MAX-PLANCK-INSTITUT FÜR QUANTENOPTIK

LPIC++

A Parallel One-dimensional Relativistic Electromagnetic Particle-In-Cell Code for Simulating Laser–Plasma–Interaction

Roland Lichters, Robert E. W. Pfund, and Jürgen Meyer-ter-Vehn

Max-Planck-Institut für Quantenoptik
Hans-Kopfermann-Straße 1
D-85748 Garching, Germany
fax: +49-89-32905200
email: meyer-ter-vehn@mpq.mpg.de
pfund@mpq.mpg.de

MPQ 225 August 1997

Dieser MPQ–Bericht ist als Manuskript des Autors gedruckt
Alle Rechte vorbehalten

This MPQ–Report has been printed as author's manuscript
All rights reserved

Max–Planck–Institut für Quantenoptik
85740 Garching, Bundesrepublik Deutschland

MAX-PLANCK-INSTITUT FÜR QUANTENOPTIK

LPIC++

A Parallel
One-dimensional Relativistic Electromagnetic
Particle-In-Cell Code
for Simulating
Laser–Plasma–Interaction

Roland Lichters, Robert E. W. Pfund, and Jürgen Meyer-ter-Vehn

Max-Planck-Institut für Quantenoptik
Hans-Kopfermann-Straße 1
D-85748 Garching, Germany
fax: +49-89-32905200
email: meyer-ter-vehn@mpq.mpg.de
pfund@mpq.mpg.de

MPQ 225 August 1997

Contents

1 Introduction

Particle-In-Cell (PIC) codes are well established tools for kinetic simulations in plasma physics and astro physics. In recent years, the progress in producing intense ($I > 10^{18}\mathrm{Wcm}^{-2}$) ultra-short (< 100fs) laser pulses [1] asks for improved kinetic descriptions of the interaction of such laser pulses with plasmas, since high intensities, short time scales and large density gradients occuring lead to the failure of conventional hydrodynamic approaches assuming nonrelativistic dynamics, local thermodynamic equilibrium, etc. New insight has meanwhile been obtained with the guidance of numerical kinetic treatments, mainly PIC simulations, in the field of absorption of short laser pulses [2, 3, 4], the propagation of short pulses in underdense plasma, wake field generation, fast electron production [5, 6, 13], magnetic field generation [6], harmonic generation at overdense plasma surfaces [7, 8, 9, 10], and even in a new discipline of inertial confinement fusion (ICF), the fast ignitor concept [11, 12, 13].

The code `LPIC++` presented here, is based on a one-dimensional, electromagnetic, relativistic PIC code that has originally been developed by one of the authors [14] during a PhD thesis at the *Max-Planck-Institut für Quantenoptik* for kinetic simulations of high harmonic generation from overdense plasma surfaces [10]. The code uses essentially the algorithm of Birdsall and Langdon [15] and Villasenor and Bunemann [16]. It is written in C++ in order to be easily extendable and has been parallelized to be able to grow in power linearly with the size of accessable hardware, e.g. massively parallel machines like Cray T3E. The parallel `LPIC++` version uses `PVM` for communication between processors. `PVM` is public domain software, can be downloaded from the world wide web [17].

A particular strength of `LPIC++` lies in its clear program and data structure, which uses chained lists for the organization of grid cells and enables dynamic adjustment of spatial domain sizes in a very convenient way, and therefore easy balancing of processor loads. Also particles belonging to one cell are linked in a chained list and are immediately accessable from this cell. In addition to this convenient type of data organization in a PIC code, the code shows excellent performance in both its single processor and parallel version.

It is the central aim of this paper to make the code available to those interested in the simulation of laser-plasma interaction. This manual is mainly intended to explain

- how to receive and install (section 2)
- how to use (sections 3 and 4)

the code LPIC++. It does not give a complete introduction to plasma simulation using particle-in-cell codes. Section 4 presents a collection of test cases, which help in getting used to the code. Sections 5 and 6 finally deal with the algorithm on which the code is based, the data and code structure.

2 Receiving and Installing

The source code distribution contains a README file in ASCII format and a compressed tar-file `lpic-1.0.tar.gz` (550 kByte). Since LPIC++ is written in C++, it is supposed to run on any Unix platform like Linux, Solaris, SunOS, AIX, etc. It has been tested and used extensively under AIX.

2.1 Plain version

How to install the single processor version of LPIC++:

1. Unpack the archive:
   Copy `lpic-1.0.tar.gz` to the directory where you would like to install LPIC++, change to this directory and type

   ```
   gunzip lpic-1.0.tar.gz
   tar -xvf lpic-1.0.tar
   ```

   This will create a directory `pic/` with subdirectories `pic/data/`, `pic/doc/`, `pic/lpic/` and `pic/post/`. `lpic/` contains the LPIC++ source code `src/*.C` with Makefile `src/Makefile`, include files `src/include/*.h`, input files `input/*`, an empty object directory `obj/` and an empty data directory `data/`. `doc/` contains the documentation `lpic.ps`. `post/` contains the source code for the postprocessor.

   In addition you will find directories `pic/idl/` and `pic/fresnel/`. For explainations see sections 3 and 4, respectively.

2. Compile LPIC++:
   Change to directory `pic/` and type

   ```
   make all
   ```

   This produces the LPIC++ executable `lpic_plain` in directory `pic/lpic/` and the postprocessor executable `postprocessor` in directory `pic/post/`.

   Alternative:

- Change to directory `pic/lpic/src/`. You may have to edit the `Makefile` in order to choose your favourite compiler and the corresponding compiler options. Default is the GNU C compiler ('CC=g++'). Alternatively, you may insert the AIX C compiler ('CC=xlC'). Type

  ```
  make plain
  ```

  This produces the executable `pic/lpic/lpic_plain`.

- Compile the postprocessor:
  Change to directory `pic/post/src/`, edit the `Makefile` and type

  ```
  make post
  ```

  This produces the executable `pic/post/postprocessor`.

3. Run an example:
   Change to directory `pic/lpic/` and type

   ```
   lpic_plain input/input.performance
   ```

   Output will be written by default to directory `pic/data/`.

2.2 Parallel version

The parallel version of LPIC++ uses the public domain software PVM [17] for communication between processors. Unless not already installed on your system(s), you can download the software, its documentation and instructions for installing from the web site `http://www.netlib.org/pvm3`. PVM is also available on Cray T3E.
After installing PVM you can also compile the parallel version of LPIC++:
Change to directory `pic/lpic/src/` and type

```
make parallel
```

This produces the executable `pic/lpic/lpic_parallel`. You need not remove the plain version and its object files before.

The programs, their input and output will be described in detail in the following section.

3 User's Guide

This section deals with in- and output to and from LPIC++ and its postprocessor. LPIC++ (in both versions) reads input from an input file, e.g. in directory `pic/lpic/input/`, which has to be specified as an argument on the commandline when calling LPIC++, e.g.

```
lpic_plain input/input.fresnel.0
```

If LPIC++ is called without arguments, the input file `pic/lpic/input/input.list` is read by default. In the following section the input parameters are described in detail.

3.1 Input for LPIC++

The parameters are discussed following the example in file `input/input.fresnel.0`, see Figs. 1 and 2.
Groups of parameters follow typical key words like '&pulse_front' or '&electrons'. There are six major groups of input parameters concerning

1. laser pulses (&pulse_front, &pulse_rear)
2. simulation time (&propagate)
3. simulation box, plasma shape and density (&box)
4. particle species (&electrons, &ions)
5. diagnostics (&output, &energy, &flux, ...)
6. technical stuff (&restart, &parallel)

The key words starting with an '&' and **parameter names** like **'amplitude'** should *not* be changed, since the code uses them to identify parameter values in the input file. Only numbers should be touched unless you want to change the input routines (see section 6).

3.1.1 Laser pulses (&pulse_front, &pulse_rear)

The plasma can be irradiated with laser pulses from two sides, from the left or front side (&pulse_front) and the right or rear side (&pulse_rear). The pulses enter the simulation box at the left and right hand boundary at time $t = 0$.

```
////////////////////////////////////////////////////////////////////////////////////////
//
//  input for lpic
//
////////////////////////////////////////////////////////////////////////////////////////

&pulse_front
------------------------------------------------------------------------------------------
Q               = 1          # switch ON (Q=1) OFF (Q=0)
amplitude       = 0.01       # dimensionless laser field amplitude
amplitude2      = 0.0        # dimensionless laser field amplitude, 2nd harmonic
amplitude3      = 0.0        # dimensionless laser field amplitude, 3rd harmonic
phase2          = 0          # 2nd harmonic's phase with respect to fundamental [degree]
phase3          = 0          # 3rd harmonic's phase [degree]

angle           = 0          # in degree
polarization    = 1          # s=1, p=2, c=3
shape           = 2          # linear=1, sin=2, sin^2=3
raise           = 0          # pulse raise/fall time in periods
duration        = 60         # pulse duration in periods
pulse_save      = 1          # save pulse shape? yes=1, no=0
pulse_save_step = 0.02       # time step in periods


&pulse_rear
------------------------------------------------------------------------------------------
Q               = 0          # switch ON (Q=1) OFF (Q=0)
amplitude       = 0.0        # dimensionless laser field amplitude
amplitude2      = 0.0        # dimensionless laser field amplitude, 2nd harmonic
amplitude3      = 0.0        # dimensionless laser field amplitude, 3rd harmonic
phase2          = 0          # 2nd harmonic's phase with respect to fundamental
phase3          = 0          # 3rd harmonic's phase

angle           = 0          # in degree
polarization    = 1          # s=1, p=2, c=3
shape           = 2          # linear=1, sin=2, sin^2=3
raise           = 0          # pulse raise/fall time in periods
duration        = 60         # pulse duration in periods
pulse_save      = 1          # save pulse shape? yes=1, no=0
pulse_save_step = 0.02       # time step in periods

&propagate
------------------------------------------------------------------------------------------
prop_start      = 0          # start time in periods
prop_stop       = 30         # stop time in periods


&box
------------------------------------------------------------------------------------------
cells_per_wl    = 50         # cells per wavelength, lab frame
cells           = 1700       # total number of cells
cells_left      = 100        # cells vacuum left
cells_plasma    = 1500       # occupied cells for cells_ramp=0
cells_ramp      = 0          # cells in the linear ramp region
n_ion_over_nc   = 0.64       # maximum density/critical density
box_save        = 1          # save configuration? yes=1, no=0


&electrons
------------------------------------------------------------------------------------------
fix         = 0              # 0->not fixed   1->fixed
ppc         = 20             # max. number of MacroParticles per cell
vtherm      = 0.00           # thermal velocity in units of C


&ions
------------------------------------------------------------------------------------------
fix         = 1              # 0->not fixed   1->fixed
z           = 1              # q/e
m           = 50000          # m/m_e
ppc         = 10             # max. number of MacroParticles per cell
```

Fig. 1: *The input file* `pic/lpic/input/input.fresnel.0`, *part I.*

```
vtherm      = 0.00           # thermal velocity in units of C

&output
------------------------------------------------------------------------------------------
path = ../data               # output path

&energy
        Q           = 1      # energy plot?
        t_start     = 0      # start time in periods
        t_stop      = 100    # stop time in periods
        t_step      = 0      # time step in periods

&flux
      Q             = 1      # flux plot?
      t_start       = 0      # start time in periods
      t_stop        = 100    # stop time in periods
      t_step        = 0      # time step in periods

&reflex
        Q           = 1      # reflectivity plot?
        t_start     = 0      # start time in periods
        t_stop      = 100    # stop time in periods
        t_step      = 1      # time step in periods

&snapshot
        Q           = 0      # snapshots?
        t_start     = 0      # start time in periods
        t_stop      = 100    # stop time in periods
        t_step      = 10     # time step in periods

&el_phasespace
        Q           = 0      # phasespace plots?
        t_start     = 0      # start time in periods
        t_stop      = 20     # stop time in periods
        t_step      = 2      # time step in periods

&ion_phasespace
        Q           = 0      # phasespace plots?
        t_start     = 0      # start time in periods
        t_stop      = 20     # stop time in periods
        t_step      = 2      # time step in periods

&el_velocity
        Q           = 0      # electron velocity distributions?
        t_start     = 0      # start time in periods
        t_stop      = 100    # stop time in periods
        t_step      = 10     # time step in periods

&ion_velocity
        Q           = 0      # ion velocity distributions?
        t_start     = 0      # start time in periods
        t_stop      = 100    # stop time in periods
        t_step      = 10     # time step in periods

&de
     Q              = 0      # electron density plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&di
     Q              = 0      # ion density plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&jx
     Q              = 0      # jx plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
```



```
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&jy
     Q              = 0      # jy plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&jz
     Q              = 0      # jz plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&ex
     Q              = 0      # ex plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&ey
     Q              = 0      # ey plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&ez
     Q              = 0      # ez plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&bx
     Q              = 0      # bx plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&by
     Q              = 0      # by plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&bz
     Q              = 0      # bz plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&edens
     Q              = 0      # field energy density plots?
     t_start        = 10     # start time in periods
     t_stop         = 20     # stop time in periods
     x_start        = 0      # left boundary in cells
     x_stop         = 500    # right boundary in cells

&traces
        Q           = 1      # traces?
        t_start     = 0      # start time in periods
        t_stop      = 100    # stop time in periods
        traces      = 5      # # of traces at fixed positions x:
                t0=2, t1=100, t2=850, t3=1600, t4=1698
```

Fig. 2: *The input file* `pic/lpic/input/input.fresnel.0`, *part II.*

```
////////////////////////////////////////////////////////////////////////////////////////

&restart
------------------------------------------------------------------------------------------
Q           = 0              # restart from intermediate stage?
file        = restart        # start file
Q_save      = 1              # save intermediate stages periodically?
file_save   = restart        # save file


&parallel
------------------------------------------------------------------------------------------
N_domains   = 1              # number of parallel processes
Q_reo       = 1              # periodic reorganizations?
delta_reo   = 1              # laser cycles between reo's

////////////////////////////////////////////////////////////////////////////////////////


thermal velocity v/c vs. thermal energy [eV]   M ( vx^2 + vy^2 + vz^2 ) = 3 k T
------------------------------------------------------------------------------------------
1e-4          0.005
3e-4          0.046
5e-4          0.128
1e-3          0.511
4e-3          8.2
1e-2         51
2e-2        204
3e-2        460
4e-2        818
4.42e-2    1000
1e-1       5110

choose:  v_th/c = (lambda_d/dx)  *  (omega_p/omega)        *       (2pi dx/lambda)
                          :=1        = sqrt(z_ion * n_ion_over_nc)  = 2pi/cells_per_wl

////////////////////////////////////////////////////////////////////////////////////////
```

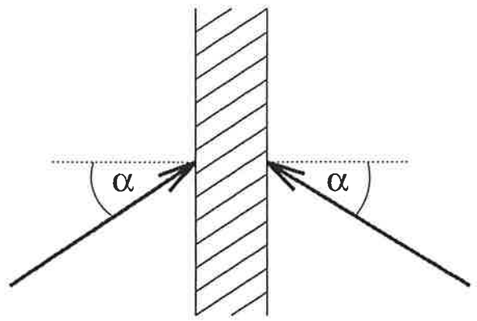

Fig. 3: *Angle of incidence with respect to the normal.*

The first pulse parameter **Q** is used to switch the laser on (**Q**=1) or off (**Q**=0).
The field **amplitude** a_0 is given in dimensionless units

$$a_0 = \frac{eE_0}{m_e \omega c}, \tag{1}$$

where E_0 ist the amplitude of the electric laser field, e, m_e, c are electron charge, mass and velocity of light in vacuum, respectively, and ω denotes the laser frequency. Intensity is expressed in these units as

$$I\lambda^2 = a_0^2 \times 1.37 \cdot 10^{18} W\mu m^2/cm^2. \tag{2}$$

Amplitude $a_0 \approx 1$ means, that a single electron in such a laser field will be relativistic, move with velocity close to the velocity of light and kinetic energy close to its rest energy $m_e c^2$. The incident laser pulses can be chosen to contain a mixture of frequencies: The fields specified by **amplitude2** and **amplitude3** are given in the same units as the fundamental field (**amplitude**) and denote contributions oscillating at 2ω and 3ω, respectively. The parameters **phase2** and **phase3** are the phases of the 2ω and 3ω frequency components with respect to the fundamental component ω and have to be given in degrees.
Notice, that the **angle**s of incidence (in degrees), see Fig. 3, cannot be chosen independently because of the one-dimensional structure of the code [1]. In case they are different and both pulses are switched on, &pulse_front has priority and the rear pulse's angle

[1] Oblique incidence is incorporated in the 1d code using Bourdier's method [18]. A Lorentz transformation is performed to a frame of reference M which moves in the plane of incidence parallel to the plasma surface such that the pulse is normally incident in frame M. This is achieved with a frame velocity $v = c \sin\alpha$ in y-direction, where α is the angle of incidence. Laser wavelength and frequency are Doppler-shifted in M, $\lambda = \lambda_0/\cos\alpha$, $\omega = \omega_0 \cos\alpha$, where index 0 denotes the laboratory frame values. Lorentz contraction occurs only in y-direction, *not* in x- or z-direction. In M, we impose transverse

will be adjusted to be equal to the front pulse's angle. The remaining pulse parameters are independent.
The **polarization** can be chosen as 's' (electric field perpendicular to the plane of incidence, **polarization**=1), 'p' (electric field parallel to the plane of incidence, **polarization**=2), and 'circular' (**polarization**=3). The pulse **duration** is expected in laser cycles. The temporal **shape** of the *pulse amplitude's* envelope is characterized by linear (**shape**=1), sinusoidal (**shape**=2), or $\sin^2$-shaped (**shape**=3) edges. Their duration **raise** is again given in laser cycles, respectively, so that the remaining pulse time with constant maximum amplitude is simply **duration**$-2\times$**raise**, as shown in Fig. 4.
Finally, one can decide to save the temporal profile of the pulse by setting the switch **pulse_save**= 1. Then the time step (in laser cycles) for saving is specified by **pulse_save_step** (see section 3.2, output file `pulse#*`).

3.1.2 Simulation time (&propagate)

The simulation time in laser cycles has to be entered following the key word &propagate. The simulation usually starts at time **prop_start**=0 and stops at time **prop_stop**, which may be larger or smaller than the pulse duration.

translational symmetry in y and z direction, and all quantities (densities, fields, ...) depend on only one spatial coordinate, x. In the laboratory frame L this symmetry means that all quantities have the same phase velocity component parallel to the surface in y-direction. For details see appendix.

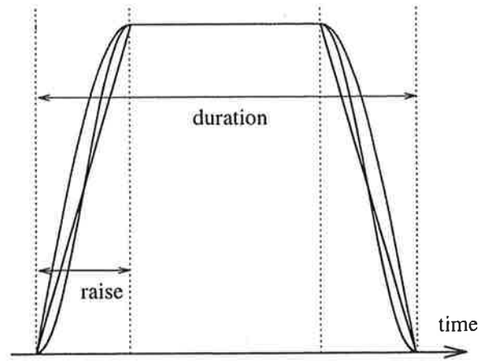

Fig. 4: *Pulse shapes*

3.1.3 Simulation box, plasma shape and density (&box)

The parameters following the key word &box are used to design the simulation box, target size and shape, and plasma density, as shown in Fig. 5. The simulation box is discretized in cells of equal length Δx with numbers 1, 2, ..., **cells**, where **cells** is the total number of cells in the simulation box. Parameter **cells_per_wl** specifies the number of grid cells in x-direction per laser wavelength λ_0 in the laboratory frame, **cells_left** is the number of initially empty cells left or in front of the plasma, **cells_plasma** the total number of grid cells initially occupied by plasma, and **cells_ramp** the number of cells in a linear ramp region at the front side of the plasma.
The maximum ion density in the plasma profile is specified by the parameter **n_ion_over_nc** which is the number density of ions in units of the critical density

$$n_c = \omega^2 \cdot \frac{m_e \varepsilon_0}{e^2}, \tag{3}$$

see appendix B. Electron density is given by **n_ion_over_nc** $\times\ z$, where z is the ion's charge state (see below). One can decide to save the plasma profile setting the switch **box_save**= 1 (see section 3.2, output file `domain-*`).

3.1.4 Particle species (&electrons, &ions)

Electrons are characterized by three parameters. The first technical parameter **fix** will inhibit moving the electrons in x-direction if set equal to one. This is of course unphysical, but can be useful for tests.
ppc is the number of *macro* particles per cell at maximum initial plasma density, see Fig. 5. Macro particles are groups of electrons containing an extensive number of real electrons. The number of electrons represented by one macro particle is constant

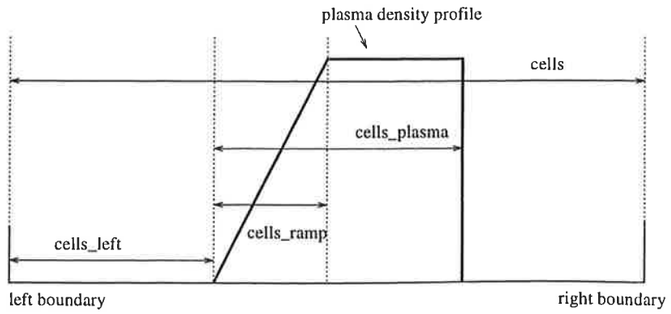

Fig. 5: *Simulation box and plasma density profile*

throughout the simulation box, and density is varied (e.g. linear ramp) by varying the number of macro particles per cell. Increasing **ppc** leads to higher accuracy and lower noise levels, but also to longer simulation times.
vtherm specifies the inital electron thermal velocity in units of the velocity of light. For the ions one has to specify two additional parameters, the charge state **z** of the ions and their mass **m** in units of the electron mass.
Concerning the initial thermal electron velocity, one has to be careful for numerical reasons. In order to prevent *numerical heating* [15], one should choose **vtherm** such that the Debye length is on the order of the spatial grid size Δx. This imposes the condition

$$\frac{v_{\text{therm}}}{c} \approx \sqrt{\frac{n_e}{n_c}}\,\frac{2\pi}{\textbf{cells\_per\_wl}}. \tag{4}$$

A table that relates the thermal velocity to energy in electron Volt is given on the bottom of any example input file in directory `pic/lpic/input/`. Notice that for temperature $T = 0$ the heating rate can be so small that the heating may be negligible during the simulation time of several tens of laser cycles.

3.1.5 Diagnostics (&output, &energy, &flux, ...)

The following sets of parameters specify the output **path** for `LPIC++` data and all implemented diagnostics. Notice that it is impossible to store all accessable data during a simulation run, since this would need more disk space than ususally available. One has to focus on parts of information. Therefore five major groups of diagnostics have been introduced:
First, diagnostics related to energy balances, absorption and reflectivity (&energy, &flux, &reflex). They can be switched on and off seperately (**Q**). Start and stop time (**t_start** and **t_stop**) for the diagnostic and the time step **t_step** between write operations to disk have to be specified in laser cycles. If **t_step**=0 is chosen, data will be written in every time step (see section 3.2, output files `energy-*`, ...). &energy is used to write longitudinal and transverse field energies, total field energy, the total kinetic energy of all particles, and the total energy in the domain, minus initial energies, respectively. &flux writes field energy fluxes at domain boundaries, and &reflex leads to reflectivity calculations at the boundaries. For the reflectivity, floating averages are taken over one laser cycle, respectively.
Second, spatial &snapshots *at fixed times* can be taken concerning grid quantities like electron and ion densities, current and field components. The switch and time para-

meters have the same meaning as above (see section 3.2, output files `snap-*`).
Third, phasespace (x, v) snapshots and velocity distributions for electrons and ions, respectively (&el_phasespace, &ion_phasespace, &el_velocity, &ion_velocity), can be produced *at fixed times*. Phasespace plots take into account all particles of a species. The corresponding velocity distribution sums the phasespace information over x, but can be taken independently (see section 3.2, output files `phase*`, `velocity-*`, and section 3.3).
Fourth, &spacetime information can be stored. These are snapshots of grid quantities taken at every time step between specified times **t_start** and **t_stop** (in laser cycles). The end points **x_start** and **x_stop** of the spatial region have to be given in cells. These diagnostics give a detailed overview over the evolution of plasma and fields in space and time, but can be quite expensive in disk space. One has to care about the spacetime window and especially number of spacetime diagnostics one selects (see section 3.2, output files `spacetime-*`, and section 3.3).
Finally, grid quantities can be stored at *selected positions in space for all time steps* between **t_start** and **t_stop** using the diagnostic &traces. Here one has to specify the number of **traces** and their x-positions **t0, t1, ...** (in cells) in addition to switch **Q** and time window. (see section 3.2, output files `trace-*`, and section 3.3).
The diagnostics are far from complete, and the user may change the code and add further diagnostics (see section 6).

3.1.6 Restarting LPIC++ (&restart)

The &restart parameters are related to the case of a system crash. Setting the switch **Q_save**=1 here leads to writing intermediate restart files periodically from which the simulation could be restarted. The save file name is specified using **file_save**. After a crash occured, you set the switch **Q**=1, set the read file name **file** equal to the name of the save file **file_save** and start the simulation again, as usual. Since LPIC++ saves once per laser cycle, you will have lost cpu time corresponding to less than one laser cycle. Except for the above mentioned parameter **Q**, the input file should remain unchanged when restarting since the input data is read and used again. However, modifying the simulation time **prop_stop** or even the laser pulses could make sense in certain cases. Restarting LPIC++ also works in the parallel version.

3.1.7 Parallel LPIC++ version (&parallel)

If only the plain LPIC++ version is used, set **N_domains**=1 and skip this subsection.

The simulation box can be split in several spatial domains, specified by parameter **N_domains**. Starting `lpic_parallel` under PVM will then generate **N_domains** PVM-tasks, each of them processing a seperate part of the simulation box. Communication between tasks involves the exchange of particles, fields and currents, as usual. Since the particles cross boundaries of cells as well as *domains*, the particle load of the tasks will generally vary in time (simulation time is spent mainly for moving particles). To prevent performance losses, it is therefore important to reorganize the box splitting from time to time setting the switch **Q_reo**=1. This may lead to exchange of groups of cells between adjacent domains (tasks), i.e. the boundaries are allowed to move, see the example in section 4.5. Immediately after reorganizing, all domains will contain approximately same numbers of particles. The time **delta_reo** between reorganizations has to be given in integer multiples of a laser cycle. Indeed, reorganization is already used to achieve equal particle loads at time $t = 0$. Therefore it is recommended to have **Q_reo**=1 by default.
Of course, the parallel version works also with parameter **N_domains**=1. Then it does not need the PVM-daemons running.

3.2 Output

The default output path is `pic/data`. All tasks write to this directory and produce similar files whose name endings `*-1*`, `*-2*`, ... correspond to the domain number, in some cases followed by time in laser periods. The number of files can be quite large, especially in case of parallel processing. Some of these files are in ASCII-format and can be checked or plotted immediately:

1. `input.lpi`: A copy of the input file you selected.
2. `output.lpi`: During the initializing procedures all LPIC++ parts (classes) echo their input (obtained from the input file and from the command line) to this file. This file is written only by the first domain.
3. `error-*`: Comments and error messages.
   These files contain output of LPIC++'s error handler. Here you will find useful debugging information. Whenever LPIC++ crashes, check these files first to find out where the error occured. This is very unlikely unless you change the code!
4. `times-*`: Cpu time used for particle pusher, field solver and diagnostics, total cpu and system time.

5. pulse#1, pulse#2: Selected laser pulses (front pulse, rear pulse). These files are written in case of **pulse_save**=1. Columns:
   (a) time in laser cycles
   (b) dimensionless amplitude
6. domain-*: Initial plasma density profile in this domain. Columns:
   (a) cell number (including buffer cells, e.g. -1, 0, see Fig. 22)
   (b) x-coordinate of the cell
   (c) electron density, maximum normalized to 1
   (d) ion density, maximum normalized to 1
   (e) number of macro electrons in this cell
   (f) number of macro ions in this cell
7. energy-*: Energy balance in the moving frame M, for units see appendix. Columns:
   (a) time in laser cycles
   (b) time-integrated energy gain of this domain via electromagnetic field flux across domain boundaries
   (c) total energy in this domain (minus initial)
   (d) total field energy (minus initial)
   (e) transverse field energy (minus initial)
   (f) longitudinal field energy (minus initial)
   (g) total kinetic energy, all species (minus initial)
8. flux-*: Electromagnetic field flux $\propto \boldsymbol{E} \times \boldsymbol{B}$ across domain boundaries in frame M, for units see appendix. Columns:
   (a) time in laser cycles
   (b) incoming flux left
   (c) outgoing flux left
   (d) incoming flux right
   (e) outgoing flux right

9. reflex-*: Cycle-averaged reflected intensity at domain boundaries, divided by cycle-averaged incident intensity. Columns:
   (a) time in laser cycles
   (b) 'reflectivity' left boundary
   (c) 'reflectivity' right boundary
10. snap-*: Snapshots of grid quantities in frame M. The ending denotes the domain number and time in laser cycles (e.g. `snap-1-5.000`: domain 1, time 5 cycles). For units see appendix. Columns:
    (a) x-coordinate
    (b) E_x
    (c) E_y
    (d) E_z
    (e) B_y
    (f) B_z
    (g) electron density
    (h) ion density
    (i) j_x
    (j) j_y
    (k) j_z
    (l) number of macro electrons
    (m) number of macro ions
11. velocity-*: Velocity distributions in the laboratory frame, endings denote domain number, particle species (sp0=electrons, sp1=ions), time in laser cycles (e.g. `velocity-1-sp0-5.000`: domain 1, electrons, time 5 cycles). Columns:
    (a) v = velocity / (velocity of light)
    (b) number of particles with $v_x = v$ (within velocity range Δv)
    (c) number of particles with $v_y = v$ (within velocity range Δv)
    (d) number of particles with $v_z = v$ (within velocity range Δv)

(e) number of particles with $\sqrt{v_x^2 + v_y^2 + v_z^2} = v$ (within velocity range Δv)

12. reo-*: Box splitting, domain boundaries, particle numbers per domain. Ending denotes domain number. These files are only written in the parallel version with reorganization. Columns:

    (a) time

    (b) number of cell at left boundary

    (c) number of cell at right boundary

    (d) total number of particles in this domain

The remaining files are binary files and have to be postprocessed. These are `phase*`, `spacetime-*` and `trace-*`.

3.3 Postprocessing

Directory `pic/post/` contains the executable `postprocessor`, the shell script `lpic.post` and the input file `input.post`. The source code is contained in directory `src/`, the object directory is `obj/`. The input file is organized similar to the `LPIC++` input files, see Fig. 6 for an example. There are key words (&traces, &spacetime and &phasespace) followed by corresponding parameters. Again, only values should be changed, not key words or parameter names. After editing the input file the postprocessor is invoked using the shell script by typing

```
lpic.post
```

This script creates a subdirectory `Post/` in the data directory `pic/data/`, and the postprocessor's output will then be written to `pic/data/Post/`.
In the following the input parameters in `input.post` and resulting output are discussed.

3.3.1 &traces, Fourier transforms

Traces are grid quantities (fields, densitites, currents) written *at selected positions* (cells) but for each time step in a given time interval. The original binary files `trace-*` are postprocessed here. Specify the time window to process (**period_start** and **period_stop** in laser cycles). In the following list of switches (**ex**, **ey**, ...) one can select which quantities to analyze by setting their corresponding switch equal to one.
Meaning of switches (for units see appendix):


```
////////////////////////////////////////////////////////////////////////////////
//
//  input parameters for the lpic-postprocessor
//
////////////////////////////////////////////////////////////////////////////////

&traces
--------------------------------------------------------------------------------
period_start  = 1
period_stop   = 20
period_screen = 1

ex = 0,   ey = 0,   ez = 0,   by = 0,   bz = 0,
fp = 1,   fm = 1,   gp = 1,   gm = 1,
Pi = 0,   Pr = 0,   Sr = 0,   Si = 0,
de = 0,   di = 0,   jx = 0,   jy = 0,   jz = 0

&spacetime
--------------------------------------------------------------------------------
t_start      = 0
t_stop       = 9
x_start      = 0
x_stop       = 6

imagesize    = 400
smooth       = 1

Q_de    = 0,   C_de    = 6
Q_di    = 0,   C_di    = 0
Q_jx    = 0,   C_jx    = 0
Q_jy    = 0,   C_jy    = 0
Q_jz    = 0,   C_jz    = 0
Q_ex    = 0,   C_ex    = 0
Q_ey    = 0,   C_ey    = 0
Q_ez    = 0,   C_ez    = 0
Q_by    = 0,   C_by    = 0
Q_bz    = 0,   C_bz    = 0
Q_edens = 1,   C_edens = 0.0004

Q_kt    = 0,   C_kt    = 1e-3
Q_kw    = 0,   C_kw    = 1e-3
K_cut   = 10,  W_cut   = 10,

x_offset     = 0
contour_1    = 100
contour_2    = 200
contour_3    = 300

&phasespace
--------------------------------------------------------------------------------
period_start = 0.0
period_stop  = 20.0
period_step  = 2.0

Q_vx = 0
Q_vy = 0
Q_vz = 0

xmax         = 5
xoffset      = 1
================================================================================
```

Fig. 6: *The postprocessor input file* `pic/post/input/input.fresnel.0`.

- ex, ey, ez, by, bz: field components
- fp: right going part of p-polarized light (contributes to ey, bz), $F^+ := \frac{1}{2}(E_y + cB_z)$, see section 5, Eqs. (11-13)
- fm: left going part of p-polarized light (contributes to ey, bz), $F^- := \frac{1}{2}(E_y - cB_z)$
- gp: left going part of s-polarized light (contributes to ez, by), $G^+ := \frac{1}{2}(E_z + cB_y)$
- gm: right going part of s-polarized light (contributes to ez, by), $G^- := \frac{1}{2}(E_z - cB_y)$
- Pi: sum of power spectra of right going ey and bz

- Pr: sum of power spectra of left going ey and bz
- Si: sum of power spectra of right going ez and by
- Sr: sum of power spectra of left going ez and by
- de, di: electron and ion density
- jx, jy, jz: current components

in frame M, respectively.
For example, setting **ex**=1 will generate two ASCII files, `ft-ex-trace`, `ft-ex` in directory `pic/data/Post`,

1. `ft-ex-trace`: 'ex' as a function of time
   Column 1: time in laser cycles
   Column 2: 'ex' at first trace position (cell)
   Coulmn 3: 'ex' at second position, etc.

2. `ft-ex`: power spectrum of 'ex', square of the absolut value of the fourier transform with respect to time
   Column 1: frequency scales to laser frequency
   Column 2: power spectrum of 'ex' taken at first trace position (cell)
   Coulmn 3: power spectrum at second position, etc.

and similar for the remaining quantities. Parameter **periods_screen** is used for taking the power spectra. It denotes the number of laser periods at the beginning and the end of the signal, respectively, which are used to cut the signal smoothly off to zero. One cycle is sufficient here.

3.3.2 &spacetime

Here, the &spacetime files (`spacetime-*`in directory `pic/data/`) are postprocessed. First select the time window in laser cycles, (**period_start**, **period_stop**) and spatial window in wavelangths λ_0 (**x_start**, **x_stop**). The postprocessor will map this space-time-window to a matrix of size **imagesize** $\times$ **imagesize**, thereby averageing in x-direction over **smooth** cells. The switches **Q_de**, ... are used to select the quantity

(in frame M) to process, and the corresponding parameters **C_de**, ...specify *vertical cutoffs*, i.e. maximum values to be plotted (for units see appendix). In case of **Q_de**=1, for example, electron densities from zero to **C_de** (for units see appendix) are then mapped linearly to the range 0 to 255, respectively, and written to the binary file `pic/data/Post/spacetime-de`. The file structure is as follows, see Fig. 7: The first **imagesize** bytes contain the data corresponding to t = **period_start** and

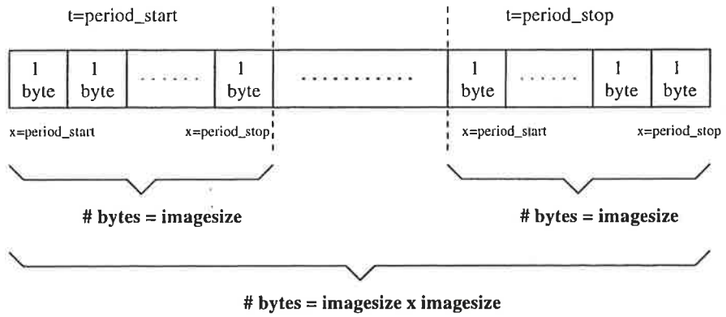

Fig. 7: *Data structure of* `pic/data/Post/spacetime-de`.

x_start $\leq x \leq$ **x_stop**, and so on, and the last **imagesize** bytes correspond to time t = **period_stop**. This file can then be plotted, for example using `IDL`.
In addition to plotting data versus coordinate and time, one can choose Fourier transformations of the same data window with respect to coordinate only (switch **Q_kt**, vertical cutoff **C_kt**), and with respect to both coordinate and time (switch **Q_kw**, vertical cutoff **C_kw**). The results are power spectra (absolut values of Fourier transforms squared), k-space versus time and k-space versus ω-space, respectively. In these cases, additional k-space and ω-space cutoffs (switches **K_cut**, **W_cut**) can be set. Here **K_cut**=1 means that the k-space is plotted from 0 to $+k_0$ and from $-k_0$ to $+k_0$, respectively, where k_0 is the laser wave number in laboratory frame. **W_cut**=1 means that ω-space is plotted from 0 to ω_0. For example, the data is written to files `pic/data/Post/spacetime-kt-de` and `pic/data/Post/spacetime-kw-de`, respectively. The files are organized as follows. **Q_kt**: The first **imagesize** bytes contain the data corresponding to t = **period_start** and $0 \leq k/k_0 \leq$ **K_cut**, and so on, and the last **imagesize** bytes correspond to time t = **period_stop**. **Q_kw**: The first **imagesize** bytes contain the data corresponding to $\omega = 0$ and $-$**K_cut** $\leq k/k_0 \leq +$**K_cut**, and so on, and the last **imagesize** bytes correspond to frequency ω/ω_0 = **W_cut**.
If you have access to `IDL`, then the following parameters **x_offset**, **contour_1**, **con-**

tour_2, **contour_3**can be used to define a plotting offset in x-direction and to make IDL draw contour lines.
IDL scripts are also generated by the postprocessor and written to directory `pic/-data/Post/`.

3.3.3 &phasespace

Here, the phasespace files are postprocessed, essentially the output of several domains is linked to files `phasex-*`, `phasey-*`, `phasez-*` in directory `pic/data/Post`, where the ending denotes the point of time in laser cycles. One has to select the time window (**period_start**, **period_stop**), the time step **period_step** in cycles between phasespace output (see corresponding files in `pic/data`), and switch the velocity components (in the laboratory frame) on or off (**Q_vx**, **Q_vy**, **Q_vz**). The resulting binary files contain phasespace images of size 400 × 400 pixels (bytes). The velocity range is $[-c, c]$, and the spatial range corresponds to the width of the simulation box. Each byte contains the number of macro particles in the corresponding phasespace interval $\Delta x \times \Delta v$. The file structure is as follows, see Fig. 8: The first 400 bytes contain the

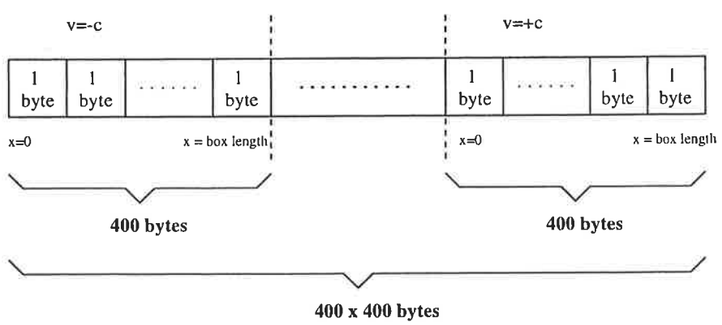

Fig. 8: *Data structure of* `phasex-*`, `phasey-*`, `phasez-*` *in directory* `pic/data/Post`.

data corresponding to velocity $v = -c$ and the whole box length, and so on, and the last 400 bytes correspond to $v = c$. This file can be plotted, for example using IDL.
The parameters **xmax** and **xoffset** are used for generating an IDL script that displays a phasespace movie.

3.4 Running LPIC++ under PVM

If PVM is available, start the PVM daemons in the background on your parallel (virtual) machine, e.g. using a hostfile `pvmhosts`,

```
pvmd pvmhosts &
```

Then the parallel LPIC++ version can be invoked with several domains, e.g.

```
lpic_parallel input/input.positron,
```

see section 4.5.

4 Examples

In directory `pic/lpic/input/` a collection of example input files is prepared. In the following we will briefly comment on these examples and show parts of the simulation results. In order to get used to LPIC++, one should run these examples and compare the results to those shown in the following.

4.1 Diagnostics

To start, run the input file `input.diagnostics`,

```
lpic_plain input/input.diagnostics &
```

This will only demonstrate all available types of output files in directory `pic/data/` in case of the plain version.
Try also `input.pulse` for two laser pulses propagating in opposite direction through an empty box.

4.2 Performance

The input file `input.performance` is designed to test the performance of the computer, where you are running LPIC++. Output is reduced to a minimum. The essential parameters are: 100 cells plasma, 100 particles per cell, 100 time steps per laser cycle and only one cycle of simulation time. This makes $100 \times 100 \times 100 = 10^6$ particle pushes in total. The output to screen shows the cpu time in seconds used for pushing particles, propagating fields, output diagnostics, and the total cpu time of this run. Particle pushes generally contribute the dominating part to the total simulation time. Here, one obtains the time for a *single* particle push by dividing the particle time by 10^6. Table 9 shows the time per particle push for several platforms we tried.

4.3 Reflection from Overdense Plasma

Next, consider reflection of a laser pulse from an overdense plasma slab with step-like shape, input file `input.reflection`. Some results are given in Fig. 10. Light is totally reflected from the plasma surface, but notice the *precurser* [20, 15] of E_z in Fig. 10 (a) running with *vacuum* velocity of light into the *overdense plasma*. At later times, the amplitudes of E_z and B_y show the typical exponential decay inside

Computer	Compiler + Option	Time per Particle Push [μs]
Pentium 100 MHz, Linux	g++ -O3	12.8
RS6000 P2SC@135 MHz	xlC -O3 -Q -qarch=pwr2 -qtune=pwr2 -qalign=power	2.1
RS6000 P2SC@120 MHz	xlC -O3 -Q -qarch=pwr2 -qtune=pwr2 -qalign=power	2.6
RS6000 Model 590	xlC -O3 -Q -qarch=pwr2 -qtune=pwr2 -qalign=power	4.3
RS6000, PowerPC 604e 43P-140 (AIX 4.1.5)	xlC -O3 -Q	4.8
RS6000 Model 250	xlC -O3 -Q	16.8
Cray T3E-600 LC512 DEC Alpha eV5 (21164)	CC -O3	3.8

Fig. 9: LPIC++ *performance on various platforms.*

the plasma (skin effect), modified here due to the fact that the plasma slab is *finite*. Fig. 10 (b) shows the built-up of the standing wave in front of the surface. Its amplitude is $eE_z/(m_e\omega c) = 0.02$. Fig. 10 (b) was created using IDL and the idl-script `pic/data/Post/idl-edens.pro`. To postprocess the LPIC++ results you can copy the template input file `pic/post/input/input.post.reflection` to file `pic/post/input.post`.

4.4 Reflection from Underdense Plasma

The following cases deal with reflection of low intensity light from *underdense* ($n_e/n_c = 0.64$) steplike density profiles. The LPIC++ results in Figs. 11 – 13 show the reflected intensity versus time. They are compared here with numerical solutions based on Fresnel's formulas [21, 10]. The laser pulses have rectangular shapes, so that their spectra contain broad frequency bands. Therefore the amplitude of the reflected intensity is not constant in time but varies and converges slowly towards the Fresnel-reflectivity. The numerical solutions are obtained using the program `pic/fresnel/fresnel`. The reader can find all necessary information for compiling and using in `pic/fresnel/-README`.
Now run the LPIC++ input file `input.fresnel.0` for normal laser incidence and com-

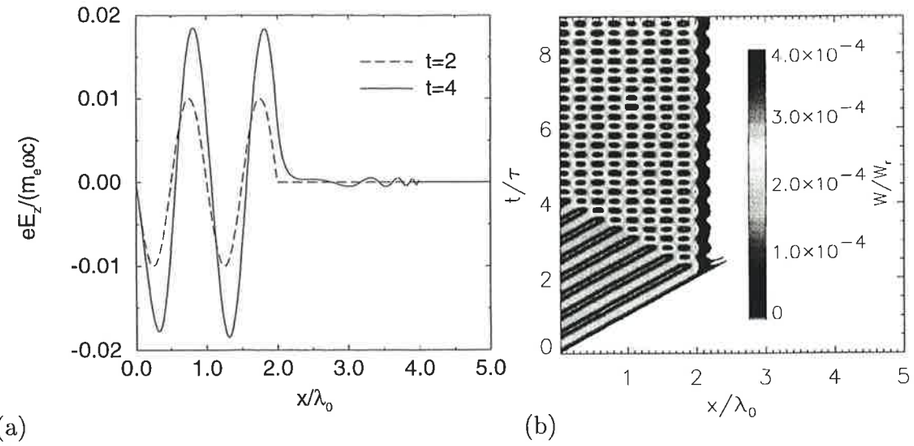

Fig. 10: *Reflection from overdense plasma. Parameters: Square pulse,* $a_0 = 0.01$, $\alpha = 0$, $n_e/n_c = 5$. *The plasma slab is located between* $x = 2$ *and* $x = 4$. *(a)* E_z *versus coordinate at times* $t = 2$ *and* $t = 4$. *(b) Field energy density versus coordinate* x *and time* t. *For units see appendix.*

pare to Fig. 11. Additional files for various angles and both polarizations (s, p) are also prepared (`input.fresnel.25s`, `input.fresnel.25p`, ...). The input files with endings `_fine` use especially fine meshes for accurate results.

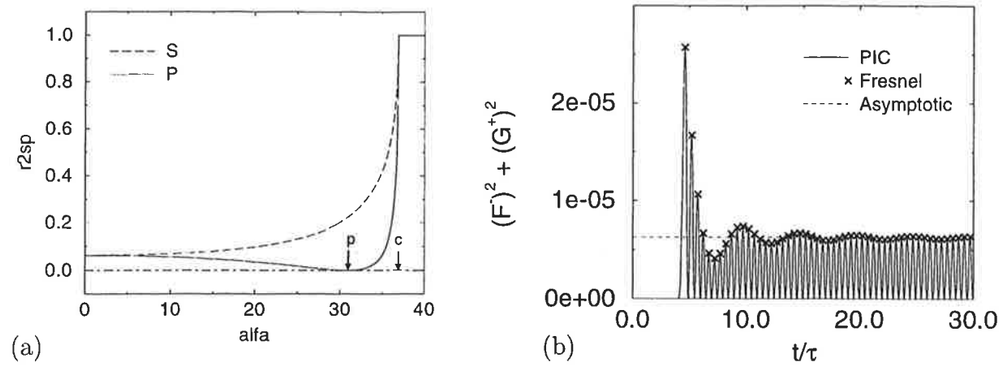

Fig. 11: *(a) Fresnel reflectivity for* $n_e/n_c = 0.64$. *The Brewster angle is* $\alpha_p = 31.0°$ *in this case, the angle of total reflection is* $\alpha_c = 36.9°$. *(b) A linearly polarized low intensity* ($a_0^2 = 10^{-4}$) *laser pulse with rectangular shape is incident under* $\alpha = 0$. *Reflected intensity* ($(F^-)^2+(G^+)^2$) *is plotted versus time.* `LPIC++` *results are plotted by means of lines, the crosses denote the numerical solution based on Fresnel's formulas. The dashed line is the reflected intensity* $R^2 \cdot a_0^2$ *for a infinite plane wave, where* $R = (1-N)/(1+N)$ *for* $\alpha = 0$ *and* $N^2 = 1 - n_e/n_c$.

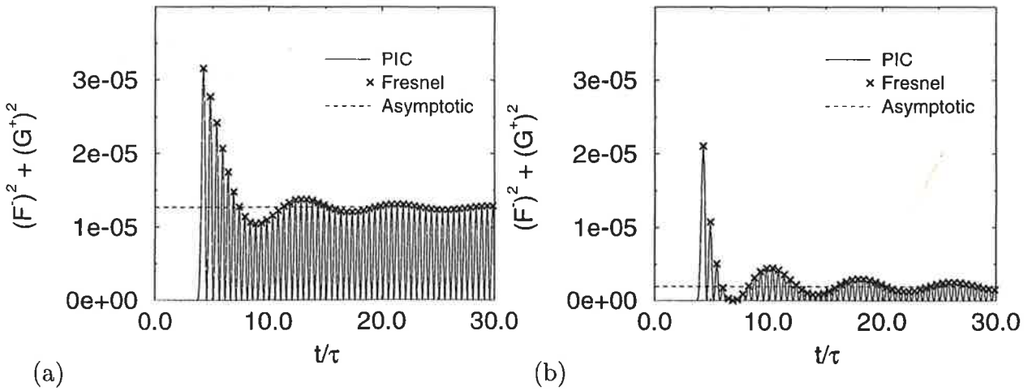

Fig. 12: *(a)* $\alpha = 24.6°$, *s-polarization. (b)* $\alpha = 24.6°$, *p-polarization.*

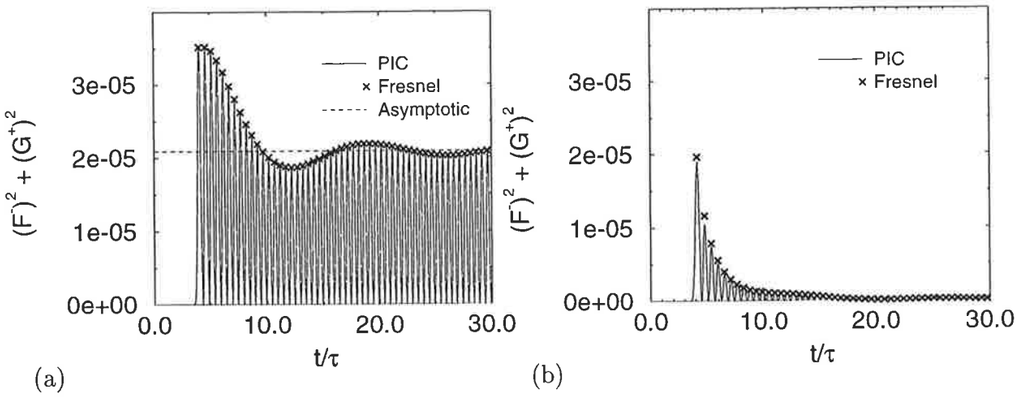

Fig. 13: *(a)* $\alpha = 30.45°$, *s-polarization. (b)* $\alpha = 30.45°$, *p-polarization.*

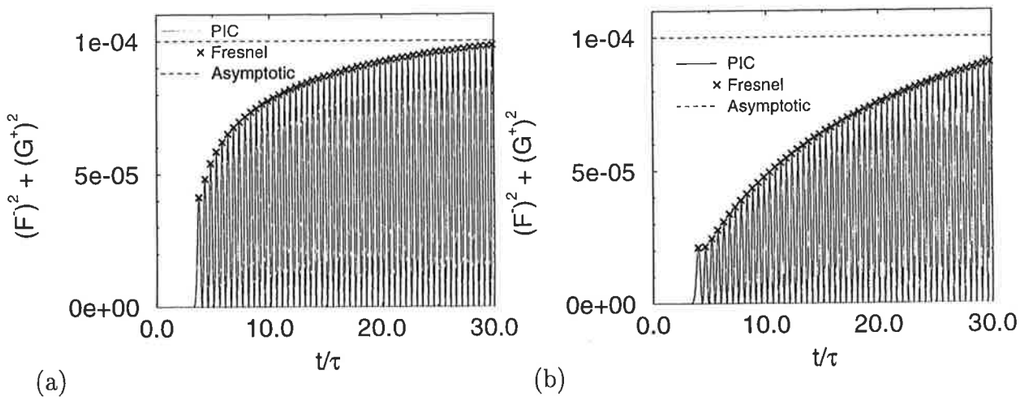

Fig. 14: *(a)* $\alpha = 37.47°$, *s-polarization. (b)* $\alpha = 37.47°$, *p-polarization.*

4.5 Parallel version and Restart

Input file `input.positron` can be used to test the parallel `LPIC++` version and the effect of killing and restarting the process(es),

```
lpic_parallel input/input.positron &
```

In this example, we use a strong laser pulse from the left and 'ions' with *electron* mass, so that the whole plasma is pushed by the laser to the right within a short time period (20 cycles). The simulation box is split into three domains, and the box is reorganized once per period, starting at $t = 0$. Fig. 15 (a) and (b) show the electron and 'ion' density, respectively, versus coordinate and time. This result is also obtained with killing and restarting `LPIC++` at some intermediate stage of the simulation, and also when using the plain `LPIC++` version. Fig. 16 (a) shows the box decomposition versus time, where the lines are the cell numbers at domain boundaries. Fig. 16 (b) shows the total number of particles in each domain.

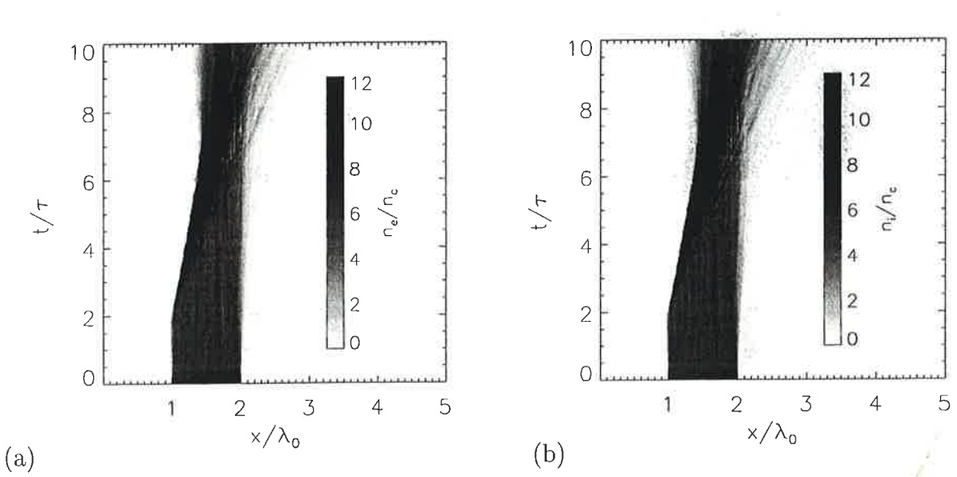

Fig. 15: *A laser pulse with amplitude* $a_0 = 0.5$ *is normally incident on a plasma slab with* $n_e/n_c = 6$ *and equal electron and 'ion' masses and charges. (a) Electron density versus coordinate and time. (b) 'Ion' density versus coordinate and time.*

4.6 More Examples

Moreover, directory `pic/lpic/input` contains prepared input files for simulating the generation of laser harmonics by interaction of an ultrashort laser pulse with a step

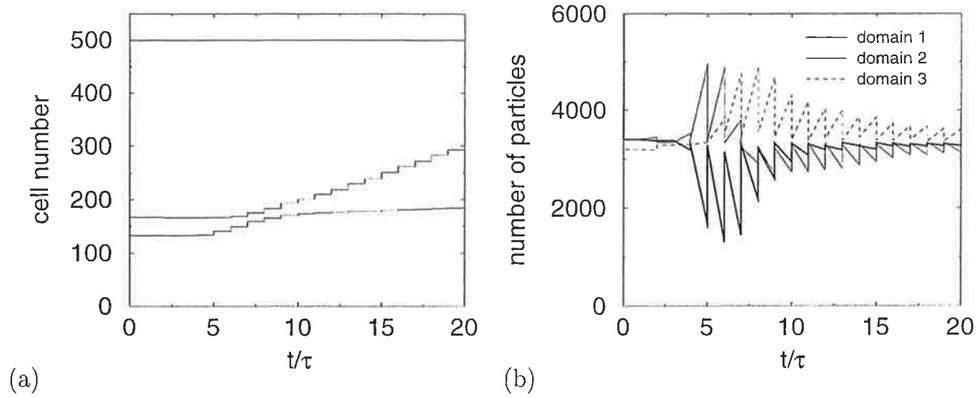

Fig. 16: *Reorganization: (a) domain interfaces, (b) number of particles.*

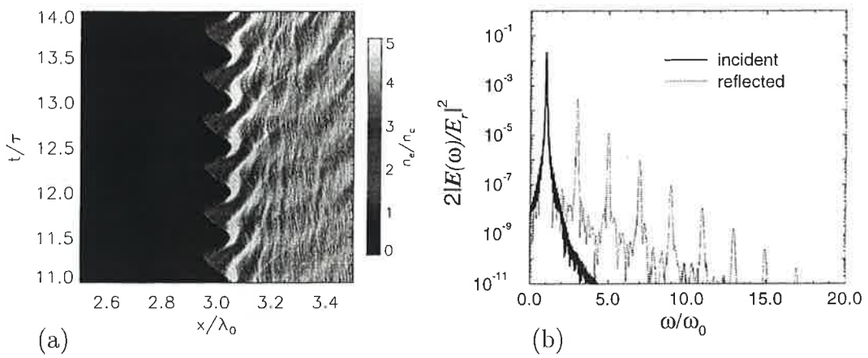

Fig. 17: *Generation of laser harmonics. Parameters: dimensinal amplitude* $a_0 = 0.5$*, electron density in units of the critical density* $n_e/n_c = 4$*. The plasma slab is located between* $x/\lambda_0 = 3$ *and* $x/\lambda_0 = 4$*. (a)* n_e/n_c *versus coordinate and time. (b) Power spectrum of incident and reflected light.*

boundary of a plane overdense plasma layer (`input.harmonics.*`), see Fig. 17, and input files `input.2omegap.*` are for simulating the generation of radiation at $2\omega_p$ from inverse two-plasmon decay in overcritical plasma [10].

5 Algorithm

LPIC++ solves Maxwell equations for the fields and the equations of motion for macro particles simultaneously.
The relativistic equations of motion for a collisionless plasma

$$\begin{aligned} \dot{\boldsymbol{p}} &= q_s(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}), & \boldsymbol{p} &= m_s \gamma \boldsymbol{v}, \\ \dot{\boldsymbol{r}} &= \boldsymbol{v}, & \gamma &= \sqrt{1 + (\boldsymbol{p}/m_s\, c)^2} \end{aligned} \tag{5}$$

are solved for each particle once per time step Δt. The macro particles contribute to charge and current densities ρ and $\boldsymbol{j}$ on a spatial grid with spacing Δx. Maxwell equations

$$\begin{aligned} &\nabla \times \boldsymbol{E} = -\partial_t \boldsymbol{B}, \quad \nabla \times \boldsymbol{B} = \frac{1}{\varepsilon_0\, c^2}\, \boldsymbol{j} + \frac{1}{c^2} \partial_t \boldsymbol{E}, \\ &\nabla \cdot \boldsymbol{B} = 0, \qquad\quad \nabla \cdot \boldsymbol{E} = \frac{1}{\varepsilon_0}\, \rho \end{aligned} \tag{6}$$

are then solved on this grid. This procedure is iterated leading to the selfconsistent evolution of plasma and fields. The PIC-cycle is shown in Fig. 18, and our grid structure is shown in Fig. 19.

5.1 Maxwell Equations

In one spatial dimension, the integration of Maxwell equations is divided into two parts, integration of transverse fields $E_{y,z}$ and $B_{y,z}$, and integration of the longitudinal field E_x.

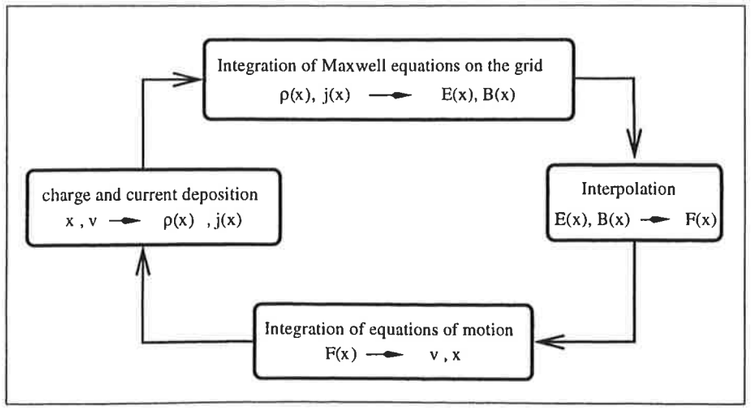

Fig. 18: *PIC code cycle.*

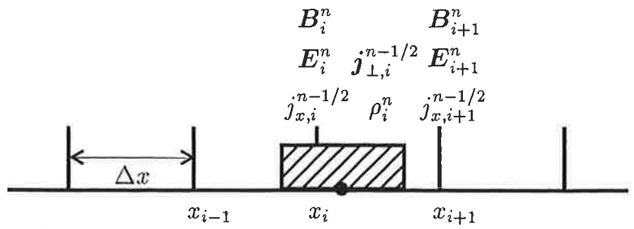

Fig. 19: *Grid structure: The macro particle belongs to cell i if its center x lies in $x_i \leq x < x_{i+1}$. Charge and transverse currents are located in cell centers, electromagnetic fields and longitudinal currents are located at cell boundaries.*

5.1.1 Transverse Fields

In one spatial dimension, Maxwell equations for the transverse fields can be rewritten in the following way [15]:

$$(\partial_t - c\,\partial_x)\,(E_y - cB_z) = -\frac{1}{\varepsilon_0} j_y, \tag{7}$$

$$(\partial_t + c\,\partial_x)\,(E_y + cB_z) = -\frac{1}{\varepsilon_0} j_y, \tag{8}$$

$$(\partial_t - c\,\partial_x)\,(E_z + cB_y) = -\frac{1}{\varepsilon_0} j_z, \tag{9}$$

$$(\partial_t + c\,\partial_x)\,(E_z - cB_y) = -\frac{1}{\varepsilon_0} j_z, \tag{10}$$

which can be combined to

$$(\partial_t \pm c\,\partial_x)\,F^{\pm} = -\frac{1}{2\varepsilon_0} j_y, \qquad F^{\pm} := \frac{1}{2}\,(E_y \pm cB_z)\,, \tag{11}$$

$$(\partial_t \mp c\,\partial_x)\,G^{\pm} = -\frac{1}{2\varepsilon_0} j_z, \qquad G^{\pm} := \frac{1}{2}\,(E_z \pm cB_y)\,. \tag{12}$$

The fields are obtained from $F^{\pm}$ and $G^{\pm}$ by

$$\begin{aligned} E_y &= F^+ + F^-, \quad cB_z = F^+ - F^-, \\ E_z &= G^+ + G^-, \quad cB_y = G^+ - G^-. \end{aligned} \tag{13}$$

The quantities $F^{\pm}$ and $G^{\pm}$ are integrated along *vacuum characteristics*. F^+ is the right going contribution to E_y and B_z, F^- the left going contribution. Similarly, G^+ is the *left* going contribution to E_z and B_y, and G^- the *right* going part.
The discretized version for F^+ in Eq. (11) reads

$$\frac{F^+(x+\Delta x, t+\Delta t) - F^+(x,t)}{\Delta t} = -\frac{1}{2\varepsilon_0}\, j_y\left(x + \frac{\Delta x}{2}, t + \frac{\Delta t}{2}\right),$$

which is centered in space and time. The grid step Δx is connected to the time step Δt by

$$\Delta x = c\Delta t,$$

so that the vacuum dispersion is exactly fulfilled. In summary we have

$$F_{i+1}^+(n+1) = F_i^+(n) - \frac{\Delta t}{2\varepsilon_0}\, j_{y,i+1/2}^{n+1/2}, \tag{14}$$

$$F_i^-(n+1) = F_{i+1}^-(n) - \frac{\Delta t}{2\varepsilon_0}\, j_{y,i+1/2}^{n+1/2}, \tag{15}$$

$$G_i^+(n+1) = G_{i+1}^+(n) - \frac{\Delta t}{2\varepsilon_0}\, j_{z,i+1/2}^{n+1/2}, \tag{16}$$

$$G_{i+1}^-(n+1) = G_i^-(n) - \frac{\Delta t}{2\varepsilon_0}\, j_{z,i+1/2}^{n+1/2}. \tag{17}$$

Here, index i stands for the cell coordinate $x_i = i\Delta x$ (see Fig. 19) and n for the time $t_n = n\Delta t$. Light waves are coupled into the simulation box simply by imposing boundary conditions on these quantities $F^{\pm}$ and $G^{\pm}$.
For units used in `LPIC++`, see appendix.

5.1.2 Longitudinal Field

According to Villasenor and Buneman [16], we use only currents to integrate the longitudinal equation of motion. In one spatial dimension, one finds from Ampère's law $(\nabla \times \boldsymbol{B})_x = \partial_y B_z - \partial_z B_y = 0 = j_x/\varepsilon_0 c^2 + \partial_t E_x/c^2$ the following *local* procedure,

$$\partial_t E_x(x,t) = -\frac{1}{\varepsilon_0} j_x(x,t). \tag{18}$$

Its discretized time and space centered version reads

$$E_i^{n+1} = E_i^n - \frac{\Delta t}{\varepsilon_0}\, j_{x,i}^{n+1/2}, \tag{19}$$

which is equivalent to solving Poisson's equation using the *charge density*, which is a *global* procedure in space.
For units used in `LPIC++`, see appendix.

5.2 Equation of Motion

A time-centered 'Leap-Frog' scheme according to Birdsall and Langdon [15] is used for integrating the equation of motion, i.e. particle position $x(t)$ and velocity $\boldsymbol{v}(t + \Delta t/2)$ are staggered in time.

Since arbitrary polarization can be chosen for the laser light, three velocity components are taken into account. The relativistic equation of motion

$$m\partial_t \boldsymbol{u} = q\left(\boldsymbol{E} + \frac{1}{\gamma}\boldsymbol{u}\times\boldsymbol{B}\right), \qquad \boldsymbol{u} = \gamma\boldsymbol{v}$$

is integrated using the method of *Boris* [19] cited in Birdsall and Langdon [15]. Fields given at time $t = n\Delta t$ are interpolated linearly to the particle position from neighbouring grid points. The discretized and time centered version of the equation of motion then reads

$$\frac{\boldsymbol{u}^{n+1/2} - \boldsymbol{u}^{n-1/2}}{\Delta t} = \frac{q}{m}\left(\boldsymbol{E}^n + \frac{1}{2\gamma^n}\left(\boldsymbol{u}^{n+1/2} + \boldsymbol{u}^{n-1/2}\right)\times\boldsymbol{B}^n\right). \tag{20}$$

Substituting

$$\boldsymbol{u}^{n-1/2} = \boldsymbol{u}^- - \frac{q}{2m}\boldsymbol{E}^n\Delta t, \qquad \boldsymbol{u}^{n+1/2} = \boldsymbol{u}^+ + \frac{q}{2m}\boldsymbol{E}^n\Delta t \tag{21}$$

leads to

$$\frac{\boldsymbol{u}^+ - \boldsymbol{u}^-}{\Delta t} = \frac{q}{2m\gamma^n}\left(\boldsymbol{u}^+ + \boldsymbol{u}^-\right)\times\boldsymbol{B}^n, \tag{22}$$

which describes a rotation of vector $\boldsymbol{u}^-$ to $\boldsymbol{u}^+$, since multiplying Eq. (22) by $(\boldsymbol{u}^+ + \boldsymbol{u}^-)$ one obtains $(\boldsymbol{u}^+)^2 - (\boldsymbol{u}^-)^2 = 0$ and $\gamma^n = \sqrt{1 + (\boldsymbol{u}^+/c)^2} = \sqrt{1 + (\boldsymbol{u}^-/c)^2}$.
This corresponds to a three-step acceleration, first, 'half acceleration', second, 'rotation', third, 'half acceleration'.
We combine Birdsall's and Langdon's [15], two-step rotation

$$\begin{aligned}\boldsymbol{u}' &= \boldsymbol{u}^- + \boldsymbol{u}^-\times\boldsymbol{t},\\ \boldsymbol{u}^+ &= \boldsymbol{u}^- + \boldsymbol{u}'\times\boldsymbol{s},\end{aligned}$$

where $\boldsymbol{t} = \boldsymbol{B}\,q\Delta t/(2m\gamma^n)$ and $\boldsymbol{s} = 2\boldsymbol{t}/(1+\boldsymbol{t}^2)$, into

$$\boldsymbol{u}^+ = \begin{pmatrix} 1 - s_z t_z - s_y t_y & s_z & -s_y \\ -s_z & 1 - s_z t_z & s_z t_y \\ s_y & s_y t_z & 1 - s_y t_y \end{pmatrix} \boldsymbol{u}^- \tag{23}$$

with an error in angle between $\boldsymbol{u}^+$ and $\boldsymbol{u}^-$ on the order of $\mathcal{O}(\boldsymbol{t}^3)$.
Velocity is finally integrated to give the new particle position

$$\boldsymbol{r}^{n+1} = \boldsymbol{r}^n + \frac{\boldsymbol{u}^{n+1/2}}{\gamma^{n+1/2}}\Delta t, \qquad (\gamma^{n+1/2})^2 = 1 + (\boldsymbol{u}^{n+1/2}/c)^2. \tag{24}$$

For units used in LPIC++, see appendix.

5.3 Charge and Current Deposition

The following scheme is taken from [16] and has been reduced to one spatial dimension. We consider a particle with position x^n at $t^n = n\Delta t$ in $x_i \le x^n < x_{i+1/2}$, as shown in Fig. 19. Its charge contribution to cells i and $i-1$ at time t^n is

$$\rho_i^n = Z\,n_e/n_c\left(\frac{1}{2} + \frac{x^n - x_i}{\Delta x}\right), \qquad \rho_{i-1}^n = Z\,n_e/n_c\left(\frac{1}{2} - \frac{x^n - x_i}{\Delta x}\right), \tag{25}$$

dimensionless units are used, see appendix. A particle can only be shifted by less than than $\Delta x = c\,\Delta t$. Three cases have to be distinguished concerning the particle's final position (x^{n+1}) which lead to different charge and current depositions:

1. $x_i \le x^n < x_{i+1/2} \quad \wedge \quad x_{i-1/2} \le x^{n+1} < x_{i+1/2}$
   Charge contributions only to cells i und $i-1$ at time t^{n+1}

   $$\rho_i^{n+1} = Z\,n_e/n_c\left(\frac{1}{2} + \frac{x^{n+1} - x_i}{\Delta x}\right), \qquad \rho_{i-1}^{n+1} = Z\,n_e/n_c\left(\frac{1}{2} - \frac{x^{n+1} - x_i}{\Delta x}\right).$$

   From the equation of motion

   $$\rho_i^{n+1} - \rho_i^n = -\left(j_{x,i+1}^{n+1/2} - j_{x,i}^{n+1/2}\right)$$

   one obtains immediately the longitudinal current contribution at the boundary between cell $i-1$ and i,

   $$j_{x,i}^{n+1/2} = \rho_i^{n+1} - \rho_i^n = Z\,n_e/n_c\,\frac{x^{n+1} - x^n}{\Delta x}.$$

   The time centered transverse current contributions in cell i and $i-1$ are

   $$j_{y,i}^{n+1/2} = \frac{\rho_i^{n+1} + \rho_i^n}{2}\,\frac{v_y^{n+1/2}}{c} = Z\,n_e/n_c\left(\frac{1}{2} + \frac{x^{n+1} - x_i}{2\,\Delta x} + \frac{x^n - x_i}{2\,\Delta x}\right)\frac{v_y^{n+1/2}}{c}$$

   and

   $$j_{y,i-1}^{n+1/2} = \frac{\rho_{i-1}^{n+1} + \rho_{i-1}^n}{2}\,\frac{v_y^{n+1/2}}{c} = Z\,n_e/n_c\left(\frac{1}{2} - \frac{x^{n+1} + x_i}{2\,\Delta x} - \frac{x^n - x_i}{2\,\Delta x}\right)\frac{v_y^{n+1/2}}{c},$$

   respectively, and similarly for j_z.

2. $x_i \le x^n < x_{i+1/2} \quad \wedge \quad x^{n+1} < x_{i-1/2}$
   Charge and transverse current contributions to three cells and longitudinal current contributions at two cell boundaries. The time step Δt has to be split in two parts, ϵ and $(1-\epsilon)$, where

   $$\epsilon = \frac{x^n - x_{i-1/2}}{x^n - x^{n+1}}.$$

During time $\epsilon\Delta t$ there is longitudinal current across the boundary between cells i and $i-1$, charge contributions to cell i and $i-1$. During time $(1-\epsilon)\Delta t$ cells $i-1$, $i-2$ and their interface are involved. The following contributions are weighted time averages (over Δt). Current density contributions at time t^{n+1} are

$$j_{x,i}^{n+1/2} = -Z\, n_e/n_c \left(\frac{1}{2} + \frac{x^n - x_i}{\Delta x}\right), \qquad j_{x,i-1}^{n+1/2} = -Z\, n_e/n_c \left(\frac{1}{2} - \frac{x^{n+1} - x_i}{\Delta x}\right),$$

$$j_{y,i}^{n+1/2} = \langle\rho_i\rangle \frac{v_y^{n+1/2}}{c}, \qquad j_{y,i-1}^{n+1/2} = \langle\rho_{i-1}\rangle \frac{v_y^{n+1/2}}{c}, \qquad j_{y,i-2}^{n+1/2} = \langle\rho_{i-2}\rangle \frac{v_y^{n+1/2}}{c},$$

where

$$\begin{aligned}
\langle\rho_i\rangle &= Z\frac{n}{n_c}\frac{\epsilon}{2}\left(\frac{1}{2} + \frac{x^n - x_i}{\Delta x}\right),\\
\langle\rho_{i-1}\rangle &= Z\frac{n}{n_c}\left\{\frac{\epsilon}{2}\left(\frac{3}{2} - \frac{x^n - x_i}{\Delta x}\right) + \frac{1-\epsilon}{2}\left(\frac{3}{2} + \frac{x^{n+1} - x_{i-1}}{\Delta x}\right)\right\},\\
\langle\rho_{i-2}\rangle &= Z\frac{n}{n_c}\frac{1-\epsilon}{2}\left(\frac{1}{2} - \frac{x^{n+1} - x_{i-1}}{\Delta x}\right).
\end{aligned}$$

Similar results are obtained for j_z.

3. $x_i \le x^n < x_{i+1/2} \quad \wedge \quad x^{n+1} \ge x_{i+1/2}$

Longitudinal current across two interfaces. The same procedure as above leads to

$$\epsilon = \frac{x^n - x_{i+1/2}}{x^n - x^{n+1}},$$

density contributions in cells i and $i+1$

$$\rho_i^{n+1} = Z\, n_e/n_c \left(\frac{1}{2} - \frac{x^{n+1} - x_{i+1}}{\Delta x}\right), \qquad \rho_{i+1}^{n+1} = Z\, n_e/n_c \left(\frac{1}{2} + \frac{x^{n+1} - x_{i+1}}{\Delta x}\right),$$

and current contributions

$$j_{x,i}^{n+1/2} = Z\, n_e/n_c \left(\frac{1}{2} - \frac{x^n - x_i}{\Delta x}\right), \qquad j_{x,i+1}^{n+1/2} = Z\, n_e/n_c \left(\frac{1}{2} + \frac{x^{n+1} - x_{i+1}}{\Delta x}\right),$$

$$j_{y,i-1}^{n+1/2} = \langle\rho_{i-1}\rangle \frac{v_y^{n+1/2}}{c}, \qquad j_{y,i}^{n+1/2} = \langle\rho_i\rangle \frac{v_y^{n+1/2}}{c}, \qquad j_{y,i+1}^{n+1/2} = \langle\rho_{i+1}\rangle \frac{v_y^{n+1/2}}{c},$$

where

$$\begin{aligned}
\langle\rho_{i-1}\rangle &= Z\frac{n}{n_c}\frac{\epsilon}{2}\left(\frac{1}{2} - \frac{x^n - x_i}{\Delta x}\right),\\
\langle\rho_i\rangle &= Z\frac{n}{n_c}\left\{\frac{\epsilon}{2}\left(\frac{3}{2} + \frac{x^n - x_i}{\Delta x}\right) + \frac{1-\epsilon}{2}\left(\frac{3}{2} - \frac{x^{n+1} - x_{i+1}}{\Delta x}\right)\right\},\\
\langle\rho_{i+1}\rangle &= Z\frac{n}{n_c}\frac{1-\epsilon}{2}\left(\frac{1}{2} + \frac{x^{n+1} - x_{i+1}}{\Delta x}\right).
\end{aligned}$$

Similar results are obtained for j_z.

Three analogous cases have to be distinguished for initial particle position in $x_{i+1/2} \le x^n < x_{i+1}$.

6 LPIC++ Code

6.1 Data Structure

A macro particle is represented by a C data structure defined in `/pic/lpic/src/-include/particle.h` and shown in Fig. 20.

```
#ifndef PARTICLE_H
#define PARTICLE_H

#include <cell.h>

struct particle {

  int             number;        // number of this particle
  int             species;       // particle species, 0=electron, 1=ion
  struct cell     *cell;         // pointer to the cell this particle belongs to
  struct particle *prev;         // pointer to the previous particle in this cell
  struct particle *next;         // pointer to the next particle in this cell

  int    fix;                    // fixed species? 0->no, 1->yes
  double z;                      // charge of the micro particle in units of e
  double m;                      // mass of the micro particle in units of m_e
  double zm;                     // specific charge, z/m
  double x, dx;                  // position and shift within one timestep
  double igamma;                 // inverse gamma factor
  double ux, uy, uz;             // gamma * velocity
  double n;                      // particle density in units of n_c
  double zn;                     // contribution of the particle to the charge density
                                 // in units of n_c ( = z * n )
};

#endif
```

Fig. 20: *C data structure* `particle`.

In addition to physical parameters `z` ... `zn`, you find the unique particle number `number`, its species number `species`, a switch `fix` for fixing the macro particle, and three pointers.

Particles that belong to one cell, are linked in a chained list of particle structures, where each particle points to the preceeding (`*prev`) and following (`*next`) particle in the list. If there is none, `*prev` and `*next` are `NULL`-pointers, respectively. Moreover, each particle points to the cell it belongs to (`*cell`).

The cell itself is represented by a C data structure defined in `/pic/lpic/src/include/cell.h` and shown in Fig. 21. It contains physical parameters like the position `x` of the left cell boundary, total `charge`, currents `jx, jy, jz`, fields `ex, ..., gm` and particle densities `dens[2]`. For book keeping, a unique cell `number` and the number of macro particles within this cell (`np[2], npart`) have been added. From each cell the list of these particles can be accessed using the pointers `first, last, insert`. If there are no particles in a given cell, these pointers are `NULL`-pointers. The cells in turn are also linked in a chained list, see Fig. 22, so that they contain pointers to


```
#ifndef CELL_H
#define CELL_H

#include <particle.h>

struct cell {

  int    number;                 // number of this cell
  int    domain;                 // domain number it belongs to
  struct cell *prev;             // pointer to the previous (left) cell
  struct cell *next;             // pointer to the next (right) cell

  double x;                      // position of the left cell boundary in wavelengths
  double charge;                 // charge density in units e*n_c
  double jx, jy, jz;             // current density in units e*n_c*c
  double ex, ey, ez;             // electric fields in units m*omega*c/e
  double bx, by, bz;             // magnetic fields in units m*omega/e
  double fp, fm, gp, gm;         //
  double dens[2];                // densities for each species in units n_c

  int             np[2];         // # of electrons [0] and ions [1]
  int             npart;         // # particles

  struct particle *first;        // pointer to the first particle
  struct particle *last;         // pointer to the last particle
  struct particle *insert;       // pointer to particle, in front of which new particles
                                 // have to be inserted
};

#endif
```

Fig. 21: *C data structure* `cell`.

adjacent cell structures. Here, adjacent cell structures correspond to adjacent cells in the one-dimensional coordinate space.

In the parallel version of `LPIC++`, the whole grid (simulation box) is split into several domains each containing a fraction of the grid represented by chained lists of cells. The domain number `domain` then denotes the number of the domain a given cell belongs to. At domain boundaries, two buffer cells are added, respectively, which are necessary for exchanging particles, fields and currents between adjacent domains.

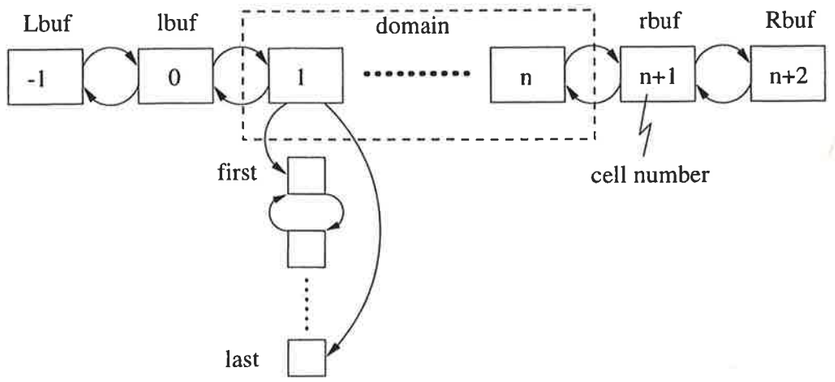

Fig. 22: *Chained lists of cells and particles.*

6.2 Classes, Dependencies and Files

All LPIC++ routines are member functions of C++ classes. These classes and their dependencies are described in the following, see Fig. 23.

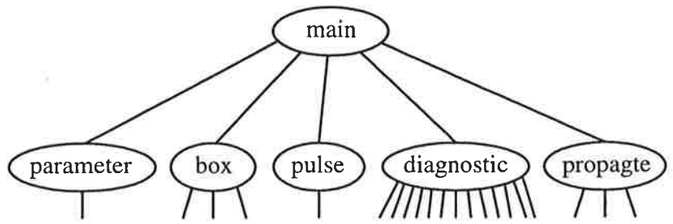

Fig. 23: *Classes and their dependencies.*

Parameter reads the command line parameters when calling LPIC++, i.e. domain number and input file name. Moreover, it reads laser pulse parameters (e.g. angle of incidence) from the input file and provides a few commonly used parameters. For details see `parameter.C` and `parameter.h`. For reading, **parameter** uses class **readfile**, see Fig. 24.

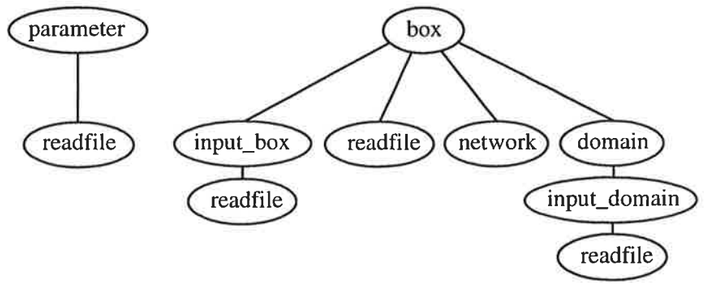

Fig. 24: *Classes* **parameter** *and* **box**.

Box organizes the communication between tasks and the periodic reorganization of domains in the parallel LPIC++ version. It contains the classes **domain** and **network**, see Fig. 24. **Network** contains all routines for starting tasks, exchanging particles, fields, currents and cells between adjacent domains. **Domain** mainly initializes the grid (chained list of cells) and particles in these cells for the given domain. Moreover, **box** contains an input class which is called here **input_box** and is responsible for reading necessary parameters from the input file. **Input_box** in turn uses **readfile**.

This type of reading input data is used in nearly all LPIC++ classes. The classes are contained in `*.C` and `*.h` files with corresponding file names.
Pulse contains all variables and functions necessary to specify a laser pulse. This class is used for both laser pulses in LPIC++, for the front and for the rear pulse, respectively.

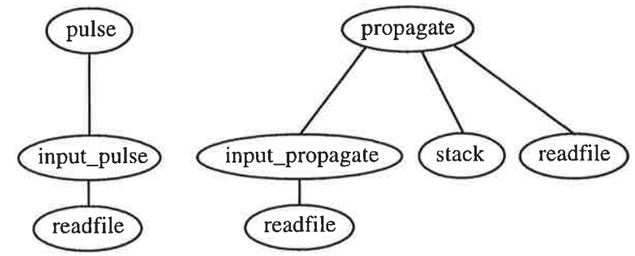

Fig. 25: *Classes* **pulse** *and* **propagate**.

Propagate contains the main loop of LPIC++, i.e. the particle push and field propagation. The corresponding function definitions and declarations are distributed in this case over four files, `propagate.C`, `propagate_fields.C`, `propagate_particles.C` and `propagate.h`. For initializing itself, it uses the classes **input_propagate** and **readfile**, see Fig. 25. Class **stack** is used for moving particles across cell boundaries, i.e. extracting particles from and inserting particles into chained particle lists of adjacent cells.

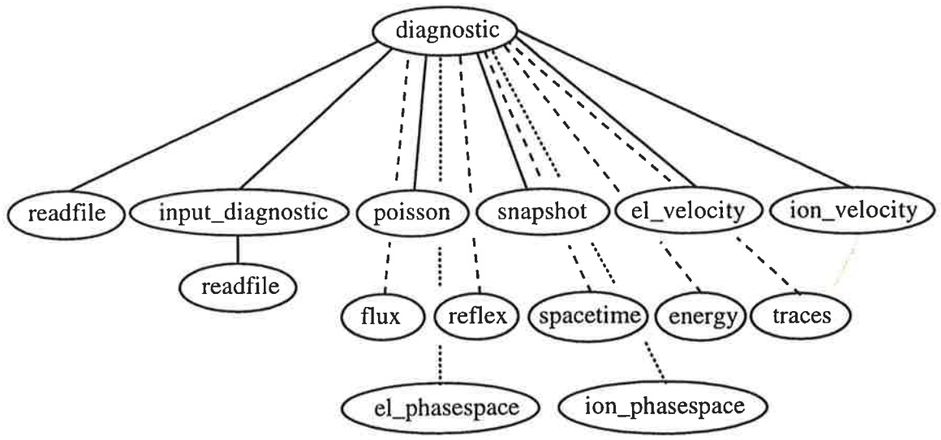

Fig. 26: *Class* **diagnostic**.

Diagnostic coordinates all diagnostic routines, which are distributed over the diagnostic classes, see Fig. 26. For further information see `diagnostic.C` and `diagnostic.h`.

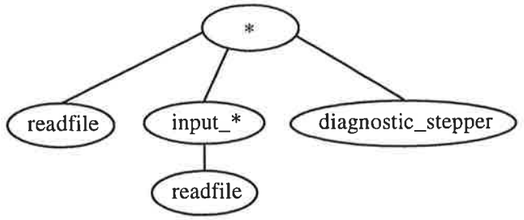

Fig. 27: *Classes used for all diagnostic classes except for* **readfile** *and* **input_diagnostic**.

All diagnostic classes except for **readfile** and **input_diagnostic** use **input_*** and **readfile** for initialization and **diagnostic_stepper** to store space and time related information, see Fig. 27. Moreover, **spacetime** uses **diagnostic_stepper** for each quantity, electron density, ..., energy density.

The names of the diagnostic classes are self-explaining, cf. section 3.1.5, and the corresponding routines can be found in `diagnostic_*.C` and `diagnostic_*.h`. **El_velocity** and **ion_velocity** are derived from class **velocity**. See `diagnostic_velocity.C` and `diagnostic_velocity.h` for more information. **El_phasespace** and **ion_phasespace** are both derived from class **phasespace**, for further details see `diagnostic_phasespace.C` and `diagnostic_phasespace.h`.

Poisson is not used so far. It solves Poisson's equation for the electrostatic potential and the longitudinal electric field. If the initial total charge distribution is not zero, **poisson** has to be used to initialize the longitudinal field. For details see `diagnostic_poisson.C` and `diagnostic_poisson.h`.

Class **error_handler** is contained in files `error.C` and `error.h`. It contains functions for writing error messages and comments to a specified error file, here `error-*` for each domain. Class **error_handler** is a member of nearly all `LPIC++` classes and is ususally initialized in their constructors.

Class **uhr** in files `uhr.C` and `uhr.h` is used for measuring cpu and system time. Variables of type **uhr** are defined and initialized in function `propagate::loop()` in order to measure cpu time for particle pushes, field propagation, diagnostics and total simulation time.

6.3 Program Flow

The rough program flow can be seen in function `main` which is shown in Fig. 28.

```
#include <main.h>

//////////////////////////////////////////////////////////////////////////////////////////

int main(int argc, char **argv)
{
    // initialize classes ////////////////////////////////////////////////////////////

    parameter p(argc,argv);                          // read parameters


    char                errname[filename_size];
    sprintf( errname, "%s/error-%d", p.path, p.domain_number );
    static error_handler bob("main",errname);        // error handler for main.C


    box         sim(p);                              // init domain, cells, particles
                                                     // spawn task for the following domain


    pulse       laser_front(p,"&pulse_front");       // init laser pulses
    pulse       laser_rear(p,"&pulse_rear");

    diagnostic diag(p,&(sim.grid));                  // init diagnostics
    propagate  prop(p,sim.grid);                     // init propagator

    // main loop /////////////////////////////////////////////////////////////////////

    prop.loop(p,sim,laser_front,laser_rear,diag);

    // exit //////////////////////////////////////////////////////////////////////////

    return main_exit(p,sim);                         // stop parallel task

}

//////////////////////////////////////////////////////////////////////////////////////////
//EOF
```

Fig. 28: *Function* `main`.

Variables of type **parameter**, **error_handler**, **box**, **pulse**, **diagnostic** and **propagate** are initialized sequentially, before the main `LPIC++` loop is entered. In the following, a detailed description of the program flow is given, for even more details check the `LPIC++` code.

class parameter:

- read parameters from command line and input file
- adjust angle such that number of steps per period in M is an integer
- save (adjusted) parameters to output.lpi

class error_handler:

- initialize class error_handler in main

class box:

- initialize **class network** (communication between tasks)
- initialize **class domain** (grid, cells, particles)
  - determines boundaries of domain from domain number
  - create chained list of cells, set cell numbers, domain pointers to left and right cells, set fields equal to zero, set normalized densities
  - allocate and link particles to cells, initialize particles
  - check particel numbers, positions and total charge in domain
  - write cell information to file `domain-*`: cell number, x-position, densities and # of particles in cell
- spawn task for the following domain
- initialize the counter for reorganization
- introduce global particle numbers (communicate with neighbour domains)
- get the total particle numbers in the simulation box (communicate with neighbour domains)
- adjust the size of the domain: reorganize for the first time in order to have a balanced load on all processors to start with

class pulse (for both pulses: laser_front and laser_rear):

- initialize classes **pulse**
- save pulse shape in file `pulse#*` (*=1 for front and *=2 for rear pulse), but only in the first domain

class diagnostic:

- initialize the following classes: **poisson**, **snapshot**, **el_velocity**, **ion_velocity**, **flux**, **reflex**, **spacetime**, **energy**, **trace**, **el_phasespace**, **ion_phasespace**: initialize specific diagnostic parameters
- use class **diagnostic_stepper** in each of them: initialize general diagnostic parameters
- initialize class **diagnostic**

class propagate:

- initialize class **stack**: create stack for particle transfer between cells
- initialize class **propagate**

main loop of LPIC++: prop.loop(p,sim,laser_front,laser_rear,diag)

- uses **parameter**, **box** (including **domain** and **network**), **pulse**s and **diagnostic**s
- initialize classes **uhr**
- start clock
- the main loop starts at time=start_time and continues until time=stop_time

clear_grid: set cell charge, dens[] and jx, jy, jz to zero for all cells in domain

particles:

- the following is performed for all particles in this domain:
  - deposit_charge: update of charge and dens[] in each cell of this domain, not necessary for the local algorithm (charge distribution of the preceeding half time step)
  - accelerate: acceleration according to Boris (in Birdsall, Langdon): interpolate fields to particle position, then half acceleration, rotation and second half acceleration
  - move: move the particle: calculate part→dx and part→x
  - has_to_change_cell: determine whether the particle has to leave its cell; in case it has to, put a marker for it on the stack
  - deposit_current: calculate the particle's current contributions; the currents are calculated from the continuity equation, assuming rectangular particle shape and area weighting (J.Villasenor and O.Buneman, Comp. Phys. Comm. 69 (1992) 306-316)
- do_change_cell: the markers for the particles which have to leave their cells are now removed from the stack and the corresponding particles are linked to their new cells; cells Lbuf and Rbuf remain empty, but particles may have moved to lbuf and rbuf
- mask_current: makes currents invisible near the box boundaries: this routine sets the currents smoothly equal to zero in a region of size 2×MASK at the left and right boundary of the simulation box; this inhibits artificial radiation at the boundaries

sim.talk.particles:
send/receive particles to/from neighbour domains:

- particles can move into the two buffer cells lbuf and rbuf within one time step, therefore particles are only sent from these two buffers and received into right and left
- domain particle numbers are updated

sim.talk.current:
send/receive current contributions and copies to/from neighbour domains:

- current contributions are sent from all four buffer cells (Lbuf, lbuf, rbuf and Rbuf) and received into right, right→prev, left and left→next, they are added to the actual currents in these cells
- for the field propagation a copy of jy and jz is needed in lbuf: a copy of currents jy and jz is needed only at the left boundary (in lbuf) in order to propagate the fields F^+ and G^- in cell "left". For the propagation of F^- and G^+ at the right boundary "right", the currents in cell "right" are sufficient. See MPQ-Report 219 p. 22 or propagate::fields in propagate.C

sim.talk.density:
send/receive density contributions to/from neighbour domains: cell charge and densities dens[] are sent from lbuf and rbuf and received into right and left, they are added to the actual values in these cells

fields:
propagate fields in this domain; the incident laser pulses enter as time dependent boundary conditions for the transverse fields at the boundaries in the first and last domain

sim.talk.field:
send/receive field copies to/from neighbour domains: field copies are sent from left and right and received into rbuf and lbuf

diag.out:
diagnostic output depending on various switches and counters

sim.reorganize:
depending on the reorganization counter:

- counts particles in domain: n_el, n_ion, n_part and checks whether they are still correct
- processor load is mainly determined by the number of particles handled, aim: almost the same number of particles on each processor
- writes to `error-*`: deviation of particle number from ideal particle number in percent
- reorganizes simulation box in order to have a balanced load on all processors
- reorganize in forward direction, from the left end of the box to the right end
- attention: do not use reorganize_f while information is stored in buffer cells (e.g. fields, charge, ...), do use it when the buffer cells are empty, e.g. in the main loop after a whole cycle.
- receive request from previous domain:
  - it has enough cells and nothing happens
  - it wants to get rid of some cells/particles and send them to this domain:
    1. get the number of cells and particles which will be received from prev
    2. allocate memory for cells and particles and link cells to domain, link all the particles to the first cell (left) and update n_left, n_cells, n_part
    3. receive and unpack the cells and particles from prev, and determine number of electrons/ions received and update n_e and n_ion

- it has not enough cells and requests a certain number of cells:
  1. determine number of cells and particles actually to be sent to previous domain
  2. inform previous domain of these numbers
  3. pack cells and particles linked to them (start with cell left) and send them to the previous domain
  4. delete memory which is still allocated by already sent cells and particles and update n_left, n_cells, n_el, n_ion, n_part, lbuf and Lbuf, lbuf→next and the pointer cell→prev of the first occupied cell

- determine request of this domain:
  - it has enough cells and nothing happens
  - it has too many cells/particles, they are sent to the next domain:
    1. determine number of cells and particles to be sent to next domain
    2. inform next domain of these numbers
    3. pack cells and particles linked to them (start with cell right) and send them to the next domain
    4. delete memory which is still allocated by already sent cells and particles and update n_right, n_cells, n_el, n_ion, n_part, rbuf and Rbuf, rbuf→prev and the pointer cell→next of the last occupied cell
  - it needs more particles:
    1. get the number of cells and particles which actually will be received from next
    2. allocate memory for cells and particles and link cells to domain, link all the particles to the last cell (right) and update n_right, n_cells, n_part
    3. receive and unpack the cells and particles from next and determine number of electrons/ions received

diag.count:
increase diagnostic counters for: poisson, snapshot, velocity, flux, reflex, spacetime, energy, trace, el_phasespace:

sim.count_reorganize: increase reorganize counter

zeit.add: update clock

- after the main loop has finished, the clock is stopped

main_exit:

- stop this PVM task
- exit program

A Moving frame for oblique incidence

Oblique incidence is treated using Bourdier's method [18]. As shown in Fig. 29, a Lorentz transformation is performed from the laboratory frame L to a frame M which moves in the plane of incidence parallel to the plasma surface such that the laser pulse is normally incident in M.

Frame M moves with velocity $\boldsymbol{v}_f = \hat{\mathbf{y}}\, c \sin\alpha$ in y-direction of frame L. Therefore the transformation parameters are

$$\beta = \sin\alpha, \qquad \Gamma = \frac{1}{\sqrt{1-\beta^2}} = \frac{1}{\cos\alpha}. \tag{26}$$

A four-vector $x^\mu = (x_0, x_1, x_2, x_3)^T$ transforms as $x^\mu(M) = \mathcal{L}\, x^\mu(L)$, where the matrix $\mathcal{L}$ is given by

$$\mathcal{L} = \begin{pmatrix} \Gamma & 0 & -\beta\Gamma & 0 \\ 0 & 1 & 0 & 0 \\ -\beta\Gamma & 0 & \Gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The results are:

1. Frequency and wavevector, $k^\mu = (\omega/c, \boldsymbol{k})^T$

$$\begin{aligned} \omega^L &= \omega_0 & \omega^M &= \omega_0 \cos\alpha \\ \boldsymbol{k}^L &= \begin{pmatrix} k_0\cos\alpha \\ k_0\sin\alpha \\ 0 \end{pmatrix} & \boldsymbol{k}^M &= \begin{pmatrix} k_0\cos\alpha \\ 0 \\ 0 \end{pmatrix}. \end{aligned} \tag{27}$$

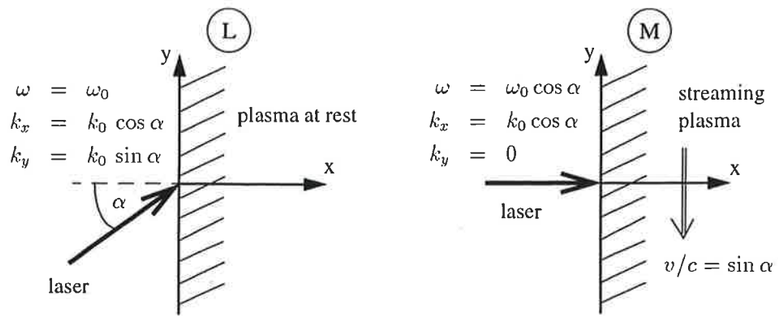

Fig. 29: *Laboratory frame L with oblique incidence and moving frame M with normal incidence. In M, the plasma streams parallel to the surface. Frequency and* $\mathbf{k}$*-vector of the incident light are shown for both frames.*

Laser light is Doppler-shifted. Of course, the vacuum dispersion $\omega = c\,k$ holds in both frames.

2. Velocities, $u^\mu = (\gamma c, \gamma \boldsymbol{v})^T$

$$\begin{pmatrix} v_x^M \\ v_y^M \\ v_z^M \end{pmatrix} = \frac{1}{1-\beta v_y^L/c} \begin{pmatrix} v_x^L/\Gamma \\ v_y^L - \beta c \\ v_z^L/\Gamma \end{pmatrix}.$$

3. Densities, $j^\mu = (c\,\rho, \boldsymbol{j})^T$

$$\begin{pmatrix} c\,\rho^M \\ j_x^M \\ j_y^M \\ j_z^M \end{pmatrix} = \begin{pmatrix} \Gamma(c\,\rho^L - \beta\, j_y^L/c) \\ j_x^L \\ \Gamma(-\beta\, c\,\rho^L + j_y^L) \\ j_z^L \end{pmatrix}.$$

For $j_y^L(x) \equiv 0$ we find that densities of each species transform as

$$n^M = \Gamma n^L, \qquad \rho = \pm z\, e\, n, \tag{28}$$

which follows from Lorentz contraction in y-direction. Since the critical density transforms as

$$n_c^M = (\omega^M)^2 \varepsilon_0 m_e/e^2 = \Gamma^{-2} n_c^L,$$

the *scaled* densities n/n_c transform as

$$\left(\frac{n}{n_c}\right)^M = \Gamma^3 \left(\frac{n}{n_c}\right)^L. \tag{29}$$

4. Fields

$$\begin{aligned} E_x^M &= \Gamma(E_x^L + \beta\, c\, B_z^L), & c\,B_x^M &= \Gamma(c\,B_x^L - \beta\, E_z^L), \\ E_y^M &= E_y^L, & B_y^M &= B_y^L, \\ E_z^M &= \Gamma(E_z^L - \beta\, c\, B_x^L), & c\,B_z^M &= \Gamma(c\,B_z^L + \beta\, E_x^L). \end{aligned} \tag{30}$$

P-Polarization

$$\begin{aligned} \boldsymbol{E}^L &= \begin{pmatrix} -E_0\sin\alpha \\ E_0\cos\alpha \\ 0 \end{pmatrix}, & \boldsymbol{E}^M &= \begin{pmatrix} 0 \\ E_0\cos\alpha \\ 0 \end{pmatrix}, \\ c\,\boldsymbol{B}^L &= \begin{pmatrix} 0 \\ 0 \\ E_0 \end{pmatrix}, & c\,\boldsymbol{B}^M &= \begin{pmatrix} 0 \\ 0 \\ E_0\cos\alpha \end{pmatrix}. \end{aligned} \tag{31}$$

S-Polarization

$$\begin{aligned} \boldsymbol{E}^L &= \begin{pmatrix} 0 \\ 0 \\ E_0 \end{pmatrix}, & \boldsymbol{E}^M &= \begin{pmatrix} 0 \\ 0 \\ E_0 \cos\alpha \end{pmatrix}, \\ c\,\boldsymbol{B}^L &= \begin{pmatrix} E_0 \sin\alpha \\ -E_0 \cos\alpha \\ 0 \end{pmatrix}, & c\,\boldsymbol{B}^M &= \begin{pmatrix} 0 \\ -E_0 \cos\alpha \\ 0 \end{pmatrix}. \end{aligned} \tag{32}$$

In summary, field *amplitudes* are reduced by a factor $\cos\alpha$. Since frequency is reduced by the same amount, the *scaled* amplitudes

$$\frac{e|\boldsymbol{E}|}{m_e\omega c}, \qquad \frac{e|\boldsymbol{B}|}{m_e c}$$

do *not* change.

B Units

As long as atomic physics is *not* included in LPIC++, the laser frequency has not to be specified. All quantities can be given in appropriate units scaling with laser frequency or wavelength:

1. Coordinates are scaled to the laser wavelength λ_0 in the laboratory frame, since lengths in x-direction do not change when performing a Lorentz transformation in y-direction,

   $$x/\lambda_0 =: x'.$$

   Nevertheless, the laser wavelength in the moving frame is Doppler-shifted, $\lambda = \Gamma \cdot \lambda_0$, $\omega = \omega_0/\Gamma$, $\Gamma = 1/\cos\alpha$.

2. Time is scaled to the laser period τ in the *moving* frame, $t/\tau =: t'$, where $\tau = \Gamma \cdot \tau_0$ and $\tau_0 = 2\pi/\omega_0$.

3. Velocities are scaled to the velocity of light in vacuum,

   $$\boldsymbol{v}/c =: \boldsymbol{v}',$$

   and the dimensionless momentum is introduced,

   $$\boldsymbol{p}/(m_e\, c) = \gamma \boldsymbol{v}/c =: \boldsymbol{u}.$$

4. Micro particle masses are given in units of the electron mass m_e, and micro particle charges are given in units of e,

   $$m/m_e =: m', \qquad q/e =: z,$$

   so that the *specific charge* q/m is scaled as

   $$\frac{q/m}{e/m_e} = \frac{z}{m'} =: \mathrm{zm}.$$

5. Fields are scaled as follows,

   $$\frac{e\boldsymbol{E}}{m_e\omega c} =: \boldsymbol{E}', \qquad \frac{e\boldsymbol{B}}{m_e\omega} =: \boldsymbol{B}',$$

   where ω is the laser frequency in the moving frame M.

6. Vector potential and electrostatic potential are scaled as

   $$\frac{e\boldsymbol{A}}{m_e c} =: \boldsymbol{A}', \qquad \frac{e\Phi}{m_e c^2} =: \Phi'.$$

7. Densities are given in units of the critical density $n_c = \omega^2 \varepsilon_0 m_e / e^2$ in the moving frame M,

$$n/n_c =: n',$$

and charge densities in units of $e \cdot n_c$,

$$\frac{\rho}{e n_c} =: \mathrm{zn},$$

for each species seperately.

8. Current densities are given in units of $e n_c c$ for each species,

$$\frac{\boldsymbol{j}}{e n_c c} =: \boldsymbol{j}' = \mathrm{zn} \cdot \boldsymbol{v}'.$$

9. Energies are given in units of $(m_e c^2\, n_c\, A\, \Delta x)$,

$$\frac{W}{m_e c^2\, n_c\, A\, \Delta x} =: W',$$

where A is an arbitrary cross section (one spatial dimension) and Δx is the grid spacing. Then the total energy, sum of field and kinetic energies can be written as

$$W'_{total} = \frac{1}{2} \sum_{cells} ((\boldsymbol{E}')^2 + (\boldsymbol{B}')^2) + \sum_{particles} m' \cdot n' \cdot (\gamma - 1),$$

In these units, the equations of motion read

$$\begin{aligned} \frac{\partial}{\partial t'} \boldsymbol{u} &= 2\pi\, \mathrm{zm} \left(\boldsymbol{E}' + \frac{1}{\gamma} \boldsymbol{u} \times \boldsymbol{B}' \right), \qquad \boldsymbol{u} = \gamma\, \boldsymbol{v}' \\ \frac{\partial}{\partial t'} x' &= \Gamma v'_x, \qquad \Gamma = 1/\cos\alpha, \end{aligned}$$

i.e. since $v'_x < 1$, the maximum distance a particle can move during one period in M is less than $\Gamma \cdot \lambda_0 = \lambda$, the laser wavelength in M.
The propagation of E_x (Eq. 18) reads in these units

$$\partial_{t'} E'_x = -2\pi\, j'_x. \tag{33}$$

The propagation of transverse fields (Eq. 11) reads in these units

$$(\partial_{t'} \pm \Gamma\, \partial_{x'})\ (F')^{\pm} = -\pi\, j'_y, \qquad (F')^{\pm} := \frac{1}{2} \left(E'_y \pm B'_z \right) \tag{34}$$

$$(\partial_{t'} \mp \Gamma\, \partial_{x'})\ (G')^{\pm} = -\pi\, j'_z, \qquad (G')^{\pm} := \frac{1}{2} \left(E'_z \pm B'_y \right). \tag{35}$$

Notice $\Delta x'/\Gamma = \Delta t'$. The dimensionless time step $\Delta t'$ is the inverse number of time steps per period **spp** ($\Delta t' = 1/\mathbf{spp}$), and the dimensionless grid step $\Delta x'$ is the inverse number of cells per laboratory–wavelength λ_0 ($\Delta x' = \Delta x = 1/\mathbf{cells\_per\_wl}$). Therefore the number of time steps per period transforms as

$$\mathbf{spp} = \Gamma \cdot \mathbf{cells\_per\_wl}.$$

This means, two simulations with equal parameters but different angles of incidence will take different amounts of cpu time, since the number of time steps per laser cycle is different.

References

[1] D. Strickland and G. Mourou, *Compression of amplified chirped optical pulses*, Opt. Comm. **56**, 219 (1985); M.D. Perry and G. Mourou, *Terawatt to Petawatt Subpicosecond Lasers*, Science **264**, 917 (1994).

[2] F. Brunel, *Not-So-Resonant, Resonant Absorption*, Phys. Rev. Lett. **59** (1), 52 (1987).

[3] P. Gibbon and A. Bell, *Collisionless Absorption in Sharp-Edged Plasmas*, Phys. Rev. Lett. **68** (10), 1535 (1992).

[4] H. Ruhl and P. Mulser, *Relativistic Vlasov simulation of intense fs laser pulse–matter interaction*, Phys. Lett. A **205**, 388 (1995);
H. Ruhl, *Electron jets produced by ultrashort laser pulses*, J. Opt. Soc. Am. **B 13**, 388 (1996).

[5] P. Gibbon, *Efficient Production of Fast Electrons from Femtosecond Laser Interaction with Solid Targets*, Phys. Rev. Lett. **73** (5), 664 (1994).

[6] A. Pukhov and J. Meyer-ter-Vehn, *Relativistic Magnetic Self-Channeling of Light in Near-Critical Plasma: Three-Dimensional Particle-in-Cell Simulation*, Phys. Rev. Lett. **76** (21), 3975 (1996).

[7] S. Wilks, W. Kruer, and W. Mori, *Odd Harmonic Generation of Ultra-Intense Laser Pulses Reflected from an Overdense Plasma*, IEEE Trans. Plasma Sci. **21** (1), 120 (1993).

[8] S. Bulanov, N. Naumova, and F. Pegoraro, *Interaction of an ultrashort, relativistically strong laser pulse with an overdense plasma*, Phys. Plasmas **1** (3), 745 (1994).

[9] P. Gibbon, *Harmonic Generation by Femtosecond Laser–Solid Interaction: A Coherent 'Water–Window' Light Source?*, Phys. Rev. Lett. **76** (1), 50 (1996).

[10] R. Lichters, J. Meyer-ter-Vehn, and A. Pukhov, *Short-pulse laser harmonics from oscillating plasma surfaces driven at relativistic intensity* Phys. Plasmas **3** (9), 3425 (1996);
R. Lichters and J. Meyer-ter-Vehn, *High laser harmonics from plasma surfaces: intensity and angular dependence, cutoffs and resonance layers at density ramps*, in

Multiphoton Processes 1996 (Institute of Physics Publishing, Bristol and Philadelphia, 1997), pp. 221-230;
R. Lichters *et al.*, *Radiation at $2\omega_p$ from inverse two-plasmon decay in overdense plasma driven by ultra-short laser pulses*, submitted to Phys. Rev. Lett. (1997).

[11] M. Tabak, J. Hammer, M. Glinsky, W. Kruer, S. Wilks, J. Woodworth, E. Campbell, M. Perry, and R. Mason, *Ignition and high gain with ultrapowerful lasers*, Phys. Plasmas **1**, 1626 (1994).

[12] A. Pukhov and J. Meyer-ter-Vehn, *Fast Ignitor Concept. Numerical Simulation.*, Gesellschaft für Schwerionenforschung, Report GSI-95-06, ISSN 0171-4546 (1995).

[13] A. Pukhov and J. Meyer-ter-Vehn, *Laser Hole Boring into Overdense Plasma and Relativistic Electron Currents for Fast Ignition of ICF Targets*, submitted to Phys. Rev. Lett. (1997).

[14] R. Lichters, *Relativistische Wechselwirkung intensiver kurzer Laserpulse mit überdichten Plasmen: Erzeugung hoher Harmonischer*, PhD thesis, Technische Universität München (1997) and MPQ Report **219**, Max-Planck-Institut für Quantenoptik, D-85748 Garching (1997).

[15] C.K. Birdsall and A.B. Langdon, *Plasma physics via computer simulation* (Adam Hilger, New York, 1991).

[16] J. Villasenor and O. Buneman, *Rigorous charge conservation for local electromagnetic field solvers*, Computer Physics Communications **69**, 306 (1992).

[17] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine System* (MIT Press, Cambridge, USA, 1994), `http://www.netlib.org/pvm3`.

[18] A. Bourdier, *Oblique incidence of a strong electromagnetic wave on a cold inhomogeneous electron plasma. Relativistic effects*, Phys. Fluids **26** (7), 1804 (1983).

[19] J. Boris, in *Proc. Fourth Conf. Num. Sim. Plasmas* (Naval Res. Lab., Washington, D.C., 1970), S. 3.

[20] A. Sommerfeld, *Optik. Vorlesungen über theoretische Physik*, Band 4, 3. Aufl. (Thun: Deutsch, 1989).

[21] M. Born and E. Wolf, *Principles of Optics, Electromagnetic Theory of Propagation, Interference and Diffraction of Light* (Pergamon Press, Ltd., Oxford, 1965).