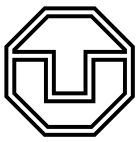

TECHNISCHE UNIVERSITÄT DRESDEN

Generating and testing flying focus laser pulses with Lasy for PIConGPU simulations

Bachelor-Arbeit
zur Erlangung des Hochschulgrades
Bachelor of Science
im Bachelor-Studiengang Physik

vorgelegt von

EDGAR MARQUARDT
geboren am 07.09.2002 in Leipzig

Institut für Kern- und Teilchenphysik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
und
Institut für Strahlenphysik
Helmholtz Zentrum Dresden Rossendorf
2025

Eingereicht am 05. Januar 2026

1. Gutachter: Prof. Dr. Ulrich Schramm
2. Gutachter: Prof. Dr. Thomas Cowan

Summary

Abstract

English: Laser wakefield acceleration is a novel, compact method to accelerate electrons. These accelerators face a problem called "dephasing", where due to the reduced velocity of the laser in the plasma medium the electrons can only be accelerated for short distances. One proposed solution to this problem is the so called flying focus laser. Experimentally, it is generated using an axiparabola and a radial group delay echelon.
To study the laser plasma interaction with flying focus lasers, Particle-in-Cell simulations, such as PIConGPU, are used. A flying focus laser has a complicated shape and phase structure and, therefore, was not available in PIConGPU. However, the Lasy library can simulate a laser pulse interacting with optical elements and can, therefore, model a flying focus pulse. This work combines these two and makes flying focus pulses generated in Lasy available in PIConGPU. Furthermore, this technique also enables modeling other laser pulses in PIConGPU and the technique has been validated with both Gaussian and flying focus lasers. For the latter, inconsitencies between predicted and measured laser velocities and beam waists were discovered and are extensively discussed in this work.

Abstract

Deutsch: "Laser-Wakefield-Acceleration" ist eine neue, kompakte Methode um Elektronen zu beschleunigen. Solche Beschleuniger haben ein Problem, das "Dephasing" genannt wird, weil die Elektronen wegen der reduzierten Geschwindigkeit des Lasers in Plasma nur über kurze Strecken beschleunigt werden können. Eine mögliche Lösung für dieses Problem ist der so genannte "Flying-Focus-Laser". Dieser kann im Experiment mithilfe einer Axiparabel und eines radialen Verzögerungs-echelons erzeugt werden. Um die Interaktionen zwischen Laser und Plasma mit dem Flying-Focus-Laser genauer zu untersuchen nutzt man "Particle-in-Cell" Simulationen wie PIConGPU. Ein Flying-Focus-Laser hat eine komplizierte Form und Phasenstruktur und war deshalb in PIConGPU bisher nicht verfügbar. Mit der Python-Bibliothek Lasy kann man allerdings einen Laserpuls simulieren, der mit optischen Elementen interagiert, und dadurch einen Flying-Focus-Laser modellieren. Diese Arbeit kombiniert die beiden Methoden und macht den in Lasy erzeugten Flying-Focus-Laser in PIConGPU verfügbar. Außerdem ermöglicht die beschriebene Technik auch andere Laserpulse in PIConGPU zu modellieren und wurde sowohl mit Gaußpulsen als auch mit dem Flying-Focus-Laser validiert. Für letzteren tauchten Inkonsistenzen zwischen vorhergesagten und beobachteten Lasergeschwindigkeiten und Strahlbreiten auf, die in dieser Arbeit ausführlich diskutiert werden.

Contents

Chapter 1

Introduction

1.1 Motivation: Laser wakefield acceleration (LWFA)

Since the invention of LWFA by Tajima and Dawson [18], compact accelerators are a useful tool in radiation physics. It enables more labs to work with high energy electrons for their research. This in turn enables technologies using these electrons, for example free electron lasers which can generate laser pulses in otherwise unreachable frequency ranges. Our Group has used and researched them for a while now and is working on the cutting edge of this technology, see for example [6], [9] and [11].
LWFA works as follows: A laser is focused into a gas, usually hydrogen, where it needs to be intense enough to ionize the gas and push the electrons out of the way. In figure 1.1 this happens on the right of the laser pulse. Behind the laser they come back together which produces extreme electric fields on the order of several 100GV/m (see [7]) in the propagation direction of the laser. This accelerating field can be seen in figure 1.1 in blue. Because these fields can become up to 3 orders of magnitude stronger than a traditional radio-frequency accelerator cavity, they can accelerate electrons to multiple GeV over a distance of just centimetres [7]. Figure 1.1 shows the electrons with a colour gradient to show their energy. The yellow region is where the electron bunch is currently being accelerated by riding on the laser wakefield.

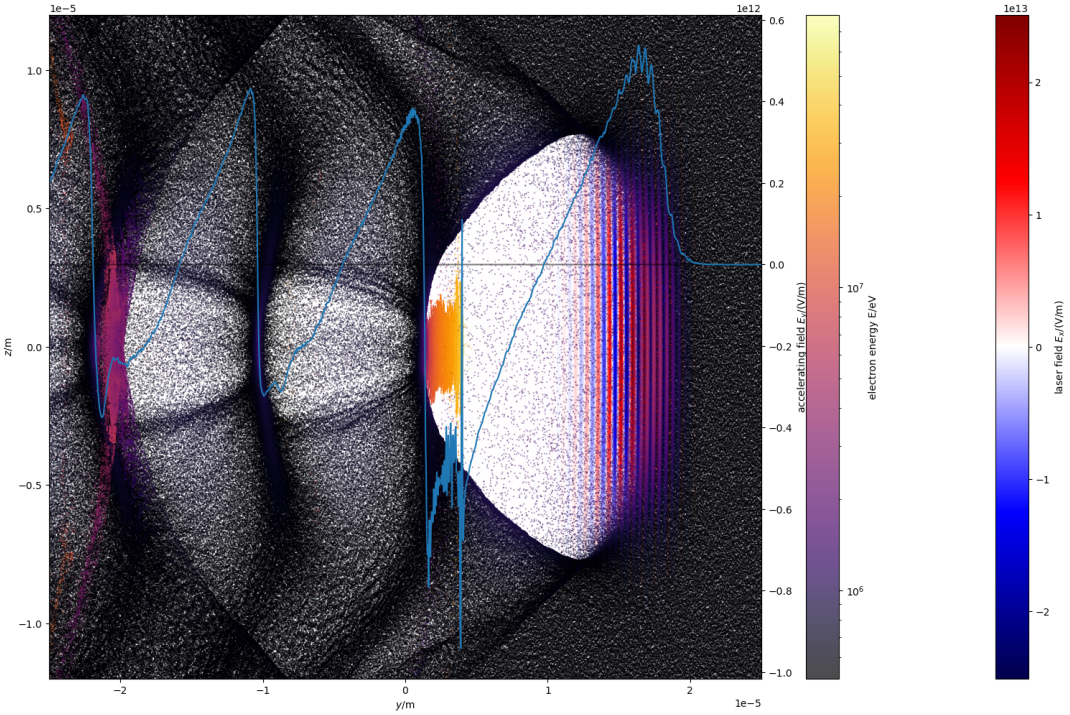

Figure 1.1: PIConGPU simulation of laser wakefield acceleration. Displayed are the electrons, coloured according to their energy E, the electric field of the laser pulse E_x and the on-axis accelerating field E_y in blue.

1.2 The problem of Dephasing

LWFA is known to be facing three major problems: Depletion, Diffraction and Dephasing. Depletion happens, when the laser pulse driving the wake has given so much of its energy to the electrons that it can no longer drive the wake. Diffraction is the laser pulse getting out of focus and therefore losing the wake. Dephasing describes the process by which electrons are accelerated to velocities over the speed of light in the plasma. Because of the high fields in the wake of the laser pulse this happens after a few centimetres of acceleration. The electrons can now slowly approach the laser pulse until the field in the propagation direction reverses and no longer accelerates the electrons and instead decelerates them. In figure 1.1 this reversal can be seen to the left of the laser pulse. This effect puts an upper limit to the energy an electron can leave the acceleration region with. A common solution is to accelerate the electrons in multiple stages to increase the energy but this is complicated and inconvenient.
Debus et al [7] have shown, that it is possible to solve Dephasing (and also Depletion) by using two separate laser pulses with tilted wave fronts. However, this is complicated to achieve. According to Palastro et al [15] as well as Miller et al [13], Dephasing could also be solved using a so called flying focus setup. It is made out of a radial group delay echelon and an axiparabola.

1.3 The Axiparabola

In 2019 Smartsev et al proposed and tested a new optical device called the axiparabola [16]. It is a mirror shaped similarly to an off-axis parabola and is defined by three parameters: the nominal focal length f_0, the length of the focal region δ and the radius R. An axiparabola does not only focus a pulse onto one specific point but onto a line on the optical axis of the length δ. It achieves this by focusing

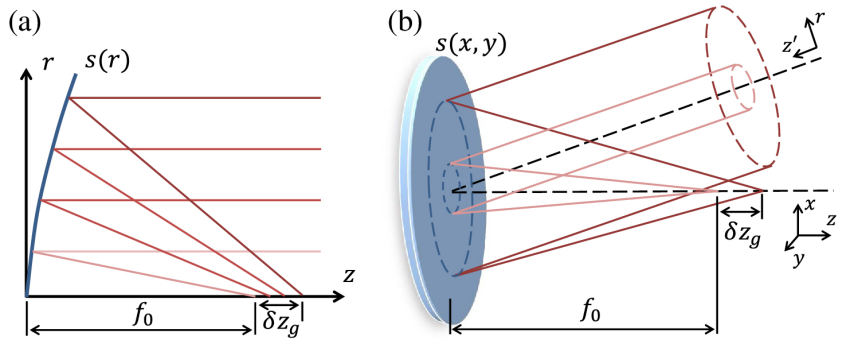

Figure 1.2: Schematic of how an axiparabola focuses light into the focal region. Rays close to the optical axis are focused to a focus length of f_0 and rays far from the axis are focused closer to $f_0 + \delta$. (a) On-axis case, (b) off-axis case. Image taken from Smartsev et al[16].

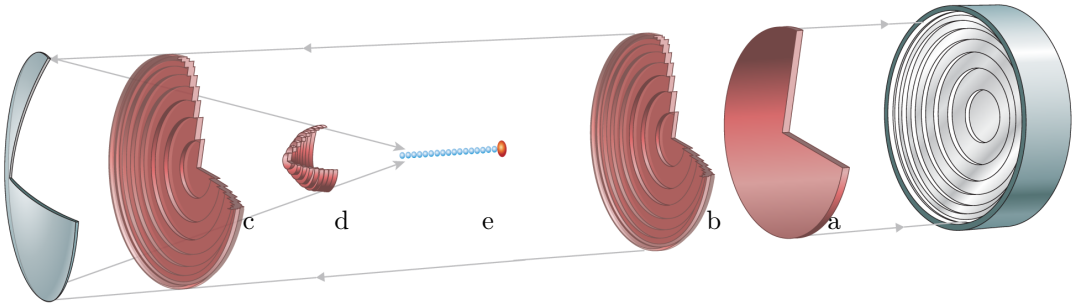

Figure 1.3: Schematic of the flying focus setup. A flat laser pulse (a) is first reflected by the radial group delay echelon (b). It imparts a time delay onto the pulse without aberrations. The pulse (c) is then reflected by the axiparabola, which focuses the pulse (d) into the focal region δ (e), where it can theoretically drive a laser wakefield accelerator without dephasing. Image taken from Palastro et al [15].

the pulse to different focus lengths at different radii:

$$f(r) = f_0 + \delta \left(\frac{r}{R}\right)^2 \tag{1.1}$$

In figure 1.2 on the left side is shown, how this should be understood. The sag function describing the shape of the axiparabola looks as follows:

$$s_f(r) = \frac{1}{4f_0} r^2 - \frac{\delta}{8f_0^2 R^2} r^4 + \frac{\delta\left(R^2 + 8f_0\delta\right)}{96 f_0^4 R^4} r^6 + \mathcal{O}(r^8)[16] \tag{1.2}$$

Ambat et al [4] also proposed a closed formula:

$$s_f(r) = \frac{R^2}{4\delta} \ln\left(1 + \frac{\delta}{f_0}\left(\frac{r}{R}\right)^2\right) \tag{1.3}$$

According to Ambat et al [4] the beam waist, the transversal radius at which field values fall to $1/e$ of their maximum, between $f_0 < z < f_0 + \delta$ should follow roughly the relation

$$w(z) \approx \frac{\lambda_0 f_0}{\pi R} \sqrt{\frac{\delta}{z - f_0}}. \tag{1.4}$$

This can of course only apply at sufficient distance from f_0, because otherwise $\lim_{z\to f_0} w(z) = \infty$. The time when the focus reaches a distance z from the axiparabola is roughly:

$$t_f(z) \approx \frac{1}{c}\left(z + \frac{(z-f_0)R^2}{\delta \cdot 2z} - 2s_f\left(\sqrt{\frac{(z-f_0)R^2}{\delta}}\right)\right) \tag{1.5}$$

$$= \frac{1}{c}\left(z + \frac{(z-f_0)R^2}{\delta \cdot 2z} - \frac{R^2}{2\delta}\ln\left(1 + \frac{z-f_0}{f_0}\right)\right) \tag{1.6}$$

In figure 1.3 the axiparabola can be seen on the left.

1.4 Spatio-temporal control

While the axiparabola allows for spatial control overthe focal region, one can not steer when it arrives at the place. The time just follows equation 1.5 and does not have more parameters. However, because the focus legth depends on r and, therefore, the light forming the focus at each z coordinate comes from a different radius r this behavior can be modified using a stepped mirror made up of flat, concentric rings of mirror surface that are at different depths half a wavelegth apart, a so called radial group delay (RGD) echelon. By delaying the laser pulse in dependence of r by $\Delta t = \tau_D(r)$, it is possible to adjust $z = z(r(t)) = z(t)$ as needed by setting the function $\tau_D(r)$ accordingly. This concept was proposed by Palastro et al [15] to solve the dephasing problem. Figure 1.3 shows this setup with the echelon on the right.

A pulse that has gone through both the RGD echelon and then the axiparabola is called a (ultrafast) flying focus pulse. Ambat et al [4] managed to achieve foci moving almost arbitrarily fast along the focal line with this technique.

Our goal for this work was to replicate this and make the flying focus laser pulse available in the PIConGPU simulation software, to make LWFA simulations with it possible.

Chapter 2

Simulation tools

For this work laser pulses were only simulated. The following tools were used for this:

2.1 Lasy

Lasy[1] stands for LAser manipulations made eaSY. It is a python library designed to simulate laser pulses on basis of the axiprop library. With it one can initialise a laser pulse using one of the provided profiles or from an array or file. Lasy offers a range of optical elements that can be applied to the pulse. One can then propagate the laser pulse by some distance. The standard propagation method uses axiprop, which has an angular spectrum propagator. Other propagation methods are also currently in development for Lasy.
In lasy the complex envelope ε of the electric field E_{pol}, defined by $\mathrm{Re}\left(\varepsilon \cdot e^{i\omega t}\right) = E_{pol}$, is rasterised on a grid. There are two options to do this: Either using `dim = "xyt"`, a cartesian transversal grid and the time axis in the longitudinal direction, or using `dim = "rt"`, a 2D representation with radius and time on the axes and the option to have multiple azimuthal modes to represent not perfectly rotationally symmetrical lasers.
Lasy assumes, that the laser pulse will move at the speed of light c and moves its grid and its time axis along accordingly. It only ever shows the time relative to a standard speed-of-light-movement. This can be seen in figure 3.1 on the time axis.

2.2 openPMD

openPMD[10][2] stands for open (source) particle mesh data standard. It is a meta-data standard for mesh data like fields and particle data. The corresponding I/O library openPMD-api is used by a number of simulation software as well as by some measurement software. PIConGPU supports this format both through the openPMD plugin for output, in its checkpoints and as a laser source. Lasy also offers functions to import and export laser data in this file format. The two interpret the format differently, however, so they can not easily communicate in this way. Part of this work was to bridge that gap.

2.3 PIConGPU

PIConGPU[5][3] is a Particle-In-Cell simulation software that is being developed mainly at the HZDR. Being a PIC simulation it divides the simulation space into cells, which each contain a field value and

some number of particles. For every time step it propagates the fields according to the Maxwell equations and pushes the particles along following their momentum and with a correction from interactions from the fields. Many interactions between particles as well as between particles and fields are modeled.

2.4 Additional modules

The following additional python modules were developed for this work. For a full documentation, see [12].

2.4.1 Module `full_field`

The module `full_field` acts as an add-on to Lasy for this work. Among other things it bridges the gap between Lasy and PIConGPU by providing the function

```
full_field.laser_to_openPMD(laser, filename)
```

It saves the full electric field of the laser pulse into an openPMD-compatible file that PIConGPU can read. Because Lasy only calculates the complex envelope ε of the field this full electric field needs to be calculated first. While Lasy does offer a function for this, namely `lasy.utils.laser_utils.get_full_field(laser)`, it only calculates a 2D cut through the lectric field, even if the laser calculation happend in 3D. Therefore in this module a similar function is implemented:

```
full_field.get_full_field(laser)
```

Parts of this function were taken from the Lasy implementation: If neccessary the field envelope is interpolated to allow for more points in the propagation direction to more properly resolve the wavelength of the laser. Then it applies a phase of $\phi = \exp(i\omega_0 t)$ and takes the real part to obtain the full electric field. The new function, however, calculates the 3D field if the laser is already 3D. It also offers some other options, see in the documentation. It is, for example, possible to calculate the field for only the central region of the laser pulse. This is useful because after focusing the laser pulse the Lasy laser object still has the field far out of the actual focused pulse.
After calculating the electric field the `laser_to_openPMD` function calls the function

```
full_field.write_to_openpmd_file(directory, filename, array, extent)
```

It writes an openPMD-compatible file to the specified location containing the array data, in this case the full electric field. Parts of its implemetation are taken from the Lasy function `lasy.utils.openpmd_helper.write_to_openpmd_file` and other parts from the PIConGPU friendly implementation of `PrepRoutines.save_to_openPMD` from PIConGPU's INSIGHT data incorporation by Dietrich [8]. It was neccessary to implement a new function for this because Lasy can only save an openPMD-compatible file containing the complex laser envelope with which PIConGPU can not work. To save on memory space, especially when loading the file into a GPU for a PIConGPU simulation, the data is store in single precision float. It is also possible to only store every nth data point transversally to save more memory space.
Importantly, the function `laser_to_openPMD` is also able to generate and then save the field of a laser that has been simulated in cylindrcal coordinates in Lasy. It does offer some more options as well, among them the option to display the field that has been saved to the file on a symlog plot. This can be seen in figure 2.2. To just calculate and display the electric field of the laser pulse there also is the function

```
full_field.show_field(laser)
```

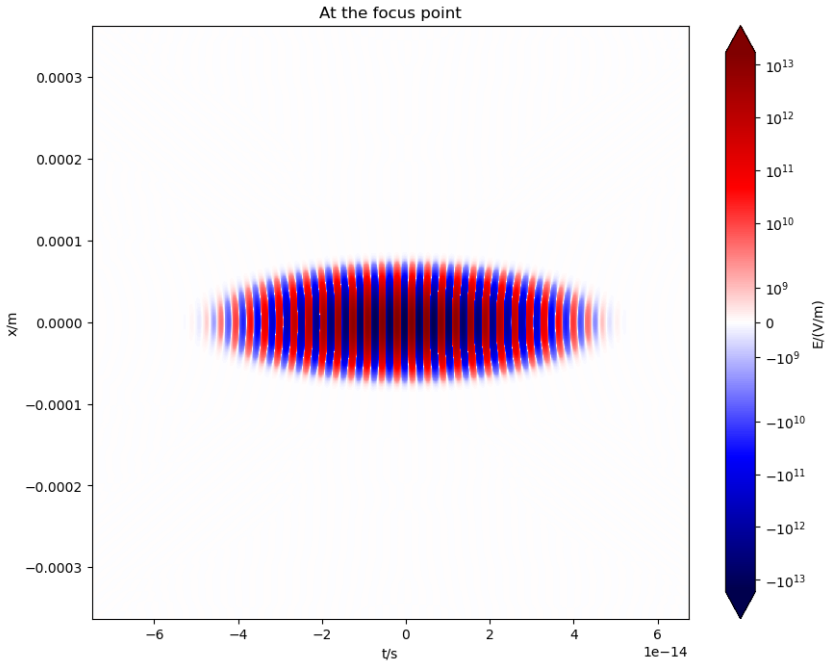

Figure 2.1: Full electric field of a gaussian laser pulse at its focus, according to a Lasy simulation.

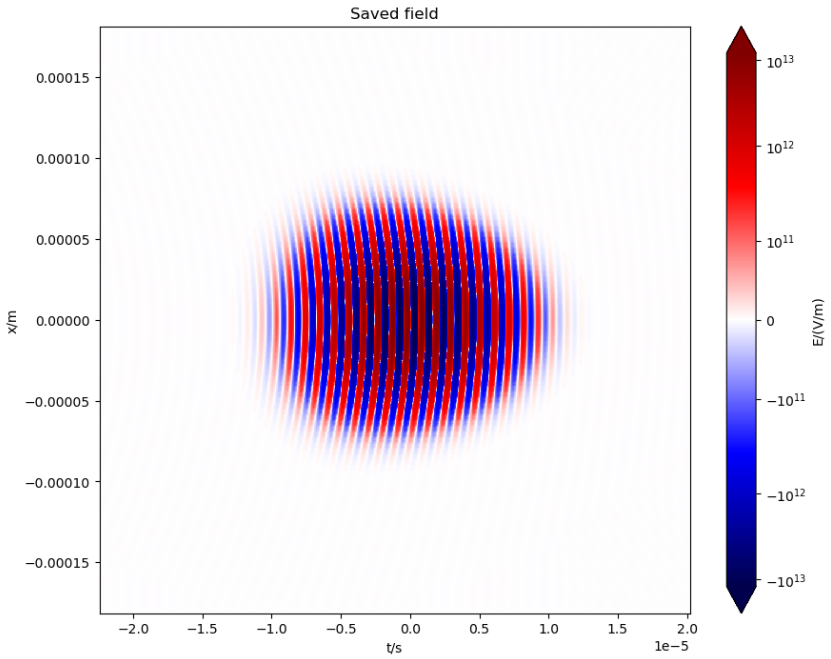

Figure 2.2: The field that has been saved as an openPMD-compatible file after Lasy simulation, one Rayleigh length before the focus.

An example of such a plot can be seen in figure 2.1. The module contains a few more functions that were useful when handling the Lasy laser and that were not available to Lasy natively.

2.4.2 Module `radialGroupDelay`

The module `radialGroupDelay` mainly offers an implementation of the radial group delay echelon as a Lasy optical element. It can be applied to a laser as follows:

```
radialDelay = radialGroupDelay.RadialGroupDelay(tau_D, lambda0)
```

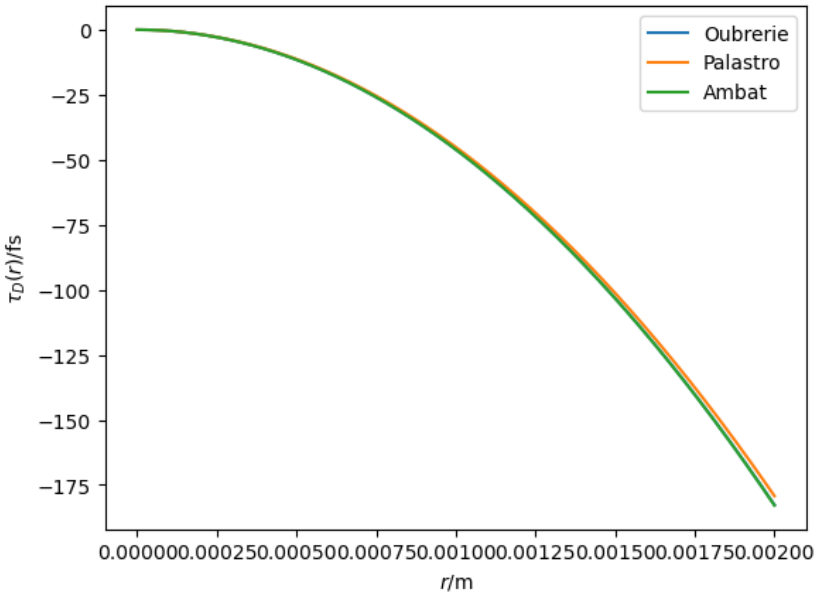

Figure 2.3: The three different $\tau_D(r)$ for $v_f = 1.02c$, according to Oubrerie [14], Ambat [4] and Palastro [15]

```
laser.apply_optics(radialDelay)
```

It requires a function `tau_D(r)`, that describes by how much the pulse should be delayed, depending on the radius r, and the main wavelength of the pulse λ_0. It applies a radially dependent phase according to the following equation taken from Ambat et al [4] eq (11).

$$\phi_D^{ech}(\omega, r) = -\frac{2\omega}{c}\left\{\frac{1}{4}\lambda_0\left[\text{ceil}\left(\frac{c\tau_D(r)}{\lambda 0}\right) + \text{floor}\left(\frac{c\tau_D(r)}{\lambda 0}\right)\right]\right\} \tag{2.1}$$

The stepped shape of the echelon is important to keep the wave fronts straight so that the pulse does not get defocused in any way.

To determine what the function $\tau_D(r)$ should be, the papers [4], [15] and [14] offer three different differential equations but only one of them, Oubrerie et al [14], offers a solution to the equation. The module contains all three options as functions depending on the desired speed of the focus and the used axiparabola:

```
radialGroupDelay.tau_D_const_v(r, v, axiparabola)
radialGroupDelay.tau_D_integrated_ambat(r, v, axiparabola)
radialGroupDelay.tau_D_integrated_palastro(r, v, axiparabola)
```

In two cases the differential equations are numerically integrated, the third is implemented as is. To use these one can, for example, use `functools.partial` to define the desired focus velocity `v` and the used `axiparabola` before setting up the `RadialGroupDelay`.

All these functions were described in the papers as making the focus point of the axiparabola laser travel at some fixed speed that can be set arbitrarily, as long as it is close to the speed of light ($v = v_0 + c$ with $|v_0| << c$). To ensure these equations all described the same function $\tau_D(r)$ figure 2.3 shows them in comparison. They are very similar.

Because an important usecase of this setup is LWFA, the module also provides a function that will return a velocity of the focus that should result in a propagation with vacuum light speed in a plasma:

```
radialGroupDelay.v_plasma(lambda0, n_e)
```

It depends, like the optical index of plasma, only on the laser wavelength $\lambda 0$ and the electron density n_e.

2.4.3 Module `axiparabola_theory`

The module `axiparabola_theory` is a collection of functions for theoretical calculations about the axiparabola and a laser pulse in its focal region. This theory is taken from Ambat et al [4]. Among the functions this module offers are implementations of equations (1.3), (1.4) and (1.5). Because it is important when dealing with PIConGPU simulations the parameters can not only be calculated in dependence of the radius r or the distance along the optical axis z but also in dependence of the arrival time of the focus point t_f. To do this it is neccessary to invert equation (1.5) numerically.
The axiparabola that Lasy provides follows equation (1.2) and with the Smartsev et al [16]. Because they are slightly different (see section 5.2)the module also offers an alternative axiparabola implementation following Ambat et al [4].

2.4.4 Module `showdata`

The module `showdata` contains functions do display metadata and data from openPMD-compatible files and calculate pulse parameters from this data. The main function of the module is

```
showdata.show_file(filename)
```

This function displays the field that is stored in the file at `filename`. It offers the option to use all the following functions in the module. They can, however, be used independently as well. It also offers a variety of other options to modify what the output will look like.
The fields are diplayed using the following function, which is also used by `full_field.show_field`:

```
showdata.show(array, extent)
```

This function plots the contents of `array` on a symlog-plot and offers a variety of options to change the appearance of the result. The plot may look similar to figure 2.4.
The next function in the module is:

```
showdata.show_metadata(filename)
```

This will display all the metadata that describes the fields and particles in the file.
The last two functions offer to calculate parameters of the pulse:

```
showdata.show_w()
```

and

```
showdata.show_lpeak()
```

These calculate the pulse waist and the longitudinal point of the peak, respectively. They each offer multiple methods to do this. One is calculating statistically, one is fitting a gaussian and, for `show_lpeak` specifically, one is finding the maximum. The statistical method is the standard.

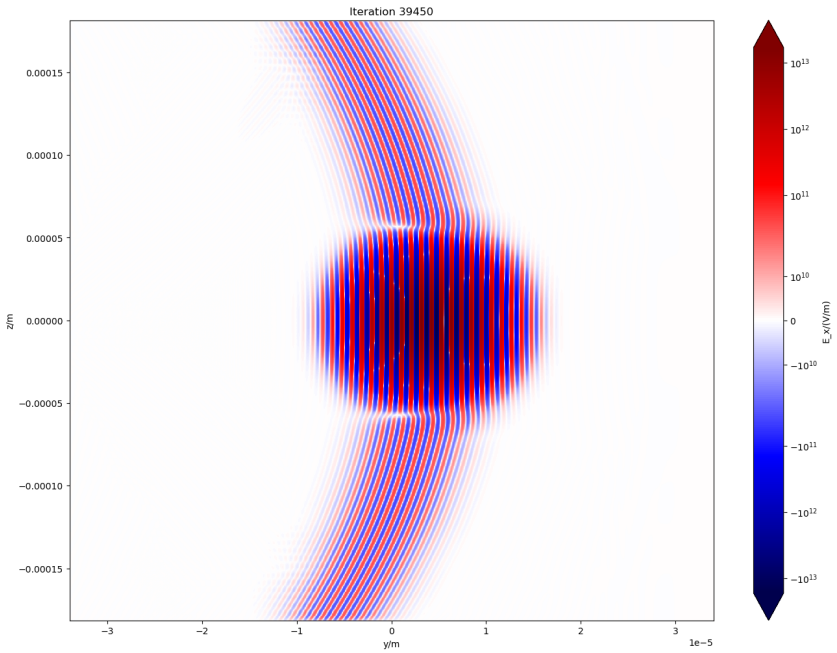

Figure 2.4: Gauss pulse at its focus point according to the PIConGPU simulation. The artifacts to the side of the pulse are orders of magnitude less intense than the main peak of the pulse (see figure 4.1 for reference).

Chapter 3

Gaussian pulses in Lasy

The Lasy library offers functionality to generate, modify and propagate a laser pulse. The documentation can be found at [1].

3.1 Testing Gaussian pulses

A Gaussian laser pulse is one of the simplest and well known pulse shapes. It is, therefore, an important test case. The Gaussian pulse is defined by the main wavelength λ_0, the beam waist at the focus w_0 and the pulse duration τ as well as the maximum amplitude A and the polarisation direction $\hat{e}_{pol}$. At the focus the electric field looks like

$$\vec{E}(r,t) = A\hat{e}_{pol} \cdot e^{i\omega_0 t} \cdot e^{-\frac{r^2}{w_0^2}} \cdot e^{-\frac{t^2}{\tau^2}} \text{ with } \omega_0 = \frac{c}{2\pi\lambda_0}. \tag{3.1}$$

Because Lasy only uses the complex envelope of the laser field, a Gaussian laser initialised in it has a field following

$$\varepsilon(r,t) = A \cdot e^{-\frac{r^2}{w_0^2}} \cdot e^{-\frac{t^2}{\tau^2}}. \tag{3.2}$$

The result of the propagation, displayed by `laser.show()`, can be seen in figure 3.1. Two parameters of the laser pulse after its propagation are of particular interest in this work: the beam waist w and time of arrival of the pulse $t(z)$.
The beam waist theoretically follows the relation

$$w(z) = w_0\sqrt{1 + \left(\frac{z}{z_R}\right)^2} \text{ with } z_R = \frac{\pi w_0^2}{\lambda_0}. \tag{3.3}$$

Here z is measured from the focus. The lasy function `lasy.utils.laser_utils.get_w0(laser.grid, laser.dim)` can be used to calculate the beam waist of a given laser pulse. As one can see in figure 3.2 on the left the two give very similar results.
The time of arrival obviously follows the relation $t(z) = \frac{z}{c}$. In figure 3.2 on the right the deviation from this relation is shown and it turned out to be very little. The measurement was done with a function from the module `full_field`: `full_field.get_tpeak(laser)`.

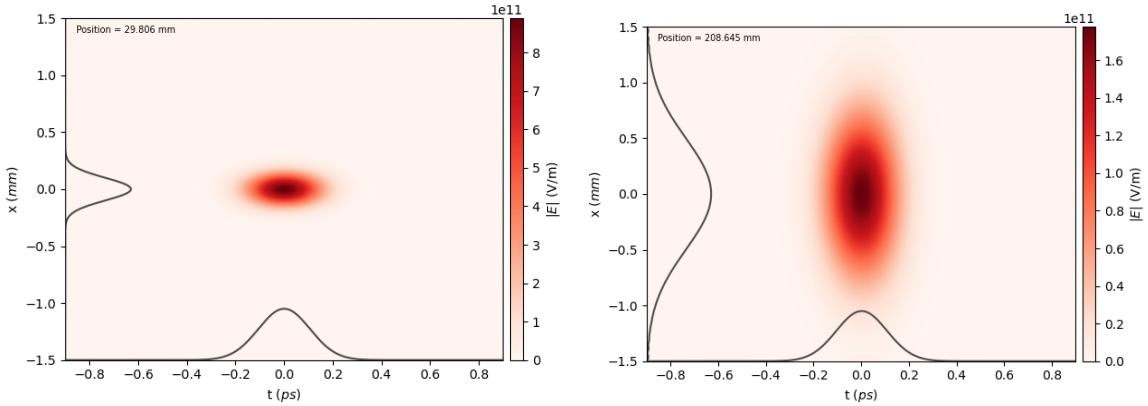

Figure 3.1: A Gaussian pulse newly generated at the focus (left) and propagated $6z_R$ (right) in Lasy.

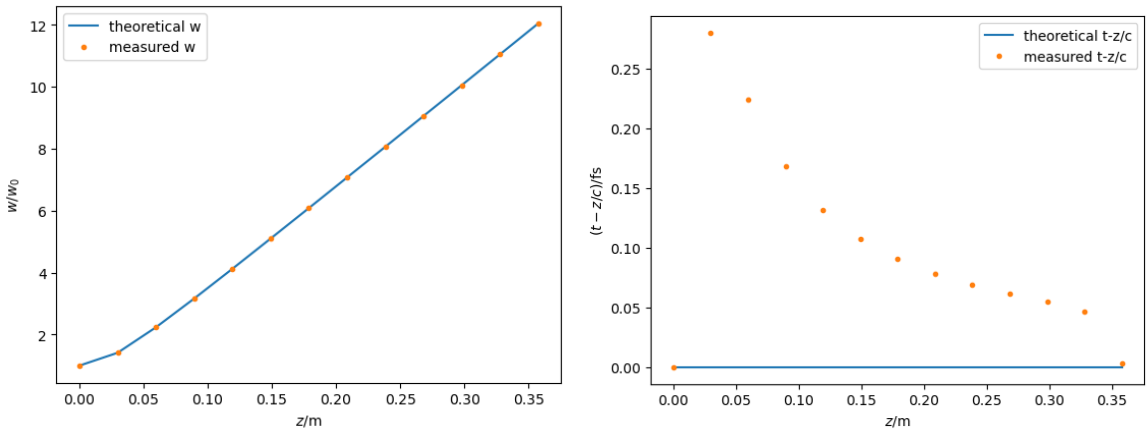

Figure 3.2: The beam waist $w(z)$ (left) and the time difference $t - \frac{z}{c}$ (right) compared to their respective theoretical values of a gaussian pulse initialised in the focus and propagated in Lasy. For comparison: the pulse duration was in this case $\tau = 10$fs.

3.2 Gaussian pulse and parabolic mirror

To generate a pulse for PIConGPU simulations, a Gaussian laser pulse with an initial beam waist of w was reflected by a parabolic mirror of focus length f. The pulse can be seen in figure 3.3 on the left side. To save on space in the RAM and on compute time while gaining precision, the laser was simulated in cylindrical coordinates using the `dim = "rt"` option for Lasy lasers. This assumes, that the laser is radially symmetric (which should be the case here, because both the laser and the mirror have this symmetry), but the computer can handle a simulation with more than 10'000 points transversally.
From the parabolic mirror the laser propagated to one Rayleigh length z_R before the focus by a distance of $l = f - z_R = f - \frac{\pi w_0^2}{\lambda_0}$ with $w_0 = \frac{f\lambda_0}{w\pi}$. Using the module `full_field`, the electric field of the laser was saved to an openPMD-compatible file. The goal was to feed it into PIConGPU one Rayleigh-length from the focus and then simulate it propagating through the focus point to one Rayleigh-length beyond it. In figure 3.3 the pulse now is shown on the right. The saved field can be seen in figure 2.2. To verify the Lasy propagation, the laser propagated into the focus and out of it again by a rayleigh length in Lasy. The beam waist at the focus was consistent with the theoretical value of w_0. The field at the focus is also shown in figure 2.1.

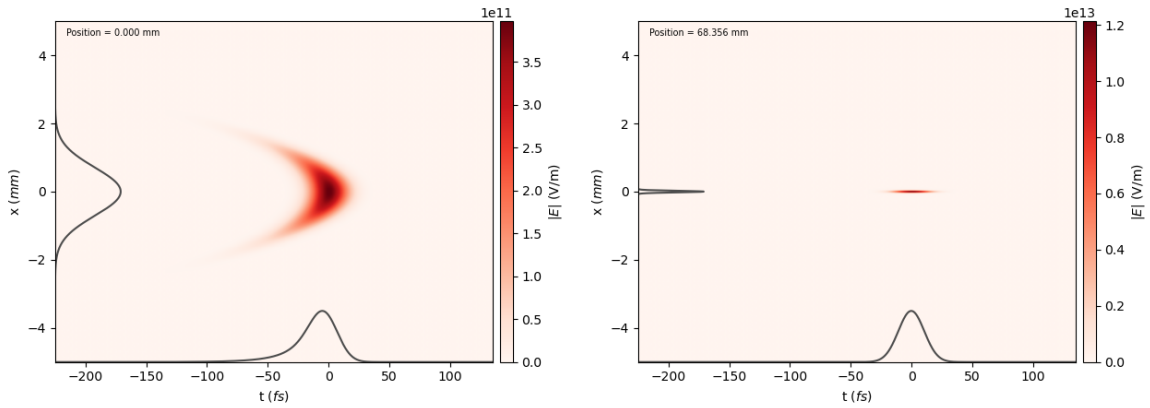

Figure 3.3: A Gaussian laser pulse in the near field of the parabolic mirror (left) and in the far field, one rayleigh length before the focus (right), simulated in Lasy.

Chapter 4

Lasy pulses in PIConGPU

To use the saved laser pulse from the Lasy propagation in a PIConGPU simulation, PIConGPU's incidentField.param file needs to be ajusted to use profiles::FromOpenPMDPulse. The propagation direction in the file must be set to "z" and the mesh name to "E". In order for the file to fit on the GPUs, it is important to change the parameter reservedGpuMemorySize in the file memory.param to accomodate the file's size in addition to what is already required here.

4.1 Courant-Friedrich-Levy (CFL) condition

Dietrich [8], who implemented the profiles::FromOpenPMDPulse, showed that it is important, that in the file simulation.param the time step is the same (or very close to the same) as the time step in the saved laser pulse. (This can easily be achieved by setting the `forced_dt` option to that same time step when calling the function `full_field.laser_to_openPMD` to generate the laser file in Lasy.) However, because it may be neccessary to change the other parameters in the simulation.param file to be able to fit the laser pulse into a reasonable amount of cells, it is important to remember to check the CFL-condition. The size of the simulation cells (Δx, Δy, Δz) needs to be in the following relation with the time step Δt (see [3]):

$$c\Delta t < \frac{1}{\sqrt{\frac{1}{\Delta x} + \frac{1}{\Delta y} + \frac{1}{\Delta z}}} \tag{4.1}$$

However, the values on both sides of the inequality should be close to one another:

$$c\Delta t = \frac{\xi}{\sqrt{\frac{1}{\Delta x} + \frac{1}{\Delta y} + \frac{1}{\Delta z}}} \text{ with } \xi \approx 0.995 \tag{4.2}$$

For this purpose the module `full_field` contains the following function:

```
full_field.cfl_condition()
```

It returns returns the ideal time step for the specified cell size. If a time step is specified it will print how far off it is.
If the deviation between the used and the ideal time step is less than 5% the PIConGPU simulation will work without major problems [3].

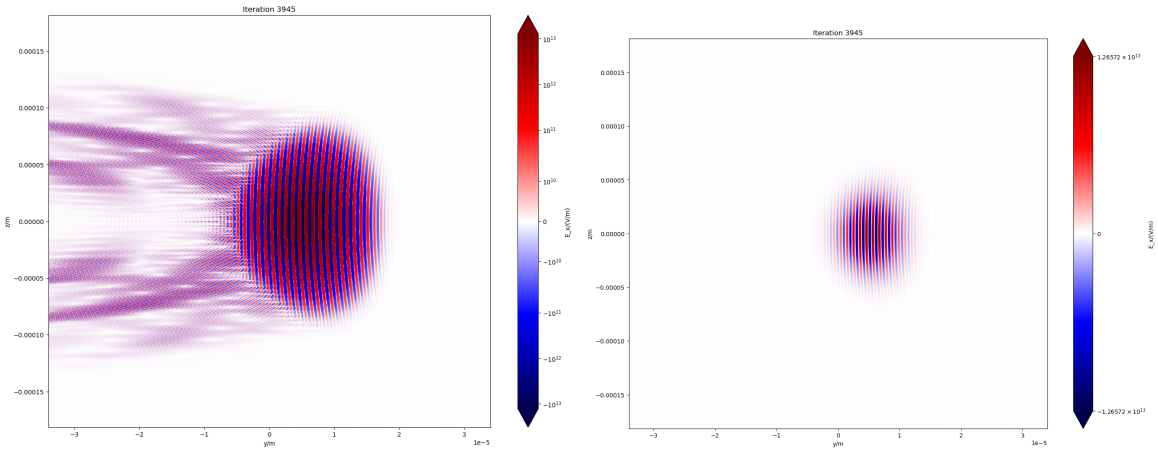

Figure 4.1: A Gaussian laser pulse propagating into the PIConGPU simulation volume. Left: On a symlog plot; right: on a linear plot. The artifacts behind the pulse are orders of magnitude less intense than the main peak of the pulse.

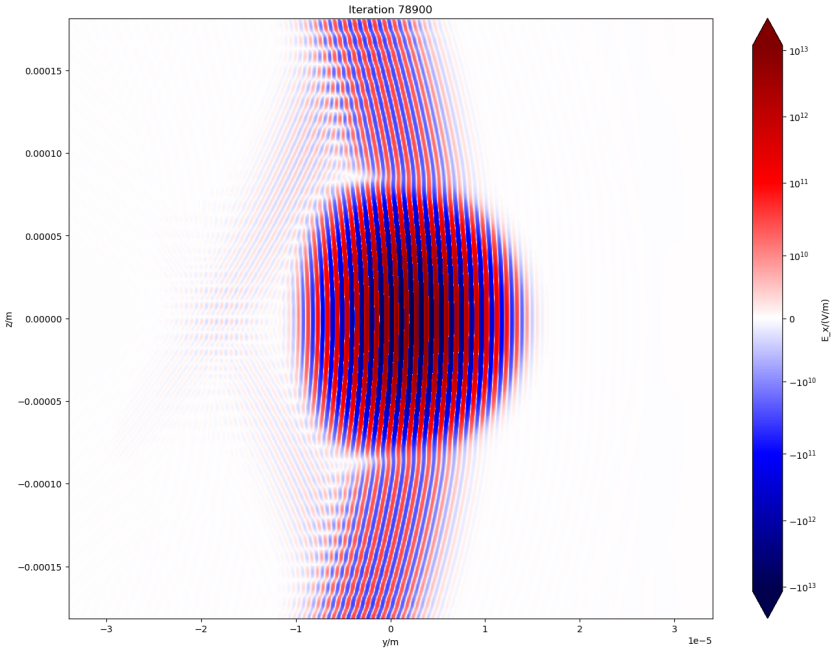

Figure 4.2: A Gaussian laser pulse propagated one Rayleigh length after the parabola focus in PIConGPU. The artifacts visible to the side of the pulse are still orders of magnitude less intense than the main peak of the pulse.

4.2 Comparitative test with the Gauss laser

The laser pulse in the file (see figure 2.2) can be fed into PIConGPU and simulated. Figure 4.1 shows the pulse as it enters the simulation volume. Of note: because PIConGPU uses the y-axis as propagation direction, all figures from its simulations are labeled accordingly. It is still almost a Rayleigh length away from the focus point. Behind the pulse there are some artifacts from the field initialisation method. These are a known issue of the simulation, see [17]. They can even move faster than light speed, as can be seen in figure 2.4 at the focus, where they have moved along significantly. Ignoring those for the moment, however, the waist of the pulse can be measured using `showdata.show_w()` and at the focus the value is very similar to the calculated w_0 from section 3.2. After a propagation of one Rayleigh length past the focus, the pulse looks like in figure 4.2. Here the waist is a lot larger again. It develops over time,

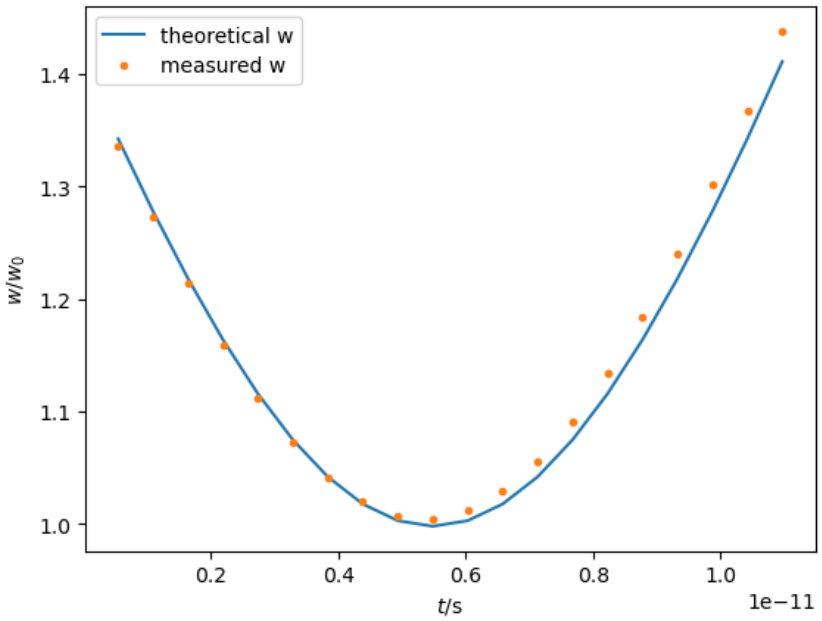

Figure 4.3: The width of the gaussian pulse over time, according to the PIConGPU simulation.

as can be seen in figure 4.3. Here the focus is in the middle and the plot shows the propagation by one Rayleigh length in each direction. It also shows the expected values from equation 3.3 for comparison. They follow a very similar development, though towards the end the simulation artifacts skew the values to be larger. They can achieve this because the values of w are calculated statistically here.
One can also see the shape of the wave fronts, always curved towards the focus point. At the focus point (figure 2.4) they are flat.

Chapter 5

The flying focus laser

The previous chapters on tests with Gaussian laser pulses suggest, that the methods for propagating the laser pulses and for translating them from Lasy to PIConGPU work. Now they can be applied to the ultrafast flying focus laser.

5.1 Testing the radial group delay

To test the functionality of the radial group delay echelon (RGD) implemetation described in section 2.4.2, a laser pulse was reflected by the RGD and then propagated to different distances. Figure 5.1 shows, that the shape of the pulse barely changes after the echelon, which is what should happen. There are some artifacts from the simulation but they appear to be minor. Figure 5.2 shows the shape of the pulse after the RGD after propagation for different distances. It is clearly visible, that for the most part the shape does not change after the RGD. Only at $r = 0$ and at the edge there are some differences. The center is an effect of the artifact in the center visible in figure 5.1 on the right. Because the simulation ran on a radial grid, it always generates artifacts at $r = 0$. At the edge the intensity has dropped so much, that noise from the simulation begins to take effect. In conclusion, the RGD works as intended.

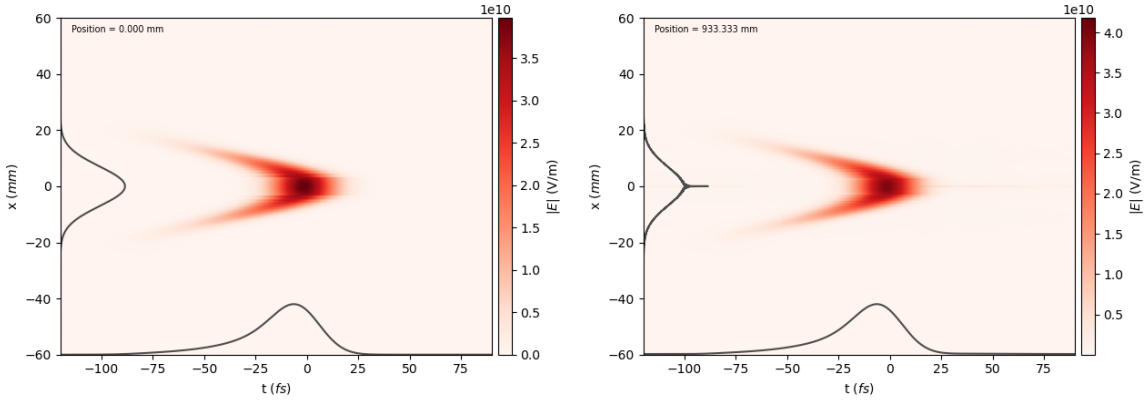

Figure 5.1: A Gaussian laser pulse in the near field of the radial group delay echelon (left) and propagated 93 cm (right), simulated in Lasy.

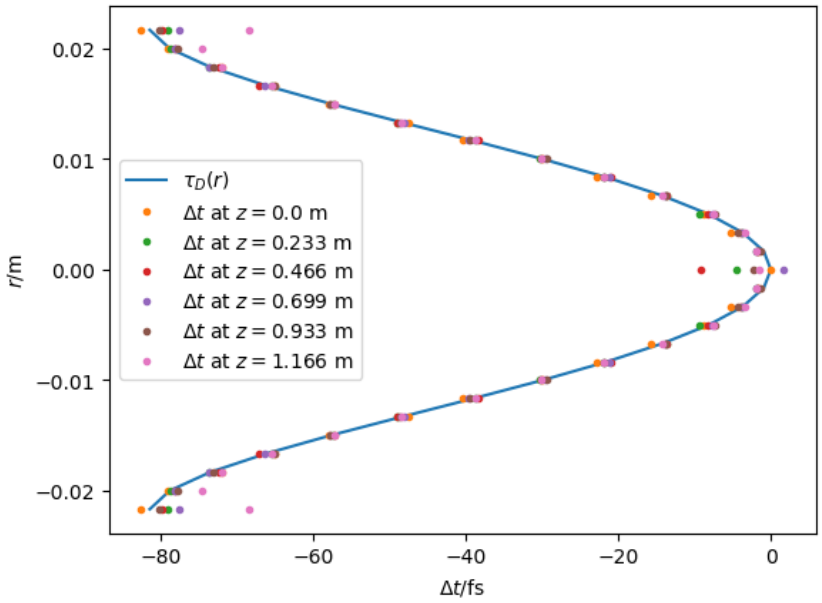

Figure 5.2: The shape of a Gaussian laser pulse after the RGD, compared to the function $\tau_D(r)$, simulated in Lasy.

5.2 Testing the axiparabola

Lasy offers an axiparabola object natively. One can just apply one to a laser pulse like this:

```
axiparabola = lasy.optical_elements.Axiparabola(f0, delta, R)
laser.apply_optics(axiparabola)
```

Figure 5.3 shows the laser pulse after the axiparabola is applied as well as after being propagated to the beginning of the focus. The full electric field at the focus is shown in figure 5.4. Because the original pulse in this case was not a Gaussian pulse but a super-Gaussian pulse of order 6 transversally, the shape at the focus is not a gaussian pulse either. It is instead close to a sinc function, as one would expect.

Figure 5.6 shows the results of a simulation where the laser propagated past the focus point through the focal region. They are compared to theoretical calculations according to equations (1.4) and (1.5). The module `axiparabola_theory` (see section 2.4.3) was used to calulate the values to compare to. As figure 5.5 shows, Ambat et al were able to see the behavior described by the equations (1.4) (see image

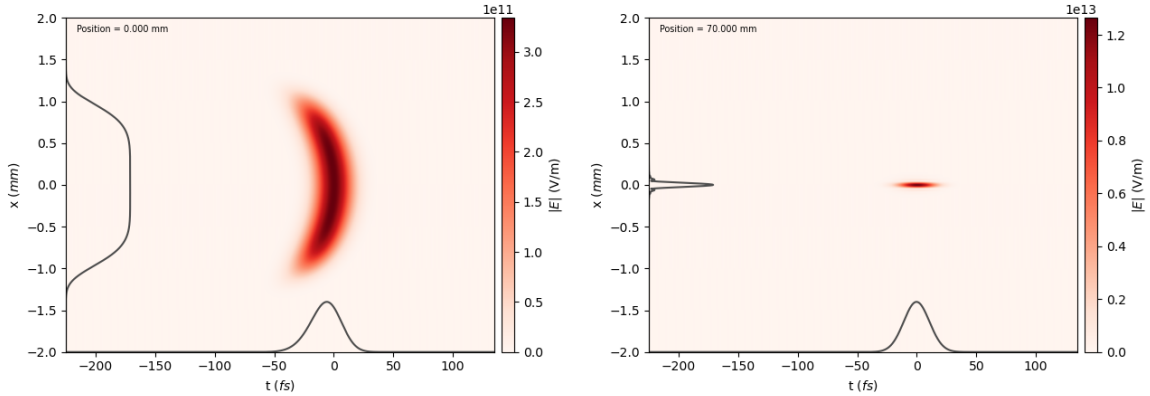

Figure 5.3: The laser pulse in the near field of the axiparabola (left) and in the far field at the focus(right), simulated in Lasy.

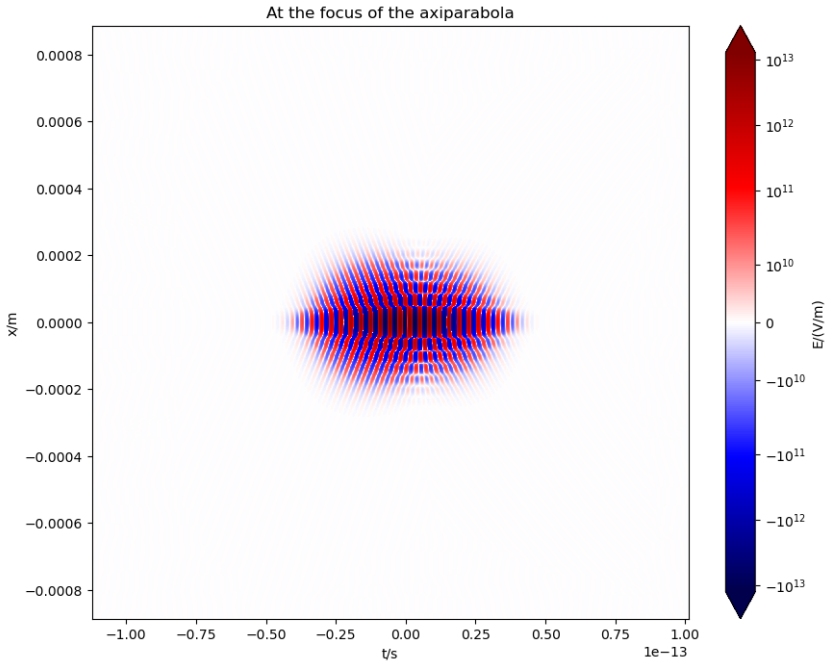

Figure 5.4: The full electric field of the pulse at the focus of the axiparabola according to a Lasy simulation.

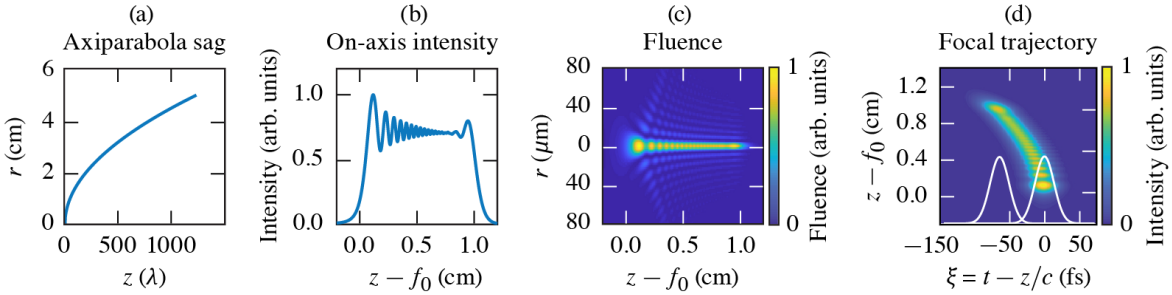

Figure 5.5: Graphic taken from Ambat et al [4] showing their axiparabola simulation. (a) the shape of the axiparabola, the so called sag function. (b) The on-axis intensity of the laser pulse as the focus moves along δ. (c) A plot showing the beam waist w of the pulse as it travels the focal region. It roughly follows equation (1.4). (d) A plot showing , that the arrival time of the pulse follows equation (1.5).

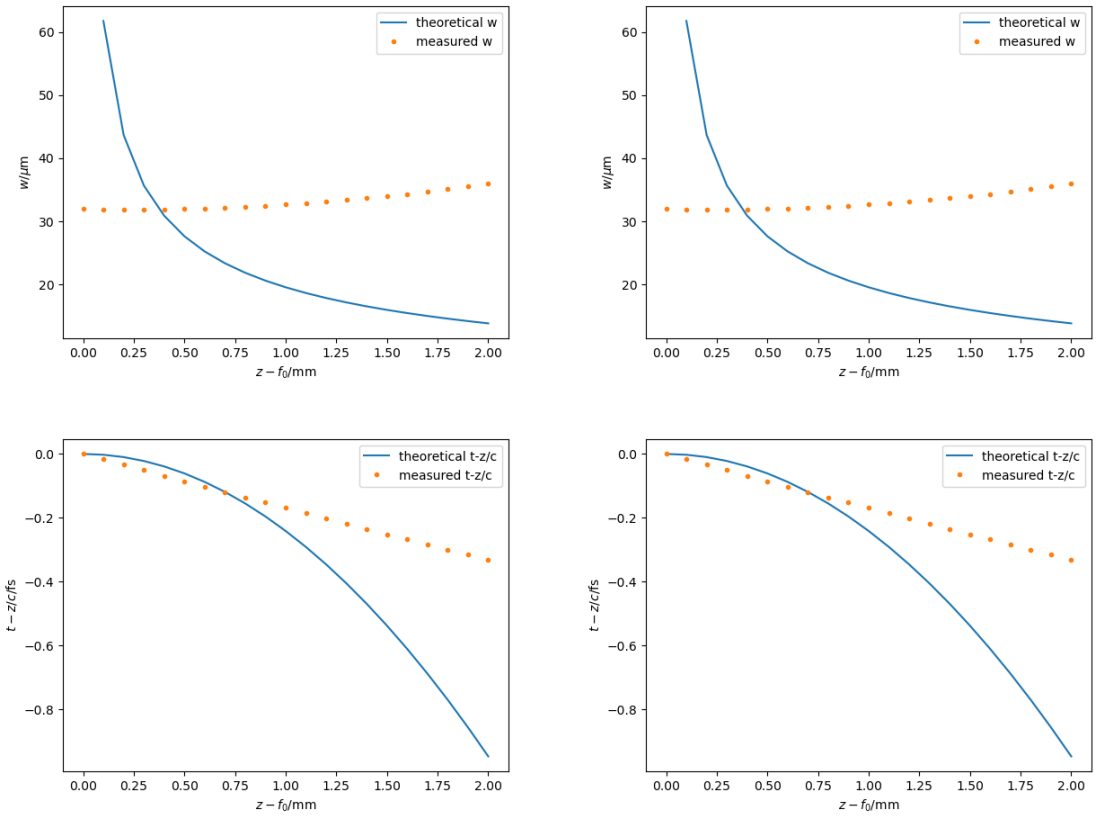

Figure 5.6: The beam waist w (top) and the relative time shift $\Delta t = t - \frac{z}{c}$ (bottom) over the focal region δ of the axiparabola according to a Lasy simulation. The left side shows a propagation from one point to another and the right shows a direct propagation from the axiparabola to each point.

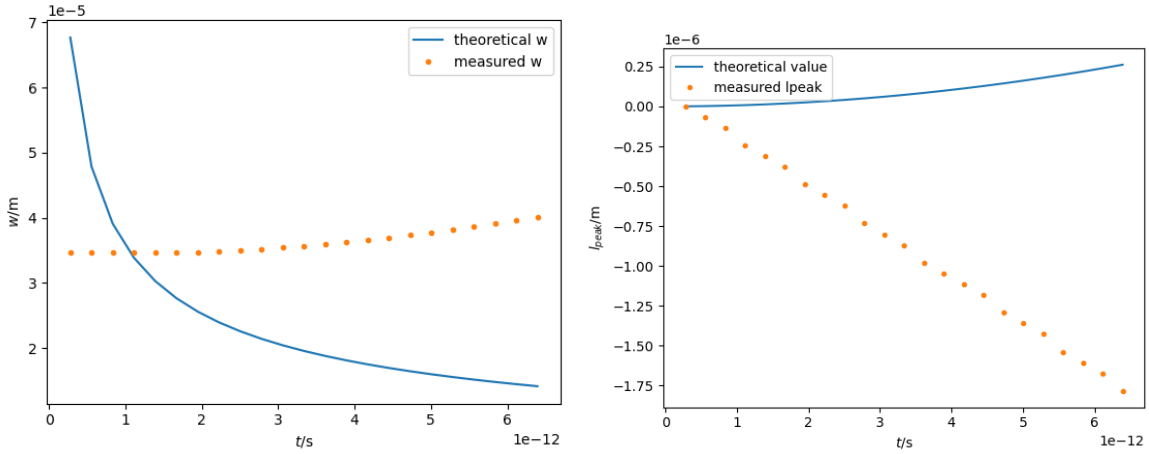

Figure 5.7: The beam waist w (left) and peak position $z - ct$ (right) of the axiparabola laser according to the PIConGPU simulation.

(c)) and (1.5) (see image (d)). This is clearly not what can be seen in the results here in figure 5.6. The naive way of doing such a simulation in lasy is to just propagate the laser to a point, measure w and Δt and then propagate it further from there. The results of that are shown on the left of figure 5.6. Because measurement and theory clearly do not align here, I tried to propagate the laser pulse to each point seperately from the axiparabola to avoid compounding errors. The results of that test are shown on the right of the figure and are clearly in line with the plots on the left. These possible

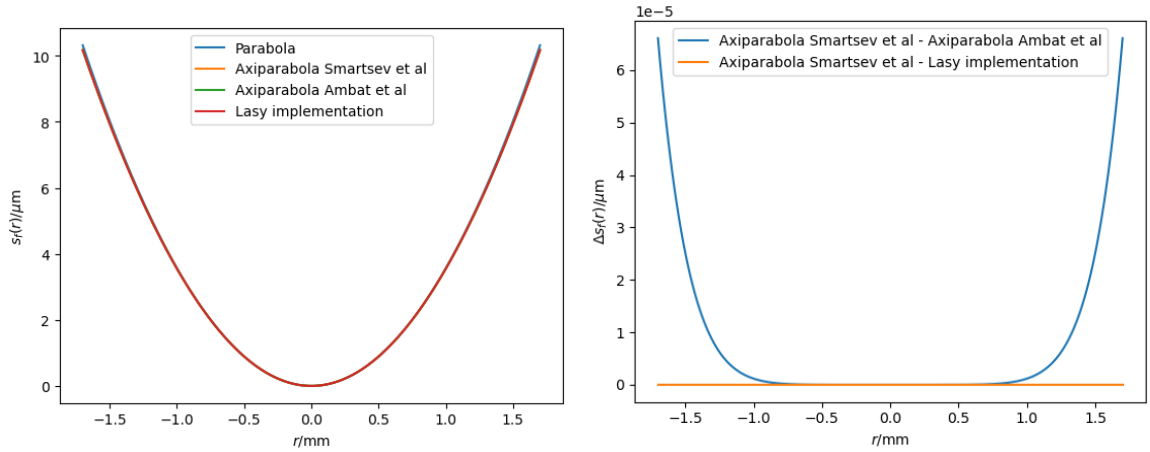

Figure 5.8: The shape of the axiparabola. Left: Smartsev [16] and Ambat [4] compared with the Lasy implementation [1] and a regular parabolic mirror. Right: A relative comparison of the same sag functions $s_f(r)$ for the axiparabola.

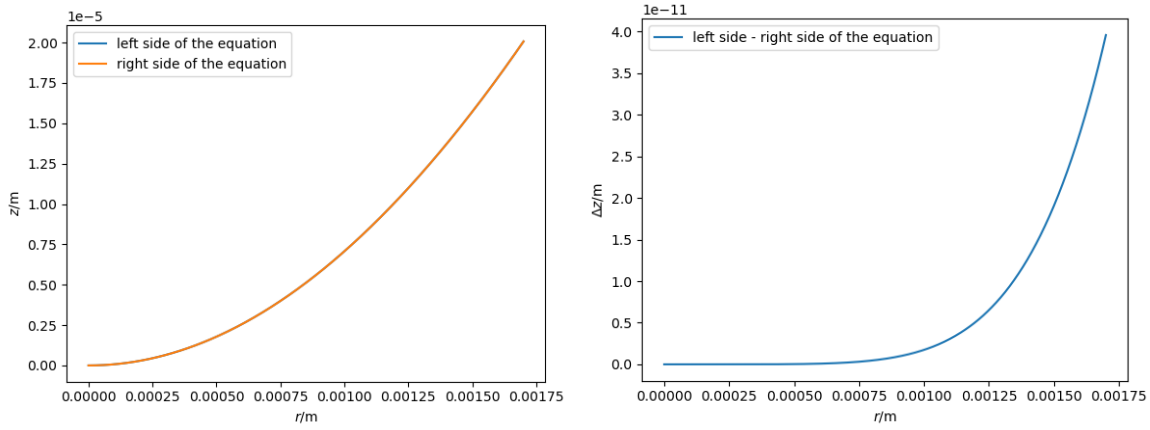

Figure 5.9: A numerical comparison of the left and right sides of equation (5.1) for the shape of the axiparabola, using the sag function $s_f(r)$ from Ambat et al [4]. Left: Both sides as absolute values. Right: The difference between the two sides.

compounding errors of the propagation can therefore be excluded as the main source of the difference between measurement and theory.

In order to exclude another possible problem, the same propagation simulation was also run in PIConGPU. The results are shown in figure 5.7. The measured values are calculated from the field of the pulse statistically for this figure but other methods give similar results. The simulation methods are extremely different from one another: Lasy works in fourier space using angular spectrum propagation, PIConGPU simply integrates the Maxwell equations. Still, the results are of similar nature: The beam waist goes up instead of down and the focus does not move as expected either. The propagation method within the focal region is clearly not the problem here.

Because both simulation methods have already been tested with the known Gauss-laser in sections 3 for the lasy propagation and 4 for the PIConGPU propagation, the methods to propagate to the focus and import the pulse to PIConGPU are unlikely to be the problem here.

The remaining options for causing the differences between theory and measurement are the axiparabola itself or the theory being incorrect. For this reason figure 5.8 shows a comparison of the sag functions described in the different papers, the one used in Lasy and a normal parabolic mirror. They are all very similar. This makes it unlikely, that the shape of the axiparabola is the problem. Nonetheless, it

is interesting to see, whether this shape fulfills the differential equation, that Smartsev et al [16] and Oubrerie et al [14] used to derive the shape:

$$s_f(r) - f(r) + \sqrt{(f(r) - s_f(r))^2 + r^2} = r \cdot \frac{\mathrm{d}s_f(r)}{\mathrm{d}r} \tag{5.1}$$

As can be seen in figure 5.9, the two sides are almost identical. Therefore, the shape is likely to be correct.

5.3 The complete assembly

Combining the two elements and applying them to the laser, one should get a flying focus laser. Figure 5.10 shows, what this will look like for a desired focal velocity $v_f = 0.98c$ or $v_f = 1.02c$. The field is now shaped quite different to what it was before, as can be seen in figure 5.11. It is no longer symmetrical in the propagation direction. However, the two plots look like mirror images of each other. This makes sense, as the peak of the pulse is supposed to be moving in opposite directions relative to the speed of light. Because the axiparabola cuts the pulse off at a specific radius R and the RGD introduces sudden jumps as well, the defraction artefacts are more pronounced than before but they should be mostly physical.
The figures 5.12 and 5.13 show the results of lasy simulations with the flying focus assembly. Like in figure 5.6 a consecutive and a pointwise separate simulation are shown. Of course theoretical formula for the beam waist, equation (1.4), does not neccessarily apply for this setup. First of all, the beam waist is appearently larger in this case, according to the simulation. The curve is shown only for the lack of a better formula. However, the other diagram is of major importance here. The flying focus assembly is meant to make the focus point move at vacuum speed of light in the plasma of an LWFA chamber. For that the focus time must move like the theoretical lines in the bottom plots of the two figures. This is clearly not the case. The RGD did not manage to fix the problem with the axiparabola in the flying focus case.

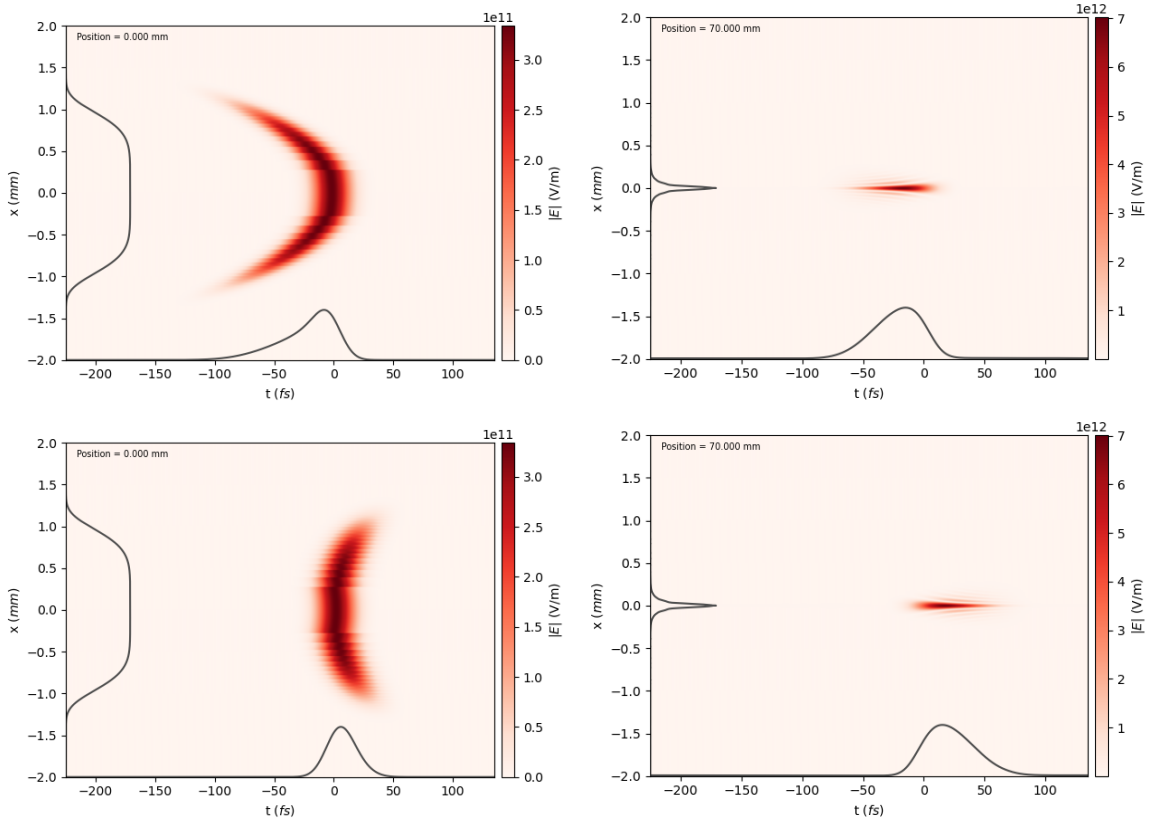

Figure 5.10: The laser pulse in the near field of the combined elements (left) and in the far field at the focus (right); designed for $v_f = 1.02c$ (top) and for $v_f = 0.98c$ (bottom); simulated in Lasy.

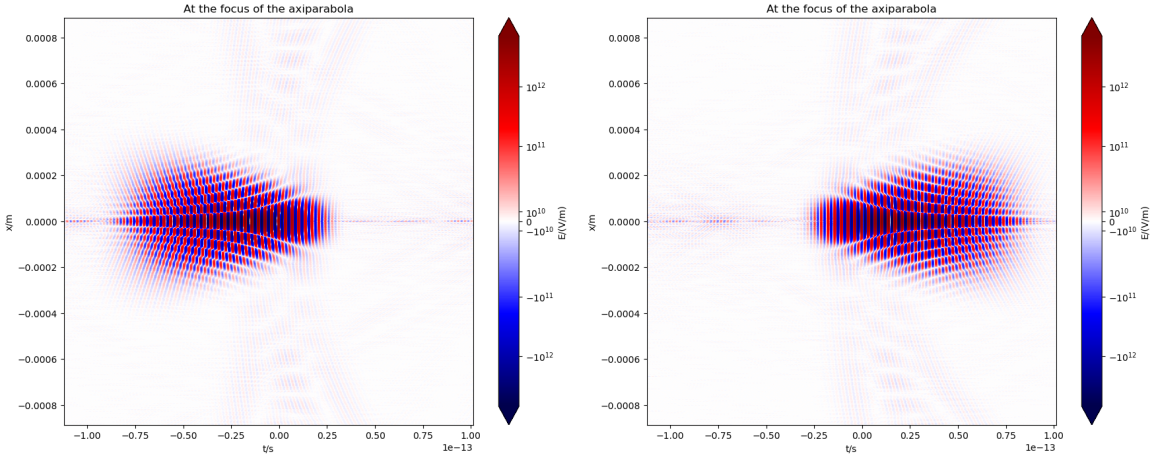

Figure 5.11: The electric field at the focus of the echelon axiparabola assembly, designed for $v_f = 1.02c$ (left) and for $v_f = 0.98c$ (right), simulated in Lasy.

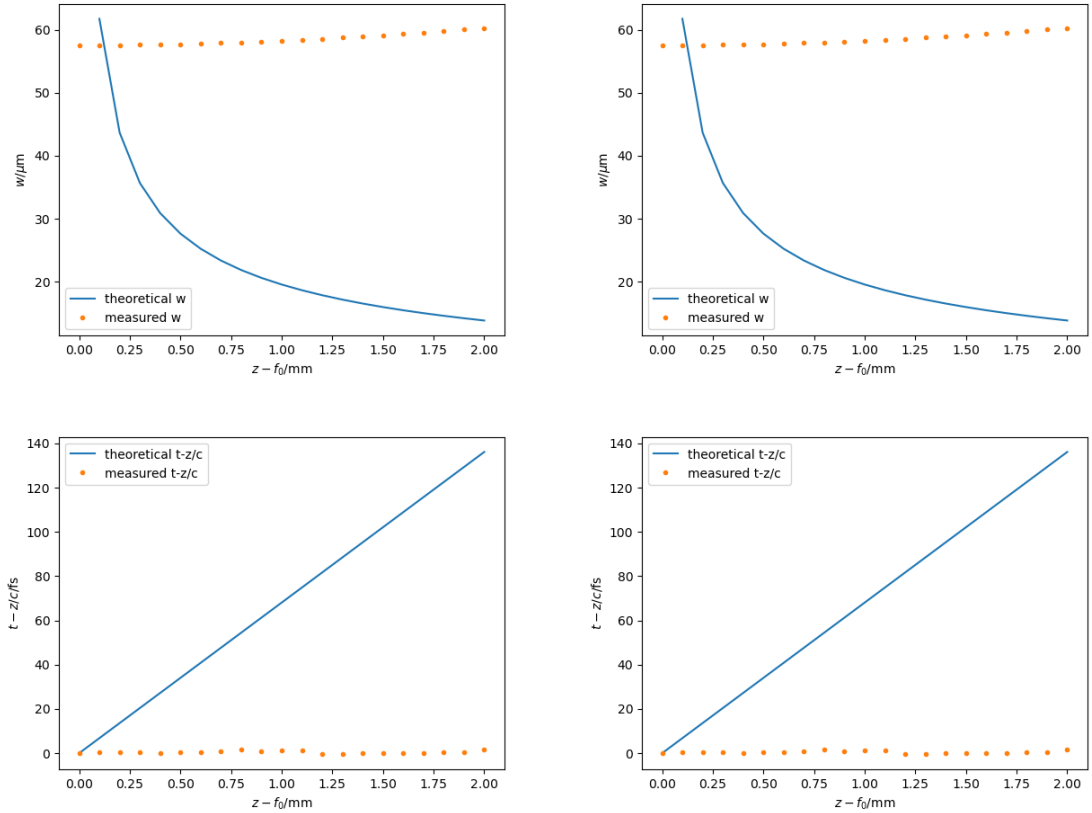

Figure 5.12: The beam waist w (top) and the relative time shift $\Delta t = t - \frac{z}{c}$ (bottom) over the focal region δ of the flying focus assembly for $v_f = 0.98c$ according to a Lasy simulation. The left side shows a propagation from one point to another and the right shows a direct propagation from the axiparabola to each point.

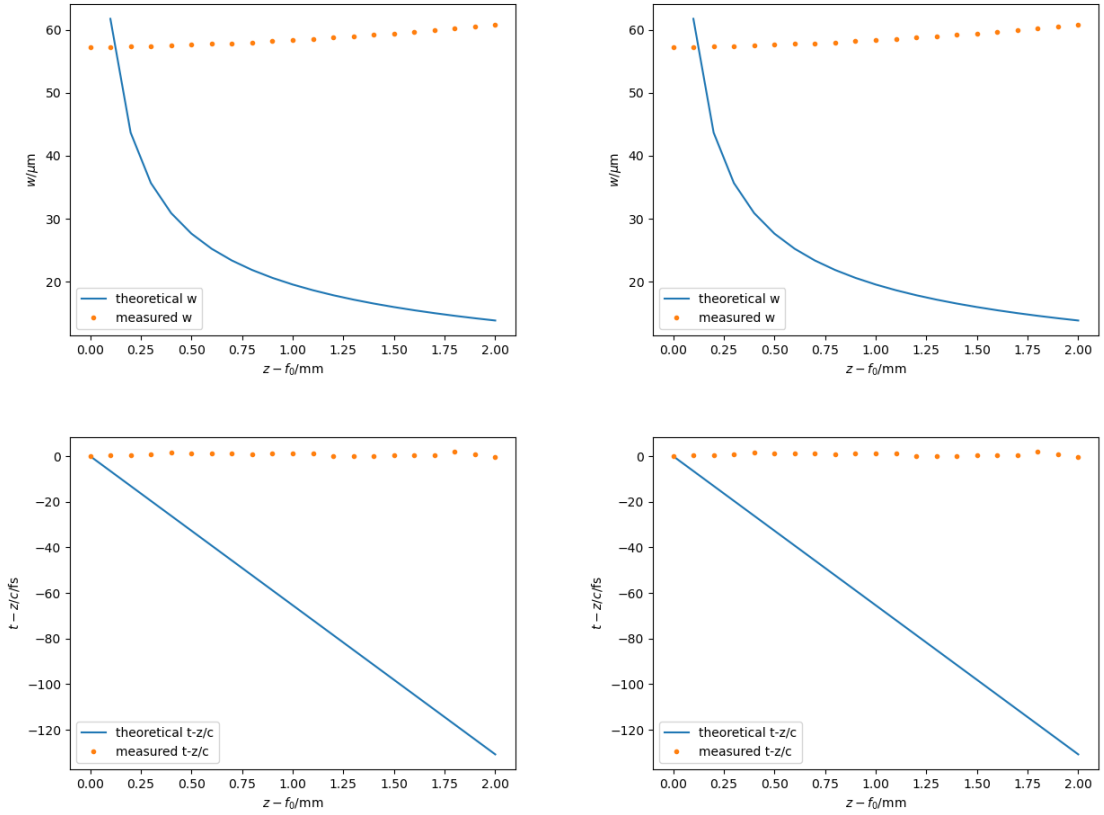

Figure 5.13: The beam waist w (top) and the relative time shift $\Delta t = t - \frac{z}{c}$ (bottom) over the focal region δ of the flying focus assembly for $v_f = 1.02c$ according to a Lasy simulation. The left side shows a propagation from one point to another and the right shows a direct propagation from the axiparabola to each point.

Chapter 6

Summary and Outlook

6.1 Summary

In conclusion, I was able to use Lasy and PIConGPU in conjunction to simulate laser pulses by first generating the pulse in Lasy, potentially sending it through optical elements and propagating it for some distances, save it to an openPMD compatible file that PIConGPU can read and finally simulate the laser from the file in PIConGPU. The previous chapters have shown, that this will correctly portray simple gauss pulses and that the translation from one simulation method to the other does not majorly change the outcome. I also managed to implement a working RGD echelon optical element that could be patched into the lasy library.
However, it was not possible to replicate the findings in other papers (Ambat et al [4] etc., see section 1.3) regarding the axiparabola and its far field. Therefore it was also not possible to generate a flying focus laser as it has been described.

6.2 Outlook

In the future people will be able to use this new framework to use more complicated laser setups for their PIConGPU simulations. It will be possible to simulate the laser pulse move through an optical setup first and then simulate it interacting with a plasma in PIConGPU.
Some work remains to find the reason(s) for the differences between the simulations shown in this work and the findings of other groups. It could, for example, be useful to test different propagation methods in the Lasy simulations. Lasy offers a variety of options, that are currently in developement. In chapter 3 is shown, that the standard propagator works for simple Gaussian pulses but it could be that the pulse here is to complicated for it. However, at their current implementation at the time of writing these alternative propagators dont support lasers in cylindrical coordinates and only cartesian coordinates are usable. This has the disavantage that it is a lot more memeory intesive, meaning it allows for fewer points transversally and longitudinally than previous simulations described in this work. This is why such tests are not included in this work. It might also be worthwhile to further investigate the axiparabola theory (see section 1.3). Mistakes here could also explain the problem.
It would also be interesting to further analyse the the artefacts visible in figures 2.4, 4.1 and 4.2. For example: how much energy do they contain? How can they move faster than the speed of light?

```
(*) There's more to explore here.
```

6.3 Code availability

All the code used in this work that I have written and mentioned here can be found on github at [12]. Most images were created in the jupyter notebooks in that repository, using the modules mentioned in section 2.4.

Bibliography

[1] Lasy 0.6.2 documentation. `https://lasydoc.readthedocs.io/en/latest`. Accessed october 2025.

[2] openpmd 0.16.1 documentation. `https://openpmd-api.readthedocs.io/en/0.16.1`. Accessed october 2025.

[3] Picongpu 0.7.0 documentation. `https://picongpu.readthedocs.io/en/0.7.0`. Accessed october 2025.

[4] M. V. Ambat, J. L. Shaw, J. J. Pigeon, K. G. Miller, T. T. Simpson, D. H. Froula, and J. P. Palastro. Programmable-trajectory ultrafast flying focus pulses. *Optics Express*, 31(19), 2023.

[5] M. Bussmann, H. Burau, T. E. Cowan, A. Debus, A. Huebl, G. Juckeland, T. Kluge, W. E. Nagel, R. Pausch, F. Schmitt, U. Schramm, J. Schuchart, and R. Widera. Radiative signatures of the relativistic kelvin-helmholtz instability. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 5:1–5:12, New York, NY, USA, 2013. ACM.

[6] J. P. Couperus, R. Pausch, A. Köhler, O. Zarini, J. M. Krämer, M. Garten, A. Huebl, R. Gebhardt, U. Helbig, S. Bock, K. Zeil, A. Debus, M. Bussmann, U. Schramm, and A. Irman. Demonstration of a beam loaded nanocoulomb-class laser wakefield accelerator. *Nature Communications*, 8, 2017.

[7] Alexander Debus, Richard Pausch, Axel Huebl, Klaus Steiniger, René Widera, Thomas E. Cowan, Ulrich Schramm, and Michael Bussmann. Circumventing the dephasing and depletion limits of laser-wakefield acceleration. *Physical Review X*, 9, 2019.

[8] Fabia Dietrich. Implementation and validation of a method for using experimentally measured laser profiles in laser-plasma simulations with picongpu. Master's thesis, HZDR, TU Dresden, 2024.

[9] A Hannasch, AL Garcia, M LaBerge, R Zgadzaj, A Köhler, JPC Cabadag, O Zarini, T Kurz, A Ferrari, M Molodtsova, L Naumann, TE Cowan, U Schramm, A Irman, and MC Downer. Compact spectroscopy of kev to mev x-rays from a laser wakefield accelerator. *Scientific reports*, 11, 2021.

[10] Axel Huebl, Rémi Lehe, Jean-Luc Vay, David P. Grote, Ivo Sbalzarini, Stephan Kuschel, David Sagan, Christopher Mayes, Frédéric Pérez, Fabian Koller, Franz Poeschel, Carsten Fortmann-Grote, Angel Ferran Pousa, Juncheng E, Maxence Thévenet, and Michael Bussmann. openPMD: A meta data standard for particle and mesh based data. https://github.com/openPMD, 2015.

[11] A Irman, JP Couperus, A Debus, A Köhler, JM Krämer, R Pausch, O Zarini, and U Schramm. Improved performance of laser wakefield acceleration by tailored self-truncated ionization injection. *Plasma Physics and Controlled Fusion*, 60, 2018.

[12] Edgar Marquardt. lasy_incorporation. `https://github.com/Ehtyar/lasy_incorporation`. Repository containing this work and all code written for it.

[13] Kyle G. Miller, Jacob R. Pierce, Manfred V. Ambat, Jessica L. Shaw, Kale Weichman, Warren B. Mori, Dustin H. Froula, and John P. Palastro. Dephasingless laser wakefield acceleration in the bubble regime. *Sci. Rep.*, 13, 2023.

[14] Kosta Oubrerie, Igor A Andriyash, Ronan Lahaye, Slava Smartsev, Victor Malka, and Cédric Thaury. Axiparabola: a new tool for high-intensity optics. *J. Opt.*, 24, 2022.

[15] J. P. Palastro, J. L. Shaw, P. Franke, D. Ramsey, T. T. Simpson, and D. H. Froula. Dephasingless laser wakefield acceleration. *Phys. Rew. Letters*, 124, 2020.

[16] Slava Smartsev, Clement Caizergues, Kosta Oubrerie, Julien Gautier, Jean-Philippe Goddet, Amar Tafzi, Kim Ta Phuoc, Victor Malka, and Cedric Thaury. Axiparabola: a long-focal-depth, high-resolution mirror for broadband high-intensity lasers. *Optics Letters*, 44, 2019.

[17] Klaus Steiniger. Issue 5269 todo in laser refactoring. `https://github.com/ComputationalRadiationPhysics/picongpu/issues/5269`. Accessed october 2025.

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

List of Figures

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Kern- und Teilchenphysik und am Institut für Strahlenphysik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Edgar Marquardt
Dresden, December 17, 2025