma dt$ is satisfied, under the requirement $\Gamma dt \ll 1$. Note that during the simulation $d\Gamma$ (and thus Γ) is a time-dependent quantity owing to the effect of the temporally varying laser pulse and electron motion. Given an emission event, the photon χ_γ is determined as the root of the sampling equation $\zeta = \Gamma(t)^{-1}\int_0^{\chi_\gamma} d\Gamma(t)$, where ζ is a uniform random number $\zeta \in [0,1]$ (In actual fact, the integral is performed from a lower limit $\varepsilon \sim 10^{-5}$, rather than zero: the emission of soft photons of energy below this cut off does not appreciably affect the particle motion (see e.g., Ref. [14]).) Next, the program determines the photon momentum from χ_γ assuming that the emission is in the direction of motion of the particle. This is valid for $\gamma \gg 1$, since in reality the emissions will be in a cone of width γ^{-1} [15, 16]. Finally, the particle momentum is updated and the simulation continues by propagating the particle via the Lorentz equation to the next time step. A schematic of the subroutine `qedsuroutine` that controls the emission process and that is called at each time step by the main driving program is shown in Fig. 2.2.

This method of implementing the emission process via statistical routines is similar to those used in a number of recently developed particle-in-cell PIC programs for the modeling of QED cascades, see e.g., [17–19]. The validity of using such a method has recently been tested in [20] where it was found that it accurately reproduces the correct photon spectra in all but a few special cases.

Note that when $\chi_e \gtrsim 1$ quantum effects dominate and pair production can occur. Being a single particle program, the current version of SIMLA is not designed to work in this parameter regime and will abort if this regime is entered.

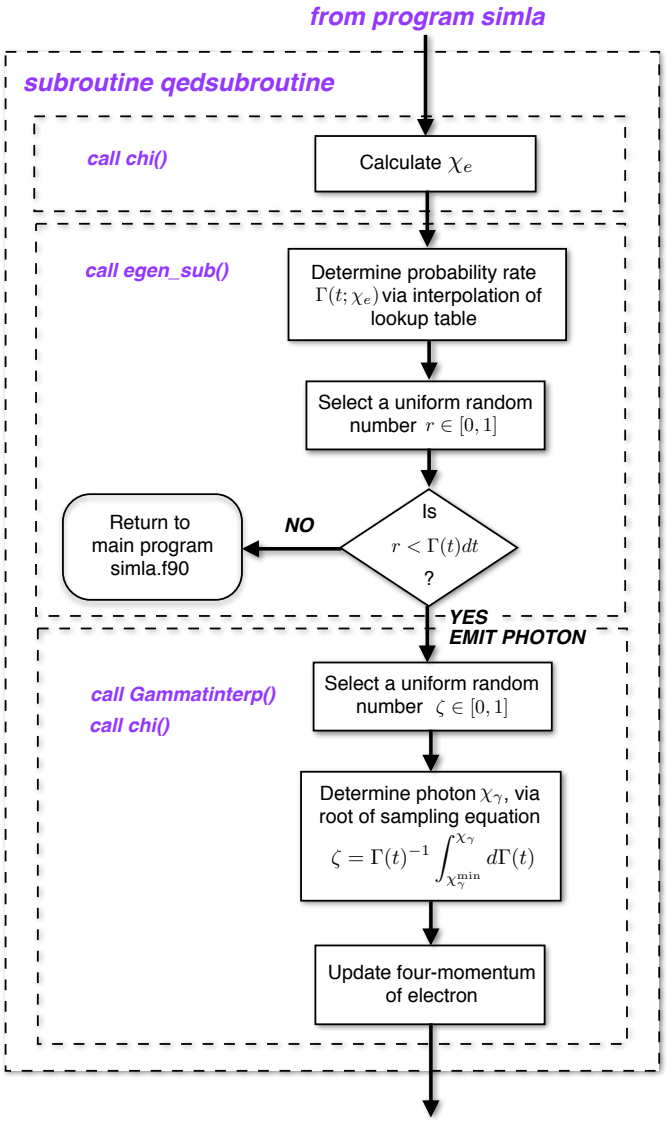

Figure 2.2: Schematic of QED subroutine. In the QED mode of operation particles are propagated between successive steps according to the Lorentz-force equation. At the end of the time step, the subroutine `qedsubroutine` is called to determine whether or not photon emission should occur during that time step, (and if so, the χ_γ and four-momentum of the photon) via a Monte-Carlo sampling of the strong-field QED amplitude $\Gamma(t;\chi_e) = \int_0^{\chi_e}(d\Gamma/d\chi_\gamma)d\chi_\gamma$, where $d\Gamma$ is given in Eq. (2.5). If a photon emission event is deemed to occur, the electron four-momentum is updated to take account of the recoil before propagating via the Lorentz-force propagation over the next time step, where the process continues. Note that rather than calculate the integral at every time step, on initialisation the main `PROGRAM SIMLA` calculates a table of the probability rate for photon emission for electron invariant parameters in the range $\chi_e \in [\chi_e^{\min}, \chi_e^{\max}]$, and during execution this table is interpolated to obtain Γ for the specific $\chi_e(t)$ of the electron. The parameters $\chi_e^{\min}$ and $\chi_e^{\max}$ are specified in the `PROGRAM SIMLA` (default values are $\chi_e^{\min} = 5\times10^{-4}$ and $\chi_e^{\max} = 1$). If, at a given time step, $\chi_e < \chi_e^{\min}$ then we assume no photon emission at that time step. If $\chi_e^{\max} > 1$ the code will abort with an error message. Note also that the condition $\Gamma dt \ll 1$ must be satisfied, and again the code will abort if this condition is not met. Full details are given in the text.

2.4 The Background Fields

2.4.1 Constant Fields

The program allows for constant-crossed-field backgrounds. Their default orientation is with the electric field $\mathbf{E}$ in x and magnetic field $\mathbf{B}$ in y, as follows:

$$\begin{aligned} E_x &= g(t-z) \times \texttt{field\_strength}, && (2.6)\\ E_y &= 0, && (2.7)\\ E_z &= 0, && (2.8)\\ B_x &= 0, && (2.9)\\ B_y &= g(t-z) \times \texttt{field\_strength}, && (2.10)\\ B_z &= 0, && (2.11) \end{aligned}$$

where $g(t-z)$ is the time profile function.

2.4.2 Plane Waves

In the case of linearly polarised plane waves the default orientation is for them to propagate in z and to be polarised in x

$$\begin{aligned} E_x &= g(t-z)E_0 \sin\left[\omega\left(t-z/c\right)\right], && (2.12)\\ E_y &= 0, && (2.13)\\ E_z &= 0, && (2.14)\\ B_x &= 0, && (2.15)\\ B_y &= g(t-z)E_0 \sin\left[\omega\left(t-z/c\right)\right], && (2.16)\\ B_z &= 0. && (2.17) \end{aligned}$$

The case of circular polarisation is

$$\begin{aligned} E_x &= g(t-z)\frac{E_0}{\sqrt{2}} \sin\left[\omega\left(t-z/c\right)\right], && (2.18)\\ E_y &= g(t-z)\frac{E_0}{\sqrt{2}} \cos\left[\omega\left(t-z/c\right)\right], && (2.19)\\ E_z &= 0, && (2.20)\\ B_x &= -g(t-z)\frac{E_0}{\sqrt{2}} \cos\left[\omega\left(t-z/c\right)\right], && (2.21)\\ B_y &= g(t-z)\frac{E_0}{\sqrt{2}} \sin\left[\omega\left(t-z/c\right)\right], && (2.22)\\ B_z &= 0. && (2.23) \end{aligned}$$

2.4.3 Paraxial Beams

Another option is to have a paraxial Gaussian beam field, which more accurately describes a pulsed laser beam. We begin by outlining the derivation of the fields so that the user can better understand the approximations and assumptions involved. Proceeding in the same way as McDonald [21] we begin by adopting the Lorentz gauge

$$\frac{\partial \phi}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{A} = 0, \tag{2.24}$$

and dictate that any vector potential describing our laser field must satisfy the vacuum wave equation

$$\nabla^2 \boldsymbol{A} = \frac{\partial^2 \boldsymbol{A}}{\partial t^2}. \tag{2.25}$$

We take the laser to propagate in the $+z$ direction and assume a generic potential, linearly polarised in x

$$\boldsymbol{A} = \hat{\boldsymbol{x}} A_0 g(\eta) \psi(x, y, z) e^{-ikz}, \tag{2.26}$$

where A_0 is the wave amplitude, $\eta = \omega t - kz$, and g is a generic pulse shape function. Inserting (2.26) into (2.25) gives us

$$\nabla^2 \psi - 2ik \frac{\partial \psi}{\partial z} \left(1 - i \frac{g'}{g} \right) = 0, \tag{2.27}$$

where $g' = dg/d\eta$. In general it is hard to satisfy (2.27) since ψ is a function of (x, y, z) and g is a function of the phase η. To proceed we begin by rescaling our coordinates

$$\xi \equiv \frac{x}{w_0}, \quad \nu \equiv \frac{y}{w_0}, \quad \zeta \equiv \frac{z}{z_r}, \tag{2.28}$$

making them dimensionless. Here w_0 is the beam waist diameter and $z_r = k w_0^2 / 2$ is the Rayleigh length. Following [21] we specify that the pulse shape function satisfies

$$g' \ll g. \tag{2.29}$$

Equation (2.27) can then be approximated by

$$\nabla_\perp^2 \psi - 4i \frac{\partial \psi}{\partial \zeta} + \theta_0^2 \frac{\partial^2 \psi}{\partial \zeta^2} = 0, \tag{2.30}$$

where

$$\nabla_\perp^2 = \frac{\partial^2}{\partial \xi^2} + \frac{\partial^2}{\partial \nu^2}, \quad \psi = \psi(\xi, \nu, \zeta), \tag{2.31}$$

and we have introduced the aspect ratio $\theta_0 = w_0/z_r$ which, when small, closely approximates the beam diffraction angle. Since θ_0 is typically small, we can expand ψ in the series

$$\psi = \psi_0 + \theta_0^2 \psi_2 + \theta_0^4 \psi_4 + \dots. \tag{2.32}$$

Equating coefficients of θ_0 we have, from (2.30),

$$\nabla_\perp^2 \psi_0 - 4i \frac{\partial \psi_0}{\partial \zeta} = 0, \tag{2.33}$$

$$\nabla_\perp^2 \psi_2 - 4i \frac{\partial \psi_2}{\partial \zeta} + \frac{\partial^2 \psi_0}{\partial \zeta^2} = 0, \tag{2.34}$$

$$\nabla_\perp^2 \psi_4 - 4i \frac{\partial \psi_4}{\partial \zeta} + \frac{\partial^2 \psi_2}{\partial \zeta^2} = 0, \tag{2.35}$$

$$\text{etc.}$$

Equation (2.33) is the paraxial wave equation from traditional Gaussian beam theory. Its solution is the well known first order Gaussian beam solution

$$\psi_0 = f e^{-f\rho^2}, \tag{2.36}$$

where

$$f = \frac{1}{\sqrt{1+\zeta^2}} e^{i \arctan \zeta}, \quad \rho^2 = \xi^2 + \nu^2. \tag{2.37}$$

The solution to (2.34) was originally found by Davis [22]

$$\psi_2 = \left(\frac{f}{2} + \frac{f^3 \rho^4}{4} \right) \psi_0, \tag{2.38}$$

and Barton and Alexander [23] proceeded to find the solution to (2.35)

$$\psi_4 = \frac{1}{32} (12 f^2 - 6 f^4 \rho^4 - 4 f^5 \rho^6 + f^6 \rho^8) \psi_0. \tag{2.39}$$

Analogously to the case of the vector potential (2.26), we assume that the scalar potential can be written in the form

$$\phi(t, x, y, z) = g(\eta) \Phi(x, y, z) e^{i\eta}. \tag{2.40}$$

Then the Lorentz gauge condition (2.24) gives us

$$\frac{\partial\phi}{\partial t} = i\omega\phi\left(1 - i\frac{g'}{g}\right) \approx i\omega\phi, \tag{2.41}$$

which means that

$$\phi = \frac{i}{k}\boldsymbol{\nabla}\cdot\boldsymbol{A}. \tag{2.42}$$

Thus our electric and magnetic field components may be found from (2.26) via

$$\boldsymbol{E} = -ik\boldsymbol{A} - \frac{i}{k}\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{A}), \tag{2.43}$$

$$\boldsymbol{B} = \boldsymbol{\nabla}\times\boldsymbol{A}, \tag{2.44}$$

(for details of the calculation see [23, 24]). Taking the real part of the resulting expressions gives us (to fifth order in θ_0)

$$\begin{aligned} E_x &= P\Big(S_0 + \frac{\theta_0^2}{4}\left[4\xi^2 S_2 - \rho^4 S_3\right] + \frac{\theta_0^4}{32}\left[4S_2 - 8\rho^2 S_3\right. \\ &\quad \left. - 2\rho^2(\rho^2 - 16\xi^2)S_4 - 4\rho^4(\rho^2 + 2\xi^2)S_5 + \rho^8 S_6\right]\Big), \end{aligned} \tag{2.45}$$

$$E_y = P\xi\nu\Big(\theta_0^2 S_2 + \frac{\theta_0^4}{4}\left[4\rho^2 S_4 - \rho^4 S_5\right]\Big), \tag{2.46}$$

$$\begin{aligned} E_z &= P\xi\Big(\theta_0 C_1 + \frac{\theta_0^3}{4}\left[-2C_2 + 4\rho^2 C_3 - \rho^4 C_4\right] \\ &\quad + \frac{\theta_0^5}{32}\left[-12C_3 - 12\rho^2 C_4 + 34\rho^4 C_5 - 12\rho^6 C_6 + \rho^8 C_7\right]\Big), \end{aligned} \tag{2.47}$$

$$B_x = 0, \tag{2.48}$$

$$\begin{aligned} B_y &= P\Big(S_0 + \frac{\theta_0^2}{4}\left[2\rho^2 S_2 - \rho^4 S_3\right] \\ &\quad + \frac{\theta_0^4}{32}\left[-4S_2 + 8\rho^2 S_3 + 10\rho^4 S_4 - 8\rho^6 S_5 + \rho^8 S_6\right]\Big), \end{aligned} \tag{2.49}$$

$$\begin{aligned} B_z &= P\nu\Big(\theta_0 C_1 + \frac{\theta_0^3}{4}\left[2C_2 + 2\rho^2 C_3 - \rho^4 C_4\right] \\ &\quad + \frac{\theta_0^5}{32}\left[12C_3 + 12\rho^2 C_4 + 6\rho^4 C_5 - 8\rho^6 C_6 + \rho^8 C_7\right]\Big), \end{aligned} \tag{2.50}$$

where the prefactor is given by

$$P = A_0\frac{w_0}{w}g(\eta)\exp\big(-\frac{r^2}{w^2}\big), \quad r^2 = x^2 + y^2. \tag{2.51}$$

Here $w = w(z)$ is a measure of the beam diameter according to

$$w(z) = w_0\sqrt{1 + \left(\frac{z}{z_r}\right)^2}, \tag{2.52}$$

and the functions S_j and C_j are defined

$$S_j = \left(\frac{w_0}{w}\right)^j \sin\Theta, \tag{2.53}$$

$$C_j = \left(\frac{w_0}{w}\right)^j \cos\Theta, \tag{2.54}$$

where

$$\Theta = \eta - \frac{kr^2}{2H} + (j+1)\arctan\zeta, \tag{2.55}$$

where $H = z + z_r^2/z$ is the radius of curvature of the field.

The electric and magnetic field components (2.45-2.50) describe the laser to fifth order in θ_0. For an optical laser of wavelength $\lambda = 1\,\mu$m focussed to a waist size $w_0 = 5\,\mu$m the expansion parameter is

$$\theta_0 = \frac{\lambda}{\pi w_0} = \frac{1}{5\pi} \approx 0.064. \tag{2.56}$$

It has been shown in [24] that, for an optical laser with waist $w_0 = 8\mu m$, the (non radiating) electron dynamics are not significantly altered when modelling the field to terms up to fifth order in θ_0 as compared to only including terms up to third order.

The SIMLA program offers the choice of two paraxial beams. Selecting `parax1` will implement equations (2.45–2.50) to first order in θ_0. For more tightly focussed fields the command `parax5` will implement the expressions to fifth order.

Chapter 3

Quick-start Examples

3.1 Example 1: Classical electron in a plane-wave field

Simulate an electron colliding with a plane wave. Plot the trajectory. Calculate the classical emission spectrum.

As a first example let us consider the case of an electron colliding head on with a pulsed plane-wave laser field. We will consider a low-intensity field (such that we can neglect radiation reaction) and thus consider the particle motion as determined by the Lorentz-force equation. We will calculate the trajectory of the electron as well as its classical radiation emission spectrum. An illustration of the setup is shown in Figure 3.1.

We begin by setting up the three input files. First, we define the background field in the file *input_fields.txt*. Since there is only one laser we set `no_fields=1`. We then define the field to be a linearly polarised plane wave, `field1=linpw`, with a gaussian time envelope, `profile1=gauss`. The FDHM time duration will

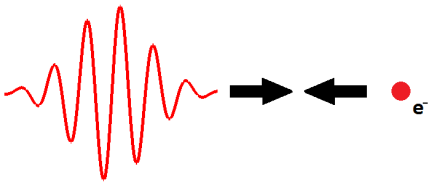

Figure 3.1: A head-on collision between a laser pulse and an electron.

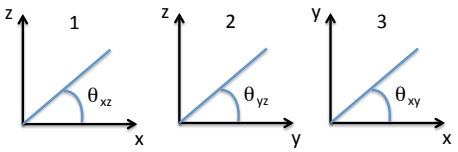

Figure 3.2: Diagram showing definitions of the field rotation angles and the order in which they are applied.

input_fields.txt

```
no_fields=1
! none, crossed,linpw,circpw,parax1,parax5,constB
field1=linpw

! inf,step,stepsin2,sech,gauss,gauss4,gauss8,gauss12
profile1=gauss

! NB no more than 4 digits before decimal
anglexz1=0
angleyz1=0
anglexy1=0

lambda1=0.8e-6

waist1=5e-6

a0_1=1

duration1=30e-15
```

Figure 3.3: The file *input_fields.txt* for Example 1.

be 30 fs, so we define `duration1=30e-15`.

The default direction for the laser field models is for them to propagate in the positive z-direction. In the case of linear polarisation, the default polarisation is in the positive x-direction. If different directions are required, it is possible to rotate the fields around any of the three coordinates axes by specifying the angles `anglexz`, `angleyz`, `anglexy`, as defined in Fig. 3.2. Since 3D rotations do not commute, one should observe carefully the order of the rotations shown in Fig. 3.2. Particular care must be taken to ensure that the polarisation direction and handedness of the beam are pointing in the right directions. This is one of the reasons why the program implements three angular rotations for the fields, even though it is possible to describe 3D rotations with just two polar angles.

In our case we simply want the beam to point along the z-axis (and to be polarised in x) so we set `anglexz1=0`, `angleyz1=0`, `anglexy1=0`. Finally, we set the wavelength to be 0.8 μm, by specifying `lambda1=0.8e-6`, and the dimensionless intensity equal to unity, `a0_1=1`. The *input_fields.txt* file should then look similar to that in Figure 3.3. Note how the file contains lines of comment (beginning with an "!") which are not read by the program. Additionally, it does not matter that additional parameters are defined which are not needed in this simulation. These will simply be ignored. Finally, note that it is essential that there are no spaces to the left or right of the "=" signs.

Next we will set up the input file describing the electron: *particle_input.csv*.

Position	Quantity	Comments
1	particle label	for reference only
2	species	e, p, H+
3	θ_0	$\theta_0 = \arctan(y_0/x_0)$
4	ϕ_0	$\phi_0 = \arccos(z_0/dist_0)$
5	$dist_0$	initial radial distance from (x_0, y_0, z_0)
6	x_0	x-coordinate particle is initially aiming for
7	y_0	y-coordinate particle is initially aiming for
8	z_0	z-coordinate particle is initially aiming for
9	γ_0	initial γ-factor
10	equation of motion	lf, ll, qed
11	output flag	t, x, ct

Table 3.1: The required variables (and their order) for each particle specified in the file *particle_input.csv*.

This file behaves slightly differently to the other two; it has specific formatting requirements which are more rigorously enforced than for *input_setup.txt* and *input_fields.txt*. If the program finds deviations from the prescribed format it will terminate. The first line of the file is just for comments and is ignored by the program. The second line should read either `repeatfirstline=off` or `repeatfirstline=on`. In the former case (normal use) the program will execute simulations for each of the subsequent lines of particle data in turn. With `repeatfirstline=on` the program will repeatedly use the first line of particle data, so that each simulation has the same initial conditions. This latter case is mainly used for generating reliable statistics when the program is used in QED mode. For the moment we will set `repeatfirstline=off`. The third line of the file must contain an integer between `1` and `99999`. This specifies the number of particles to be simulated. The remaining lines of the file give the input parameters for each of the particles in turn. The number of lines of input data must equal (or exceed) the integer specifying the number of runs. (Note that this is not necessary in the case where `repeatfirstline=on`, since then the program will repeatedly use the input parameters for the first particle in the list.) In the case where there are more lines of input than number of runs, the remaining lines will be ignored. The lines of input for each particle must contain the entrees shown in Table 3.1, all separated by commas. At the start of the simulation the particle will be at a distance $dist_0$ (in metres) from the coordinates (x_0, y_0, z_0). *It will be aiming for these coordinates, moving with an initial γ-factor of γ_0, such that it would reach this point at time $t = 0$ if there were no background fields. (The program determines the value of the time at*

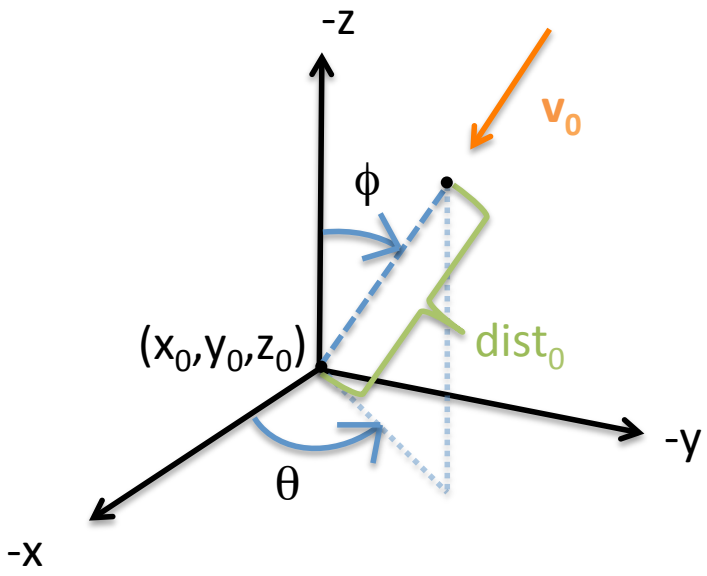

Figure 3.4: Diagram showing the orientation of the coordinate axes, their relation to the initial angles of the particle and the direction of the initial particle velocity.

the start of the simulation such that this is the case.) Moreover, the laser field propagates so that the centre of the focus reaches the coordinate origin, i.e., $x = y = z = 0$ *at time* $t = 0$*. If there are multiple laser fields specified, as will be the case in example 2, they propagate so that they all individually reach the origin at time* $t = 0$*.*

The particle species is specified by 'e' for electrons, 'p' for positrons and 'H+' for protons. The options for the equation of motion are currently 'lf' for Lorentz force, 'll', for Landau-Lifshitz and 'qed' for statistical photon emissions. Note that QED mode can only be initiated for electrons and positrons. Finally, the output flag works as follows. 't' specifies that the trajectory and velocity should be written to file while the electron is in the write box. 'ct' specifies that additionally the quantum efficiency parameter χ_e should be calculated for the trajectory. 'x' specifies that *no* information about the trajectory should be written to file. Even when 'x' is selected the final data will still be recorded when the particle leaves the simulation box and, additionally, any photon data from the QED runs will also still be written to file.

Let us choose our set up such that the particle is initially counter-propagating with the laser, $100\,\mu$m from the coordinate origin with an initial $\gamma_0 = 10$. Then our file *particle_input.csv* should look like that shown in Figure 3.5. The first entry in the fourth line is just a label: here we have used it to number the particle.

particle_input.csv

```
!no,species,theta,phi,dist0,x0,y0,z0,gama0,eom,tx
repeatfirstline=off
1
1,e,0,180,100e-6,0,0,0,10,lf,t
```

Figure 3.5: The file *particle_input.csv* for Example 1.

We have set $\theta_0 = 0°$, $\phi_0 = 180°$ so that the particle is initially travelling in the negative z-direction. Similarly, we have set $x_0 = 0$, $y_0 = 0$, $z_0 = 0$ because we want the particle to aim for the coordinate origin.

All that remains to do before we can start the simulation is to modify the file *input_setup.txt*, which specifies various properties of the simulation. We begin by defining the simulation box via the parameters `tmax`, `xmax`, `xmin`...`zmin`, with time in seconds and distances in metres. The run will terminate as soon as the particle leaves this box. (Note that we do not specify a minimum time for this box). Observe that we must set `zmax`$\geq 100\,\mu$m, otherwise the particle will not be in the simulation box at the start of the simulation (since we have already set $dist0 = 100\mu$m) and so the program will immediately terminate. In a similar manner we define the write box via `tmaxw`, `tminw`, `xmaxw`, `xminw`...`zminw`. Next we specify the numerical method (particle pusher) to be the leapfrog method, `solver=leapfrog`. Since the program has an adjustable time grid, we must specify the minimum and maximum permitted time steps (`mindt` and `maxdt`), the initial time step (`initialdt`) and the maximum permitted error at each time step (`grid_err_tol`). If the solver requires a time step smaller than `mindt` in order to keep the error above `grid_err_tol` then the program will terminate. It is also necessary to specify how frequently the trajectory and velocity data should be written to file. To write the data at every time step one should set `writeevery=1`. However, in most cases this will result in extremely large output files. Typically, a value around 1000 is appropriate. In this simulation we don't wish to output the field data to file so we set `outputintensity=off`. Finally, we are asked to specify the format of the trajectory files, which can either be `fileformat=txt` for ascii text format and `fileformat=bin` for binary format. Once completed, the file should look similar to Figure 3.6. Note how comments can be added by starting the line with an '`!`' and how the ordering is unimportant.

We are now ready to commence with the simulation! The program should be executed from the command line as detailed in Section 2.1. Once completed the following files will be created

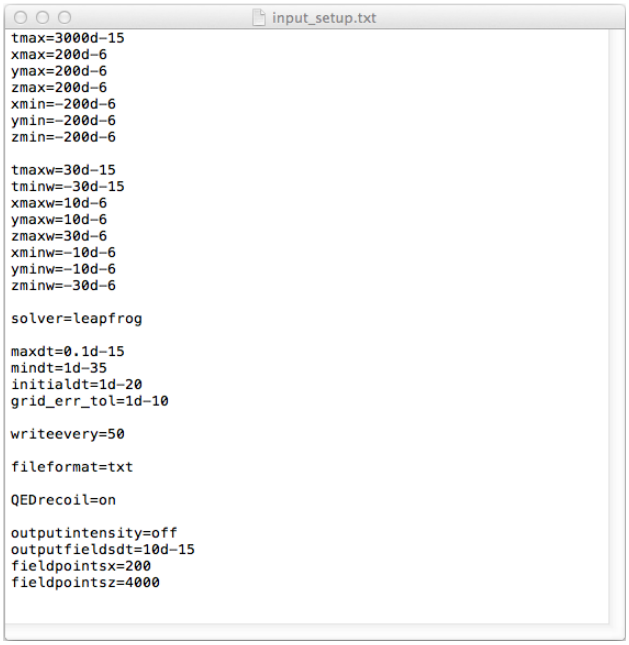

Figure 3.6: The file *input_setup.txt* for Example 1.

File name	Description
trajectories00001.dat	An 11 column tab-separated file containing the particle's orbit. The contents are as follows. Cols. 1-4: particle four-position, x_0, x_1, x_2, x_3. Cols. 5-8: particle four-velocity, γ, u_1, u_2, u_3. Cols. 9-10: χ_e and χ_γ if the program is being run in QED mode, else zeros. Col. 11: χ_e calculated using the full energy-momentum tensor expression if the flag 'c' is specified in the *particle_input.csv*, otherwise zeros.
final_data.dat	An 11 column tab-separated file containing the final data from each run. The columns are the same as for *trajectories00001.dat*, but each line contains the quantities for a single particle as it leaves the simulation box.

The data from the file *trajectories00001.dat* can be read in and plotted using the MATLAB script `simlaplot.m`. Note that since we have specified that the data is output in ascii text format, the MATLAB file must be edited to read `fileformat='txt'` at the beginning of the script. When the program is executed it will read in all the data, calculate the proper time vector τ for the particle and also plot the trajectory and energy, as shown in Fig. 3.7. Note that,

while the vectors are now stored in the memory ready for use, they are in natural units, so the position vector components are in units of eV^{-1} and the velocity vector in units of c.

Finally we can calculate the classical emission spectrum using the particle's orbit. After reading in the orbit data with the *simlaplot.m* routine, the spectrum may be calculated by running the MATLAB program *spectra.m*. The frequency range and observation angles can easily be changed by editing the script. The resulting spectrum is shown in Fig. 3.8. Note that it is crucially important that there are an adequate number of data points in the trajectory file. If the spectrum is noisy then the simulation should be rerun with a smaller value of the `writeevery` parameter. This becomes more and more essential for larger field strengths and for high intensity fields it may not be possible to get the spectrum to converge at all.

3.2 Example 2: Injection of electrons between colliding laser pulses

Simulate a beam of electrons injected into the middle of two colliding (focussed) laser pulses. Plot the trajectories and the background fields.

We are now ready to move to a slightly more sophisticated example which will demonstrate some additional features of SIMLA. We will consider the trajectories of several electrons as they are injected into the middle of two counter-propagating laser pulses. As well as plotting the trajectories of the particles, we will also plot the field background.

Once again our starting point is to create the input files, beginning with *particle_input.csv*. For the purposes of this example, our 'beam' of electrons will consist of five particles all propagating in the x-direction (so ϕ_0=90 degrees), but spread out with a 10 μm spacing in z. Thus we require five lines of particle data, as shown in Fig. 3.9.

Next we set up *input_fields.txt* to define the laser fields. Since we want to have two beams we set `no_fields=2`. They are both focussed fields so we set `field1=parax5, field2=parax5`. Note that for focussed fields it is also necessary to specify the waist diameter so, for e.g., 5 μm radius waists we set `waist1=5e-6, waist2=5e-6`. Finally, since we want the two beams to counter-propagate, we rotate one of them in the xz-plane by setting `anglexz1=0` for the first beam and `anglexz2=180` for the second beam (remember that the code propagates the lasers such that their centre focus reaches the coordinate origin at time $t = 0$). The resulting input file should look like that shown in Fig. 3.10.

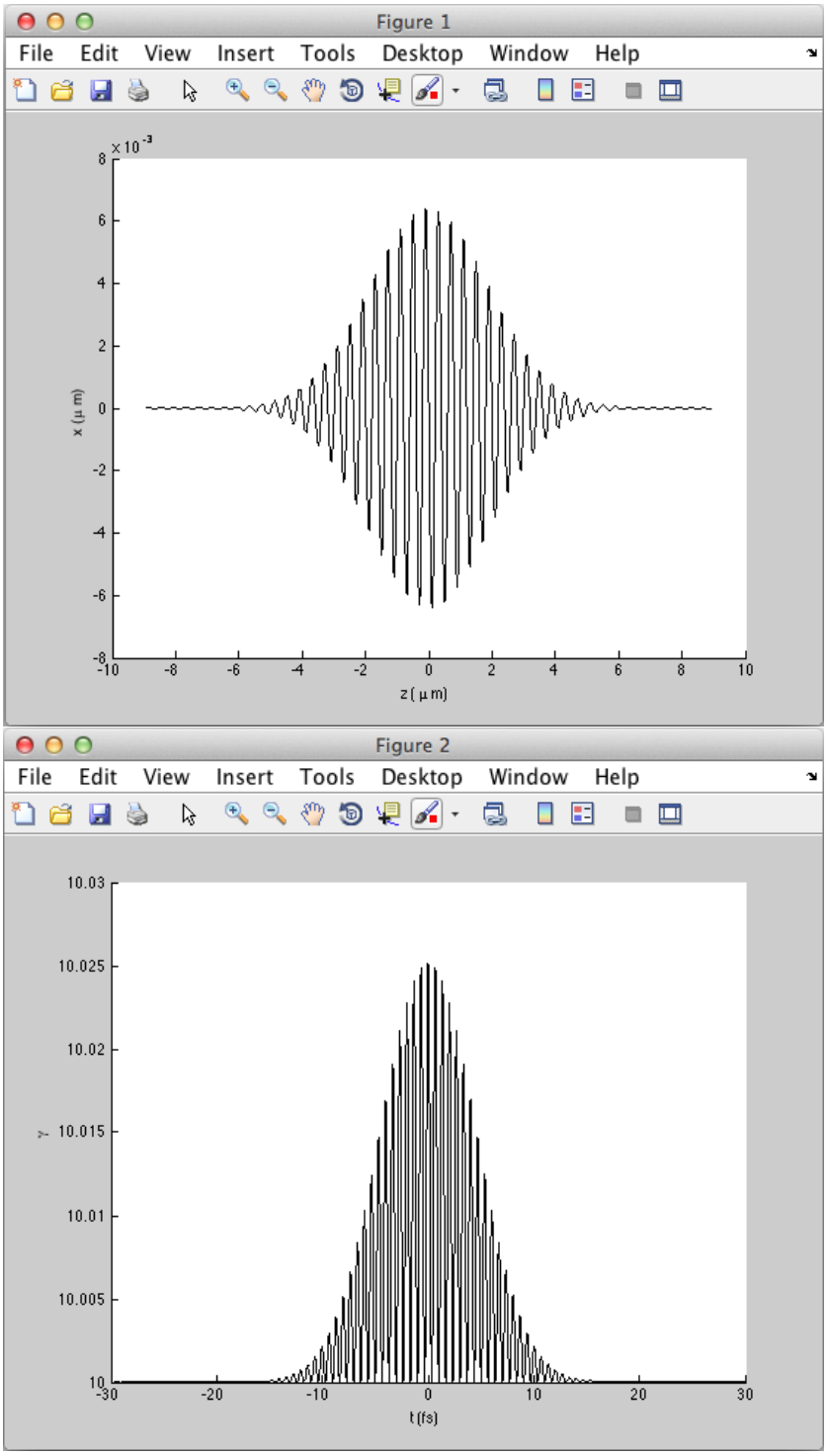

Figure 3.7: The MATLAB script *simlaplot.m* will by default plot the output trajectory (top panel) and energy (bottom panel) of the particle. The structure of the output files makes it is easy to plot of other quantities, e.g., the z coordinate of the particle as a function of time.

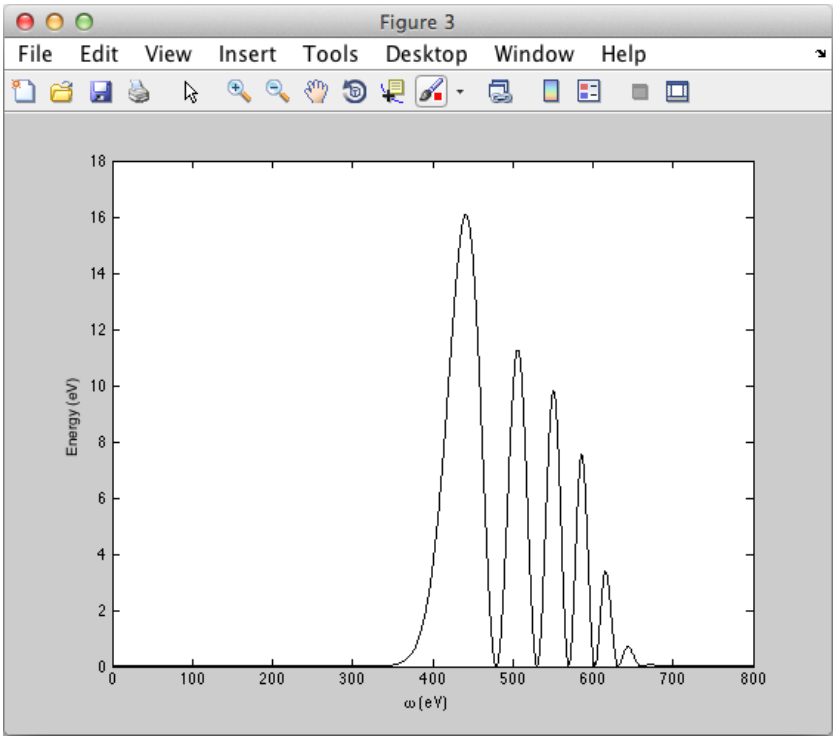

Figure 3.8: The emission spectrum for Example 1 plotted with the MATLAB script *spectrum.m*.

particle_input.csv

```
!no,species,theta,phi,dist0,x0,y0,z0,gama0,eom,tx
repeatfirstline=off
5
1,e,0,90,100e-6,0,0,-20e-6,25,lf,t
2,e,0,90,100e-6,0,0,-10e-6,25,lf,t
3,e,0,90,100e-6,0,0,0e-6,25,lf,t
4,e,0,90,100e-6,0,0,10e-6,25,lf,t
5,e,0,90,100e-6,0,0,20e-6,25,lf,t
```

Figure 3.9: The file *particle_input.csv* for Example 2.

input_fields.txt

```
no_fields=2
! none, crossed,linpw,circpw,parax1,parax5,constB
field1=parax5
field2=parax5

! inf,step,stepsin2,sech,gauss,gauss4,gauss8,gauss12
profile1=gauss
profile2=gauss

! NB no more than 4 digits before decimal
anglexz1=0
angleyz1=0
anglexy1=0

anglexz2=180
angleyz2=0
anglexy2=0

lambda1=0.8e-6
lambda2=0.8e-6

waist1=5e-6
waist2=5e-6

a0_1=10
a0_2=10

duration1=30e-15
duration2=30e-15
```

Figure 3.10: The file *input_fields.txt* for Example 2.

The setup file *input_setup.txt* for this example is shown in Fig. 3.11. It is very similar to that created for Example 1 (see Fig. 3.6) but with a few important differences. Firstly, in this example we are going to output the field intensity data so we have set `outputintensity=on`. Because this option has been selected we are required to specify the time interval between outputs of the field data. The program will begin outputting field data at the minimum time of the writebox, `tminw`, and finish outputting at `tmaxw`. Let us choose these to be -60 fs and 60 fs, respectively, and choose to output every 10 fs which we do by setting `outputfieldsdt=10e-15`. This will give us a total of 13 output files. The other two parameters we need to specify are the number of grid points in x and in z. For the outputting of the field data, these grid points will be evenly distributed over the area defined by the writebox. As the fields are both propagating along the z-axis, there will be more structure to capture in this direction than in the x-direction. Therefore we set `fieldpointsz=4000` but `fieldpointsx=200`. Finally, because we are not planning to calculate the classical emission spectra for the particles, we do not need such a high accuracy for the particle data as we did in Example 1. Thus we set `grid_err_tol=5e-10` and `writeevery=200`.

The next step is to execute the program which should produce five trajectory files as well as thirteen output files for the field intensity. We can then use the supplied MATLAB scripts to plot the field intensity and particle tracks at a given

input_setup.txt

```
tmax=3000d-15
xmax=200d-6
ymax=200d-6
zmax=200d-6
xmin=-200d-6
ymin=-200d-6
zmin=-200d-6

tmaxw=60d-15
tminw=-60d-15
xmaxw=10d-6
ymaxw=10d-6
zmaxw=30d-6
xminw=-10d-6
yminw=-10d-6
zminw=-30d-6

solver=leapfrog

maxdt=0.1d-15
mindt=1d-35
initialdt=1d-20
grid_err_tol=5d-10

writeevery=200

fileformat=txt

QEDrecoil=on

outputintensity=on
outputfieldsdt=10d-15
fieldpointsx=200
fieldpointsz=4000
```

Figure 3.11: The file *input_setup.txt* for Example 2.

time. Let's suppose we want to see how things look at time 30 fs. We first plot the field intensity for this time by running the script *simlafields.m*. This will begin by prompting the user to enter the maximum a0, which in this case is 10. It will then display the file numbers for the first two files and the final file, asking the user to specify which file to plot. We can determine from this information that the file for the 30 fs data is number 10, so we enter '10' at the prompt. The script will then generate a figure showing the field intensity in the xz-plane at 30 fs. Next we want to add the particle tracks to this figure. We do that by running the MATLAB script *simlafields_addtraj.m*. This will begin by prompting us to enter the number of tracks we want to add to the plot. In this case we want to add all five of them so we enter '5'. We are now asked to enter the file numbers for the five trajectories that we wish to plot. In this instance we enter each number '1', '2', '3', '4', '5', in turn. Finally the script will add the five trajectories to the intensity plot. If all has gone smoothly the MATLAB command window should look like that in the left hand plot of Fig. 3.12 and the figure window like that on the right. Observe how the particle tracks are incomplete, stopping to show the positions of the particles at 30 fs, rather than showing the full trajectories calculated during the simulation.

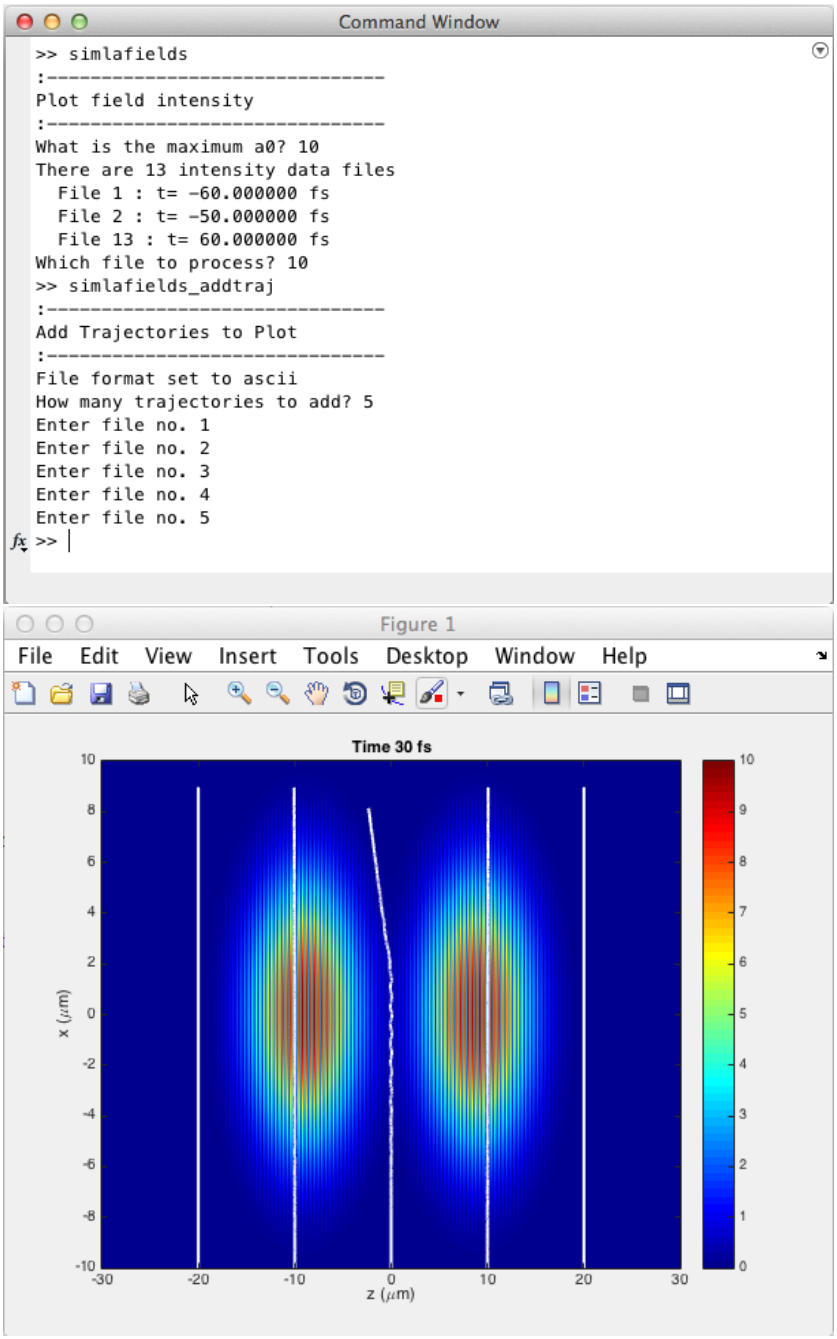

Figure 3.12: Top panel: the MATLAB command window for Example 2. Bottom panel: the MATLAB output produced in Example 2. It shows the intensity of two laser fields (colour plot, in units of a0) at $t = 30$ fs (propagating along z, initially towards each other) and trajectories of 5 particles incident perpendicular to the field (white lines).

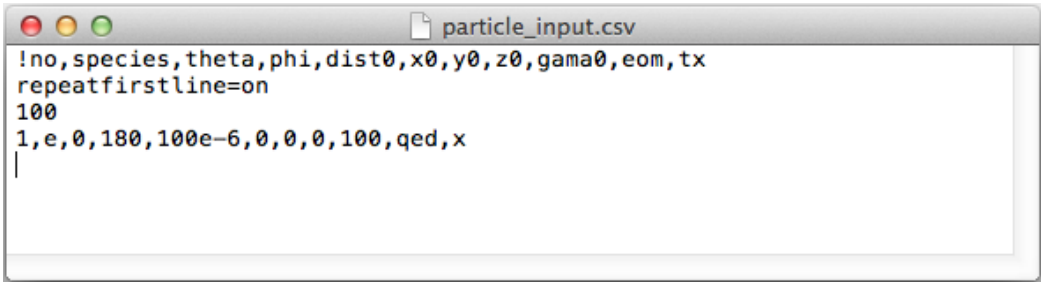

Figure 3.13: The file *particle_input.csv* for Example 3.

3.3 Example 3: Electrons in ultraintense laser fields treated via QED

Simulate an electron interacting with an intense plane-wave field. Generate statistics for QED photon emission. Compare trajectories to that of a classically radiating particle.

In this third example we will utilise the QED photon emission routines. Our aim is to generate the photon emission spectrum for an electron in an intense plane-wave field. This example is slightly different from the others due to its statistical nature. During each run only a few photons will be emitted by the particle, but the number and timing of the emissions, and the photon energies will be different each time. Therefore to generate a statistical photon distribution we have to run the program many times. There are two ways of doing this. One way would be to add hundreds of identical lines of initial conditions to the *particle_input.csv* file. The other way, which we adopt here, is to have the program repeat the simulation many times using just the first line of input. To do this we set `reapeatfirstline=on` in the second line of the file. Then, for each of the number of runs defined in the third line, the program will use the first line of particle data (i.e. the fourth line of the file) repeatedly. The format of the file is shown in Fig. 3.13. Note how we have set the equation of motion to `qed` and how, since we are only interested in the photon spectrum, we have set the output flag to `x` so that the trajectories will not be written to file. Finally, the file *input_fields.txt* is shown in Fig. 3.14. (The setup file *input_setup.txt* can be identical to that of Example 1 or 2.)

Once the program has been run the data from the file *photon.dat* can be read in to MATLAB using the script *photon.m*. This script will sort the energy and angular data into discrete bins and then produce plots of the resulting histogram data. The resulting plots for this example are shown in Fig. 3.15. The bin sizes can easily be altered by editing the MATLAB script.

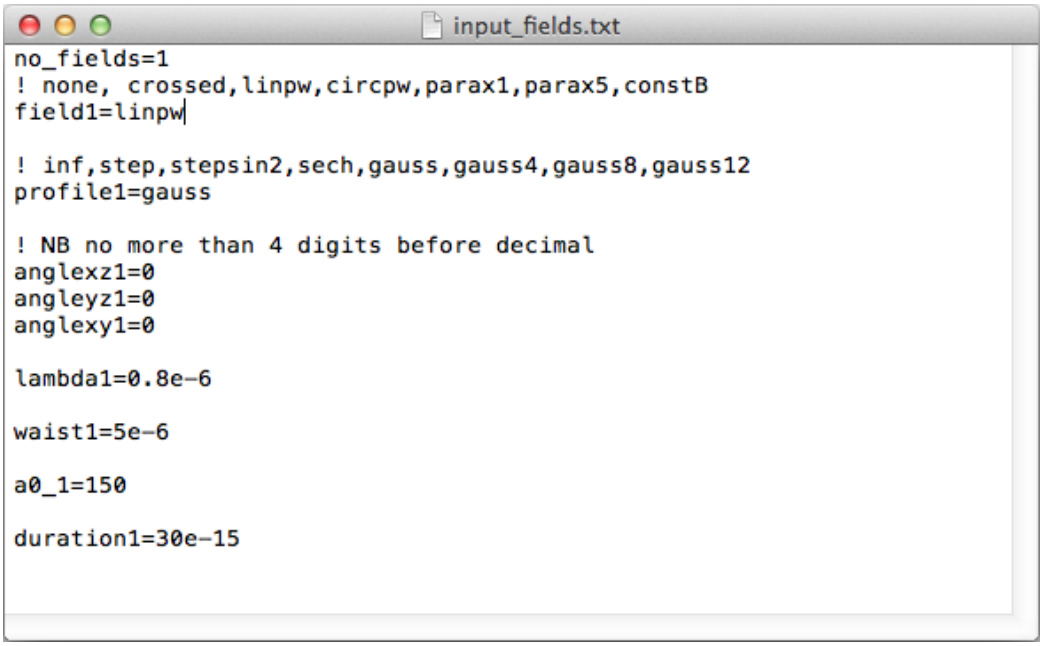

Figure 3.14: The file *input_fields.txt* for Example 3.

The user should now be fairly reasonably acquainted with the main features of SIMLA. Note that the default MATLAB scripts are set up to plot only certain quantities. Many more quantities of interest can be plotted. Examples of such alternative output are given in the corresponding SIMLA Computer Physics Communications paper.

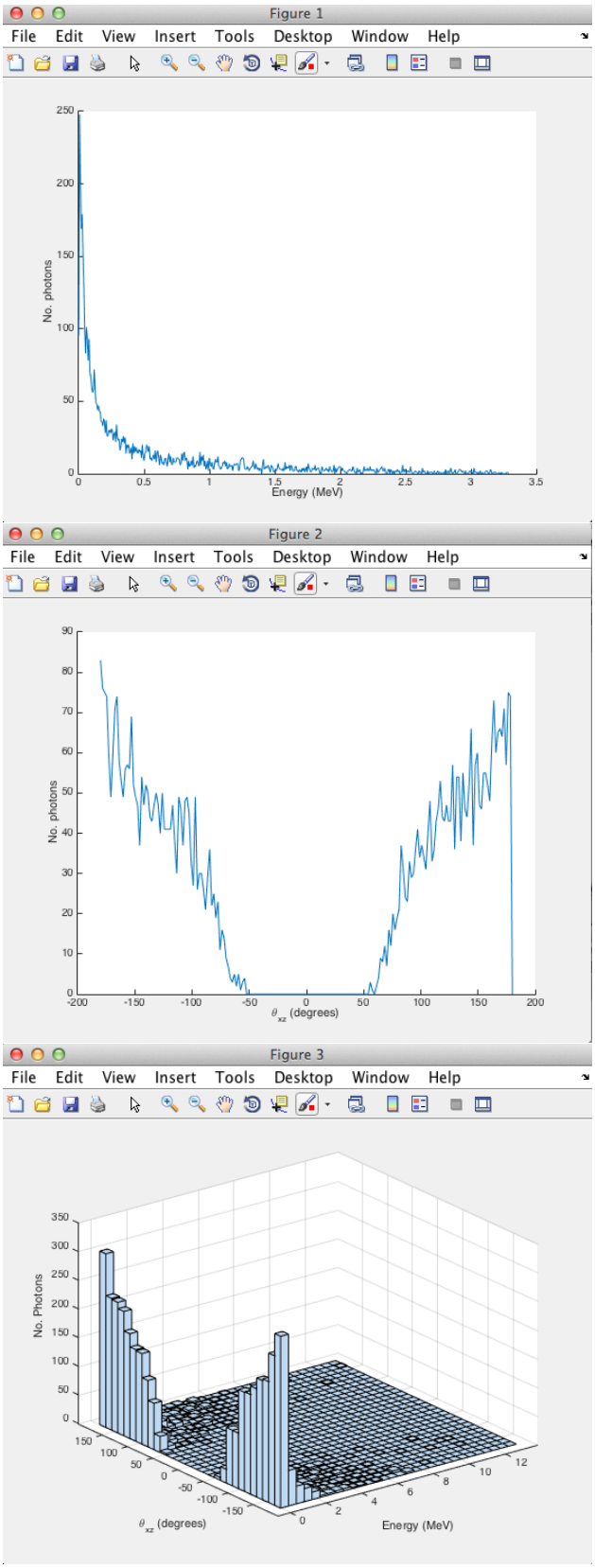

Figure 3.15: The figures produced by the MATLAB script *photon.m* for Example 3.

Chapter 4

Detailed Guide to Settings

4.1 Options for Setup File: *input_setup.txt*

4.1.1 Defining the Simulation Box

tmax the maximum time (in s) before the simulation terminates.

xmax the maximum value of the x-coordinate (in m) before the simulation terminates.

ymax the maximum value of the y-coordinate (in m) before the simulation terminates.

zmax the maximum value of the z-coordinate (in m) before the simulation terminates.

xmin the minimum value of the x-coordinate (in m) before the simulation terminates.

ymin the minimum value of the y-coordinate (in m) before the simulation terminates.

zmin the minimum value of the z-coordinate (in m) before the simulation terminates.

Please note that **tmin** is not a option.

4.1.2 Defining the Write Box

tminw the time (in s) at which the program starts writing the particle data to file.

xminw the value of the x-coordinate (in m) at which the program starts writing the particle data to file.

yminw the value of the y-coordinate (in m) at which the program starts writing the particle data to file.

zminw the value of the z-coordinate (in m) at which the program starts writing the particle data to file.

tmaxw the time (in s) at which the program stops writing the particle data to file.

xmaxw the value of the x-coordinate (in m) at which the program stops writing the particle data to file.

ymaxw the value of the y-coordinate (in m) at which the program stops writing the particle data to file.

zmaxw the value of the z-coordinate (in m) at which the program stops writing the particle data to file.

4.1.3 Settings for the Numerical Solver

solver the type of numerical method the program will use. Current options: `leapfrog`.

maxdt the maximum time step for the numerical solver (s).

mindt the minimum time step for the numerical solver (s). If a time step smaller than this is required in order to keep the errors within the specified limits then the program will terminate.

initialdt the initial time step for the numerical solver (s).

grid_err_tol the magnitude of the maximum permissible error at each time step.

writeevery number of time steps between writing data to file (must be an integer).

4.1.4 Miscellaneous Settings

fileformat the formatting of the trajectory files. Options: `bin` (binary), `txt` (ascii text).

QEDrecoil specifies whether the particle should recoil when emitting a photon (QED only). Options: `on`, `off`.

4.1.5 Options for Outputting the Field Data

outputintensity whether to create data files outputting the background field intensity at different times. The region outputted will be the y-plane. Options: `on`, `off`.

outputfieldsdt the timestep (in s) between subsequent outputs of the field intensity. The start time and end time correspond to the values of the write box, `tminw`, `tmaxw`.

fieldpointsx the number of grid points for the field intensity output in the x-direction. The total range in x will be the same as for the write box, `xminw` to `xmaxw`.

fieldpointsz the number of grid points for the field intensity output in the z-direction. The total range in z will be the same as for the write box, `zminw` to `zmaxw`.

4.2 Options for Fields File: *input_fields.txt*

no_fields the number of different background fields. Must be an integer between 0 and 9.

field1, . . . , field9 the type of each of the fields. The minimum number of fields which must be designated is `no_fields`. Fields of higher numbers may be designated in the input file but will be ignored by the program. Options: `none`, `crossed` (constant crossed fields), `linpw` (linear polarised plane wave), `circpw` (circularly polarised plane wave), `parax1` (first order paraxial), `parax5` (fifth order paraxial).

profile1, . . . , profile9 the profile of each of the fields. The minimum number of profiles which must be designated is `no_fields`. Profiles corresponding to fields of higher numbers may be designated in the input file but will be ignored by the program. Options: `inf`, `step`, `sech`, `gauss`, `gauss4`, `gauss8`, `gauss12`.

anglexz1, . . . , anglaxz9 the angles (in degrees) of the rotations in the xz-plane for each of the fields. Note that the angle must be between -9999 and 9999 degrees, otherwise it will not be read in correctly from the input file.

angleyz1, . . . , anglayz9 the angles (in degrees) of the rotations in the yz-plane for each of the fields. Note that the angle must be between -9999 and 9999 degrees, otherwise it will not be read in correctly from the input file.

anglexy1, . . . , anglaxy9 the angles (in degrees) of the rotations in the xy-plane for each of the fields. Note that the angle must be between -9999 and 9999 degrees, otherwise it will not be read in correctly from the input file.

lambda1, . . . , lambda9 the wavelengths (in m) of each of the fields (if applicable). Any unneeded allocations will be ignored.

waist1, . . . , waist9 the waist radii (in m) of each of the fields (if applicable). Any unneeded allocations will be ignored.

a0_1, . . . , a0_9 the dimensionless intensity of each of the fields (if applicable). Any unneeded allocations will be ignored.

fieldstrength1, . . . , fieldstrength9 the strength of the fields (in eV^2) in the case of constant/crossed fields. Any unneeded allocations will be ignored.

duration1, . . . , duration9 the durations (FDHM where applicable) of each of the time-dependent fields (in s). Any unneeded allocations will be ignored.

Chapter 5

Under the Bonnet

5.1 Understanding the code

The SIMLA code is written in Fortran 90. Its structure is designed to be fully modular, allowing anyone with basic programming knowledge to easily change specific aspects without having to alter the entire code. For example, alternative numerical methods and, e.g., alternative equations of motion can be inserted by adding the appropriate subroutines.

One important feature of the program is that it uses an adjustable time step. This is one reason why, in cases where collective effects can be neglected, the code is superior to particle-in-cell (PIC) programs, where such a feature is not possible. The user specifies in the input files the maximum permissible error in the norm of the spatial coordinates that can be introduced at each time step. The program then propagates the particle through the simulation, making continuous adjustments to the time step such that the error condition is maintained. Thus the grid points are not uniform; there will be very few time steps in regions where the background field is weak, while there will be many in more intense/complex regions of the field. We have found that the implementation of the adjustable grid typically decreases the computation time by around two orders of magnitude compared to a fixed size grid. A flow chart showing how the method is implemented is given in Fig. 5.1.

Finally, some remarks about the normalisations adopted in the code. In codes such as these it is common to use dimensionless units, usually by normalising the field amplitudes in terms of a_0. However, such normalisation was found to be impractical in our case, where we allow multiple fields with different frequencies and also constant fields that do not have a time dependence. Thus it was decided that the code should use natural units (Lorentz-Heaviside), obtained by setting $\hbar = c = 1$. The various quantities in the code therefore have the units specified

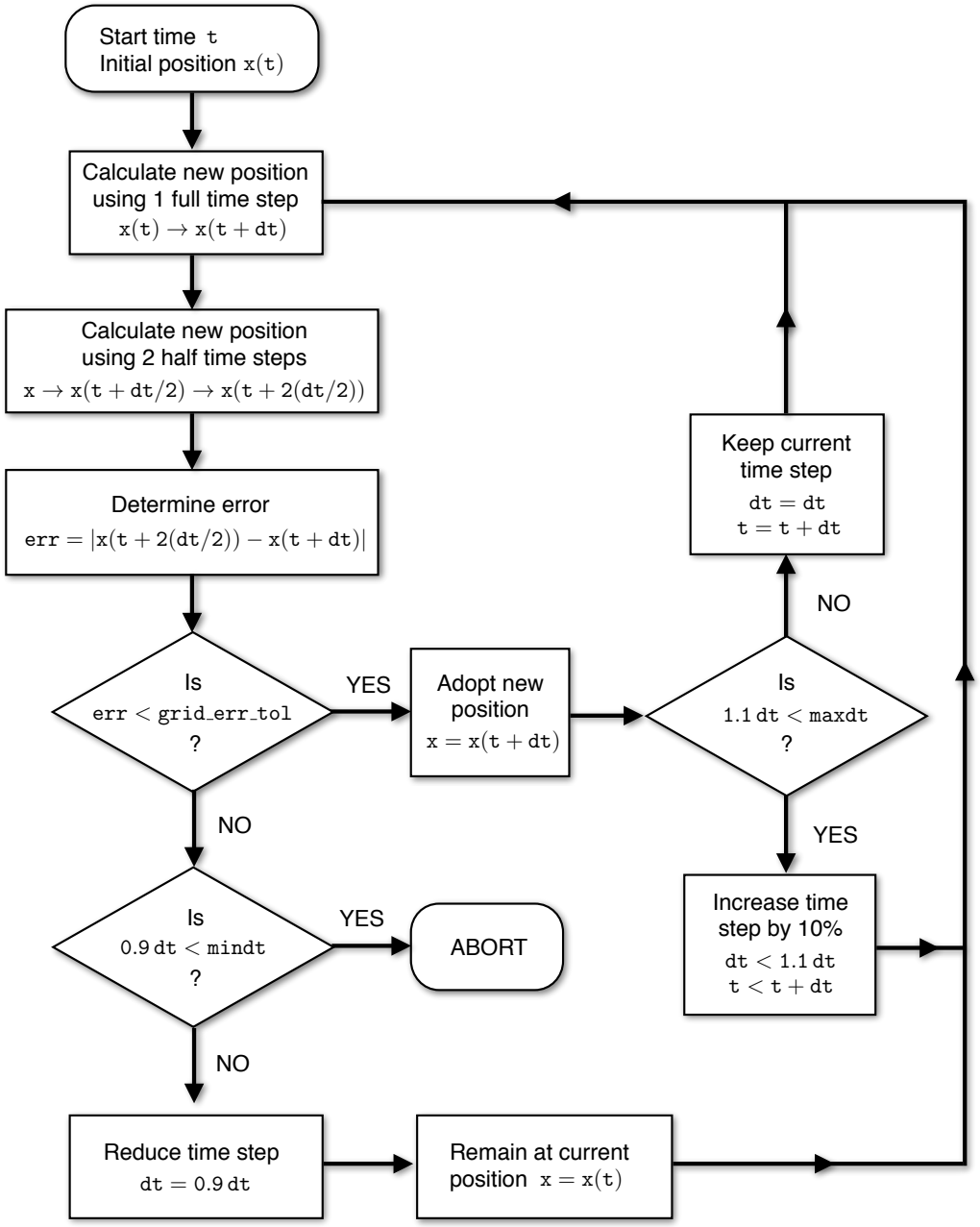

Figure 5.1: Flow chart showing the operation of the adjustable time step.

Table 5.1: The units used internally by the code.

Quantity	Units
Length	eV^{-1}
Time	eV^{-1}
Mass	eV
Charge	$e/\sqrt{4\pi\alpha}$

in Table 5.1. Note, however, that these units are *not* the same as the units of the input variables which should be defined according to the conventions specified in Chapter 4.

5.2 Overview of the subroutines

main_subroutine This is the main subroutine of the program. It is called once for each particle listed in the file *particle_input.csv*. The subroutine calls the solver routines to calculate the trajectory of the particle and operates the adjustable time grid. It monitors the feedback from the rest of the subroutines, looking for error reports and, upon detecting an error, aborts the run and calls the error routines. The routine determines if the particle is in the writebox and, if so, writes the trajectory data to file.

read_input_files This subroutine reads in the contents of *input_fields.txt* and *input_setup.txt* and then assigns this data to the appropriate variables. It is only called once at the beginning of the simulation.

leapfrog_solver This subroutine propagates the particle forward one time step using leapfrog integration.

Lorentz_force Determines the acceleration on the particle using the Lorentz equation.

Landau_Lifshitz Determines the acceleration on the particle using the Landau-Lifshitz equation, which takes into account radiation reaction.

fields Calculates the total field strength (superposition of all the background fields) at the current time and spatial position.

chicalc Calculates the quantum efficiency parameter χ for the particle using the full expression in terms of the energy momentum tensor.

record_fields Subroutine to record the field data (intensity) as a function of the x and z spatial coordinates.

errors This subroutine is called when the program encounters an error. The subroutine which detects the problem will pass the relevant information the main program which then passes it to this routine. Information about the error will be displayed on the screen and more detailed data will be written to a text file *error_reportXXXXX.txt*, where *XXXXX* is the particle number.

gammatablesub Calculates the probability rate $\Gamma(t)$ (a function of χ_e) for a given χ_e (more precisely, it is the product $\gamma_e\Gamma(t)$ that is calculated).

qedroutines Manages the photon event generation. Calls subroutines to determine if an emission should occur and, if so, its energy. Updates the electron momentum.

chi Calculates χ_e using the crossed field approximation and, if applicable, calculates χ_γ and determines the photon momentum $\mathbf{k}$.

egen_sub Determines whether an emission event should occur in a given time step.

Gammatinterp The differential probability of photon emission (at a given time t) is $dW = \Gamma(t)dt$. This subroutine calculates $\Gamma(t)$

$$\Gamma(t) = \int_{\chi_{\gamma(\mathrm{cutoff})}}^{\chi_e} d\Gamma = \int_{\chi_{\gamma(\mathrm{cutoff})}}^{\chi_e} \left(\frac{d\Gamma}{d\chi_\gamma}\right) d\chi_\gamma$$

where $d\Gamma$ is given by Eq. 2.5. Formally, the lower limit of the integral should be zero. In practice, however, we use the lower cutoff $\chi_{\gamma(\mathrm{cutoff})}$. Its value is specified in the `simla.f90` program. The emission of soft photons with energy below this cut off does not appreciably affect the electron dynamics [14].

Gammatinterpintegral Interpolates the integral above.

d2pdXdt Calculates $d\Gamma/d\chi_\gamma (= d^2P/d\chi_\gamma dt)$, the differential probability rate of a single photon emission per unit χ_γ.

besselK Uses Simpson integration to evaluate $K_{1/3}(z)$ or $K_{2/3}(z)$.

intbesselk Calculates the integral

$$\int_{\text{lower limit}}^{\infty} K_{n/3}(x)dx.$$

Bibliography

[1] T. Heinzl and A. Ilderton, Optics Communications **282**, 1879 (2009).

[2] H. A. Lorentz, *The Theory of Electrons* (Teubner, Leipzig (1905), reprinted by Dover Publications, New York, (1952) and Cosimo, New York, (2007)).

[3] M. Abraham, *Theorie der Elektrizität* (Teubner, Leipzig, 1905).

[4] P. A. M. Dirac, Proc. R. Soc. Lond. A. **167**, 148 (1938).

[5] L. D. Landau and E. M. Lifshitz, *The Classical Theory of Fields, Course of Theoretical Physics Vol. 2* (Butterworth-Heinemann, Oxford, 1987).

[6] I. V. Sokolov, N. M. Naumova, J. A. Nees, G. A. Mourou, and V. P. Yanovsky, Phys. Plasmas **16**, 093115 (2009).

[7] R. F. O'Connell, Contemp. Phys. **53**, 301 (2012).

[8] V. S. Krivitski and V. N. Tsytovich, Sov. Phys. Usp. **34**, 250 (1991).

[9] A. Ilderton and G. Torgrimsson, Phys. Lett. **B725**, 481 (2013).

[10] F. Sauter, Z. Phys. **69**, 742 (1931).

[11] W. Heisenberg and H. Euler, Z. Phys. **98**, 714 (1936).

[12] J. Schwinger, Phys. Rev. **82**, 664 (1951).

[13] V. Ritus, J. Sov. Laser Res. **6**, 497 (1985).

[14] R. Duclous, J. G. Kirk, and A. R. Bell, Plasma Phys. Contr. F. **53**, 015009 (2011).

[15] J. D. Jackson, *Classical Electrodynamics* (John Wiley and Sons, 1999).

[16] C. Harvey, T. Heinzl, and A. Ilderton, Phys. Rev. A **79**, 063407 (2009).

[17] A. R. Bell and J. G. Kirk, Phys. Rev. Lett. **101**, 200403 (2008).

[18] I. V. Sokolov, N. M. Naumova, J. A. Nees, and G. A. Mourou, Phys. Rev. Lett. **105**, 195005 (2010).

[19] N. V. Elkina, A. M. Fedotov, I. Y. Kostyukov, M. V. Legkov, N. B. Narozhny, E. N. Nerush, and H. Ruhl, Phys. Rev. ST Accel. Beams **14**, 054401 (2011).

[20] C. Harvey, A. Ilderton, and B. King, (2014), arXiv:1409.6187 [physics.plasm-ph] .

[21] K. T. McDonald, Unpublished notes: http://www.hep.princeton.edu/∼mcdonald/accel/gaussian.ps.

[22] L. W. Davis, Phys. Rev. A **19**, 1177 (1979).

[23] J. P. Barton and D. R. Alexander, Journal of Applied Physics **66**, 2800 (1989).

[24] Y. I. Salamin, G. R. Mocken, and C. H. Keitel, Phys. Rev. ST Accel. Beams **5**, 101301 (2002).