CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF NUCLEAR SCIENCES AND PHYSICAL ENGINEERING

Department: Department of Software Engineering
Study programme: Applications of Informatics in Natural Science

Modelling laser absorption using machine learning methods

MASTER THESIS

Author: Bc. Samuel Šitina
Supervisor: doc. Ing. Ondřej Klimo, Ph.D.
Year: 2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šitina**   Jméno: **Samuel**   Osobní číslo: **519950**

Fakulta/ústav: **Fakulta jaderná a fyzikálně inženýrská**

Zadávající katedra/ústav: **Katedra softwarového inženýrství**

Studijní program: **Aplikace informatiky v přírodních vědách**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Modelování laserové absorpce pomocí metod strojového učení

Název diplomové práce anglicky:

Modeling Laser Absorption Using Machine Learning Methods

Pokyny pro vypracování:

1. Vytvořte matematický model absorpce laserového záření v terči, který bude popisovat teplotu a množství laserem urychlených horkých elektronů, pomocí metod strojového učení. Za účelem zpřesnění modelu navrhněte a implementujte strategii doplnění nových dat z Particle-in-Cell simulací laserové absorpce tak, aby se s relativně malým objemem dat omezila neurčitost vytvořeného modelu.
2. Navrhněte a implementujte metodu, která určí neurčitost resp. přesnost automatického zpracování výsledků Particle-in-Cell simulací, z nichž jsou čerpána data pro model strojového učení.
3. K vytvořenému modelu navrhněte a implementujte uživatelské prostředí a vytvořte podrobnou dokumentaci.
4. Vytvořený model porovnejte s modely nebo daty dostupnými ve vědecké literatuře.

Seznam doporučené literatury:

[1] C. M. Bishop, Pattern Recognition and Machine Learning. New York, Springer, 2006.
[2] R. Garnett, Bayesian Optimization. Cambridge University Press, 2023.
[3] A. Döpp, C. Eberle, S. Howard, F. Irshad, J. Lin and M. Streeter, Data-driven science and machine learning methods in laser–plasma physics, High Power Laser Science and Engineering, Vol. 11, e55 (2023).

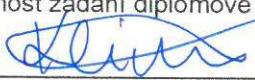Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Ondřej Klimo, Ph.D.   katedra fyzikální elektroniky   FJFI
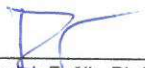
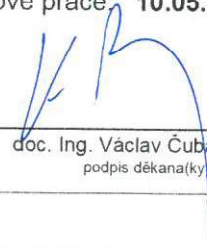Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **18.10.2023**   Termín odevzdání diplomové práce: **10.05.2024**

Platnost zadání diplomové práce: **17.10.2025**

_____
doc. Ing. Ondřej Klimo, Ph.D.
podpis vedoucí(ho) práce

_____
doc. Ing. Radek Fučík, Ph.D.
podpis vedoucí(ho) ústavu/katedry

_____
doc. Ing. Václav Čuba, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

3.11. 2023
_____
Datum převzetí zadání

_____
Podpis studenta

Statement of originality

Hereby I declare that this thesis is my original authorial work, which I have worked out on my own with the guidance of my supervisor. All sources, references, and literature used or excerpted during the elaboration of this work are properly cited and listed in complete reference to the due soruce.

In Prague on ....

Bc. Samuel Šitina

Acknowledgment

I would like to thank my supervisor Doc. Ing. Ondřej Klimo, Ph.D. for valuable guidance throughout the entire process. I would like to thank my parents who have supported unconditionally me in unimaginable ways. I would also like to thank my sister and my brother, whose advice helped me find motivation in the times of struggle.

Huge thanks goes to my friends (yes, even those of you who think plasma is not real), because without you life would be boring and it would be hard to find meaning in anything.

<div align="right">Bc. Samuel Šitina</div>

Title: Modelling laser absorption using machine learning methods

Author: Bc. Samuel Šitina

Study programme: Applications of Informatics in Natural Science

Type of thesis: Master thesis

Supervisor: doc. Ing. Ondřej Klimo, Ph.D. Department of Laser Physics and Photonics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague

Abstract: Popis práce anglicky

Key words: Klíčová slova

Title:

Modelling laser absorption using machine learning methods

Author: Bc. Samuel Šitina

Abstract: Popis práce česky

Key words: Key words

Contents

Introduction

Plasma, the fourth state of matter, makes up more than 99% of the visible universe [13]. It has been observed in technology since 1810 in various cases of electrical discharge, but a deeper study of plasma has begun in the 1950s in the context of a controlled nuclear fusion. Later, with the development of high-powered lasers, it became possible to create a plasma with densities similar to those of solids.

Advancements in laser technology have enabled us to study plasma under increasingly extreme conditions. In parallel, computer simulations offer a complementary method, allowing us to explore specific effects from fundamental principles. However, for some applications, surrogate models may be more appropriate. These models focus solely on describing the relationship between inputs and outputs, without incorporating the underlying physical principles.

In this thesis, we create a surrogate model for laser absorption in high-intensity short-pulse laser-plasma interactions. We explore a segment of the parameter space using several hundred particle-in-cell simulations and model the hot electron temperature based on these parameters. This multi-step process is presented in detail throughout the thesis.

In the first chapter, we introduce the fundamental physical effects relevant to the studied problem, providing the necessary background to understand the data obtained from simulations. This chapter also covers the motivation behind this work and discusses key aspects of particle-in-cell simulations.

Chapter 2 focuses on retrieving the hot electron temperature from the simulation results. We aim for an unsupervised process, which facilitates the addition of more simulations in the future by automating the procedure. Various fitting methods are reviewed for estimating the parameters without supervision and one explicit method is explained.

In Chapter 3, we discuss three potential surrogate models suitable for this thesis. Although we do not implement these models ourselves, we explain the underlying principles on which they are based.

The fourth chapter is dedicated to implementing the fitting method introduced in Chapter 2. We address the challenges encountered when processing large amounts of simulation data, examine the strengths and weaknesses of the proposed fitting method, and compare the results to what we consider reliable estimates of fit parameters.

Chapter 5, the final chapter, centers on applying the models discussed in Chapter 3. We compare the performance of these models on the dataset analyzed in Chapter 4, present a tool for visually inspecting the models, and propose a strategy for expanding the dataset by running additional simulations.

This comprehensive approach aims to create a robust surrogate model for better understanding laser-plasma interactions in certain part of the parameter space with possible application in optimizing the X-ray photon yields. We try to leverage both simulation data and advanced modelling techniques.

Chapter 1

Plasma

A plasma is a quasi-neutral gas of charged and neutral particles which exhibits collective behaviour [9]. In simple terms, quasi-neutrality means that the density of electrons n_e and density of positively charged ions n_i locally satisfy:

$$n_e \simeq Z n_i \tag{1.1}$$

where Ze is the charge of one positively charged ion and e is elementary charge [13].

The non-neutral particles in plasma are subject to electric and magnetic fields generated either by external sources or by neighbouring particles. The long-range nature of Coulomb potential ensures that macroscopic fields dominate over forces created by microscopic fluctuations [13]. To explain the collective behaviour properly, one can start by writing *Vlasov equation* [21]:

$$\frac{\partial f_j}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f_j}{\partial \boldsymbol{x}} + \frac{q_j}{m_j} \left(\boldsymbol{E} + \frac{\boldsymbol{v} \times \boldsymbol{B}}{c} \right) \cdot \frac{\partial f_j}{\partial \boldsymbol{v}} = 0 \tag{1.2}$$

where $f_j = f_j(\boldsymbol{x}, \boldsymbol{v}, t)$ is the phase space distribution function, which characterizes the location of the particles of species j (electrons or ions) in phase space $(\boldsymbol{x}, \boldsymbol{v})$ (position, velocity) as a function of time. q_j and m_j are charge and mass of the species j and c is the speed of light [21].

After calculating the 0th and 1st moment of Vlasov equation (averaging through $\boldsymbol{v}$), we obtain the equation of continuity and force equations for the density $n_j = \int f_j(\boldsymbol{x}, \boldsymbol{v}, t) \, d\boldsymbol{v}$ and mean velocity $\boldsymbol{u}_j$ defined by $n_j \boldsymbol{u}_j = \int \boldsymbol{v} f_j(\boldsymbol{x}, \boldsymbol{v}, t) \, d\boldsymbol{v}$:

$$\frac{\partial n_j}{\partial t} + \nabla \cdot (n_j \boldsymbol{u}_j) = 0 \tag{1.3}$$

$$n_j \left(\frac{\partial \boldsymbol{u}_j}{\partial t} + (\boldsymbol{u}_j \cdot \nabla) \boldsymbol{u}_j \right) = \frac{n_j q_j}{m_j} \left(\boldsymbol{E} + \frac{\boldsymbol{u}_j \times \boldsymbol{B}}{c} \right) - \frac{1}{m_j} \nabla p_j \tag{1.4}$$

where p_j is isotropic pressure and in case of negligible heat flow also the adiabatic state equation:

$$p_j n_j^{-\gamma} = \text{const.}, \tag{1.5}$$

where $\gamma = (2 + N)/N$ and N is the number of degrees of freedom. Equations 1.3, 1.4 and 1.5 together with the Maxwell equations are often referred to as *two-fluid*

model of plasma and describe wide range of plasma (collective) behaviour such as plasma waves or Debye shielding [21].

1.1   Temperature of plasma

In this thesis, we are studying so called *hot electrons* produced by the interaction of short laser pulse of high intensity with plasma. All important details of the physical phenomena will be covered in later sections, but let us now look at what is meant by *hot* and how the temperature of plasma is usually understood.

In a gas in the thermal equilibrium, particles velocities are given by Maxwellian distribution (in three dimensions) [9]:

$$f(\boldsymbol{v}) = n \left(\frac{m}{2\pi k_{\mathrm{B}}} \right)^{3/2} \exp\left(-\frac{\frac{1}{2}mv^2}{k_{\mathrm{B}}T} \right) \tag{1.6}$$

where $v = |\boldsymbol{v}|$ is magnitude of velocity, m is mass. k_{B} is the Boltzmann constant and T is temperature. The average kinetic energy E_{av} is then [9]:

$$E_{\mathrm{av}} = \frac{3}{2} k_{\mathrm{B}} T \tag{1.7}$$

Because of this relation between E_{av} and T, it is customary in plasma physics to express the temperature in the same units as energy. If $k_{\mathrm{B}}T = 1\,\mathrm{eV} = 1.6 \times 10^{-19}\,\mathrm{J}$, then [9]:

$$T = \frac{1.6 \times 10^{-19}}{1.38 \times 10^{-23}} = 11600 \tag{1.8}$$

From this it follows that the factor of the conversion is:

$$1\mathrm{eV} = 11600\,\mathrm{K} \tag{1.9}$$

The electrons and the ions can have different temperature [9]. Moreover, there can be multiple groups of electrons with different distributions, but we will describe this more deeply in one of the later sections.

1.2   Critical density

Now consider a high frequency electric field $\boldsymbol{E} = \boldsymbol{E}(\boldsymbol{x}) \exp\left(-i\omega t\right)$. The frequency ω is assumed to be greater than electron plasma frequency ω_{pe} defined as $\omega_{\mathrm{pe}}^2 = 4\pi e^2 n_{\mathrm{e}}/m_{\mathrm{e}}$ with $n_{\mathrm{e}} = Z n_{\mathrm{i}}$ being electron density and m_{e} electron mass. Maxwell equations plus the equation of motion of electron fluid give us:

$$\nabla \times \boldsymbol{B} = -\frac{i\omega}{c} \epsilon \boldsymbol{E}, \tag{1.10}$$

where $\epsilon = 1 - \omega_{\mathrm{pe}}^2/\omega^2$ defines the dielectric function of the plasma [21]. After further derivation using the other Maxwell equations and vector identities we can get:

$$\nabla^2 \boldsymbol{E} + \frac{\omega^2}{c^2} \epsilon \boldsymbol{E} + \nabla \left(\frac{1}{\epsilon} \nabla \cdot (\epsilon \boldsymbol{E}) \right) = 0 \tag{1.11}$$

Assuming space dependency described by $\exp{(i\boldsymbol{k} \cdot \boldsymbol{x})}$, the dispersion relation is then:

$$\omega^2 = \omega_{\mathrm{pe}}^2 + k^2 c^2. \tag{1.12}$$

It is possible to show, that k becomes imaginary for $\omega < \omega_{\mathrm{pe}}$. This can be interpreted the following way: electrons screen the field of a light wave if $\omega < \omega_{\mathrm{pe}}$. Because of that, $\omega_{\mathrm{pe}} = \omega$ defines the maximum plasma density to which a light wave can penetrate - *critical density*:

$$n_{\mathrm{cr}} = \frac{\omega^2 m_{\mathrm{e}}}{4\pi e^2} = 1.1 \times 10^{21}/\lambda_{\mu}^2 \, \mathrm{cm}^{-3}, \tag{1.13}$$

where λ_{μ} is the wavelength of the light in microns in vacuum [21].

The examples of plasmas of different densities and temperatures found in the real world can be seen in the table 1.1.

Table 1.1: Densities and temperatures of various plasma types [13].

Type	Electron density $n_{\mathrm{e}} \left[(\mathrm{cm})^{-3} \right]$	Electron temperature $T_{\mathrm{e}} \, [\mathrm{eV}]$
Stars	10^{26}	2×10^3
Laser fusion	10^{25}	3×10^3
Magnetic fusion	10^{15}	10^3
Laser-produced	10^{18} - 10^{24}	10^2 - 10^3
Discharges	10^{12}	1 - 10
Ionosphere	10^6	1.0
Interstellar medium	1	10^{-2}

1.3 Ultra-intense short pulse laser plasma interactions

In this thesis, we study the interaction of an ultra-intense, ultra-short laser pulse with solid plasma. An illustration of this scenario is shown in Figure 1.1. To put it simply, the most relevant physical process can be divided into three distinct parts.

In a real-world experiment, the first effect observed when the laser pulse strikes the target surface is ionization. This process frees electrons, which then facilitate the heating of the target and electrons are accelerated to high velocities. These high-energy electrons penetrate the remaining solid, unionized target, and, under the right conditions, X-ray photons are emitted.

The following sections provide a detailed explanation of these processes. The last section of this chapter provides an introduction to particle-in-cell simulations which are used in this thesis to obtain data for electron temperature modelling.
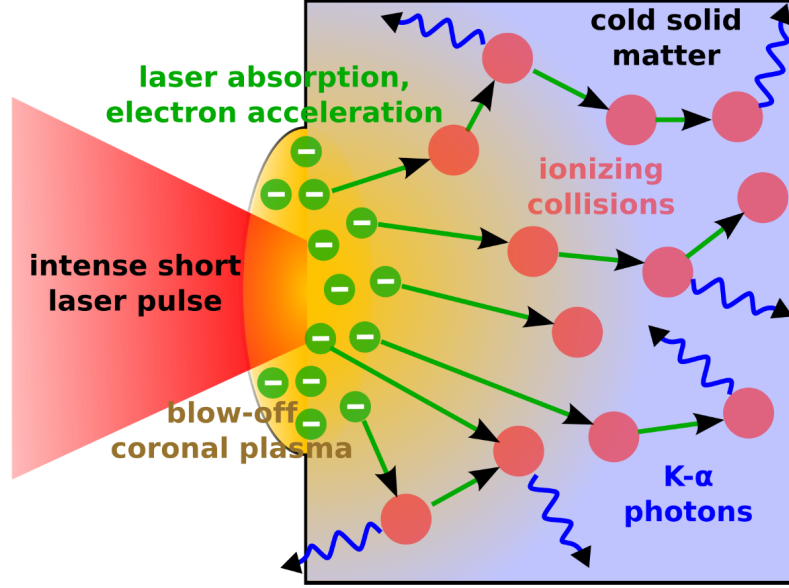
Figure 1.1: An illustration of the interaction of ultra intense laser with plasma.

1.4 Ionization

Any substance can become plasma with the sufficient increase of its temperature. The threshold can vary, but usually can be found in the order of 1 eV, because any neutral atom binds the outer electron with a binding energy in order of 1 eV [39].

Ionization mechanisms

There are several mechanisms which can be used to describe ionization. One can start with directly hitting the atoms with fast particles, but for that one would need a stream of such particles. More common way of ionization is achieved by electromagnetic radiation (photoionization) or even via electrical breakdown in strong electric fields [13]. For this thesis, the most relevant ionization is through electromagnetic radiation - in our case a laser.

Firstly, oscillating electromagnetic field makes free electrons oscillate and they can ionize other atoms via collisions. New free electrons freed by the collisions can then also hit other atoms and an avalanche if ionization events can develop.

There are also non-collisional mechanisms of ionization. Imagine field of hydrogen atom at Bohr radius a_{B} - the most probable distance of electron from the atomic nucleus:

$$a_{\mathrm{B}} = \frac{\hbar^2}{m_{\mathrm{e}}e^2} = 5.3 \times 10^{-9}\,\mathrm{cm} \qquad (1.14)$$

where $\hbar$ is the reduced Planck constant. The electric field for hydrogen E_{H} is then:

$$E_{\mathrm{H}} = \frac{e}{a_{\mathrm{B}}^2} \simeq 5.1 \times 10^9\,\mathrm{V.m^{-1}}. \qquad (1.15)$$

The corresponding so called *atomic intensity* for hydrogen I_H is [13]:

$$I_\text{H} = \frac{E_\text{H}^2}{8\pi} \simeq 3.51 \times 10^{16}\,\text{W.cm}^{-2} \qquad (1.16)$$

where c is the speed of light in vacuum.

It is reasonable to think that to ionize the hydrogen atom one needs $I_\text{L} > I_\text{H}$, where I_L is the intensity of the laser. In reality, the ionization occurs already for smaller laser intensities due to so called *multiphoton absorption* [13] and *quantum tunnelling* [39]. The first one can occur, because the electron can climb the virtual energy states one after another and it can get hit by next photon before it falls back to lower energy state [39]. The calculation of these transitions is non-trivial, because one has to solve time-dependant Schroedinger equation. The reader can find deeper analysis in [19].

The tunnelling effect is as well a consequence of the external electric field. The superposition of the electric field which binds the electron to the atom and the strong electric field of the laser results in conditions that allow the electron escape the potential well even if the electron energy is not higher than the threshold energy for instant ionization. Let $V_H(r) = -\frac{C}{r}$ Coulomb potential of hydrogen nucleus, where in CGS unit $C = e^2$. The superposition with strong external field gives us:

$$V_F = V_H(r) + eE_\text{ext}(r) \qquad (1.17)$$

Let $E_\text{ext}(r) = -10^{10} r\,\text{V/m}$. The final potential V_F together with highlighted region of tunnelling can be seen in Figure 1.2.
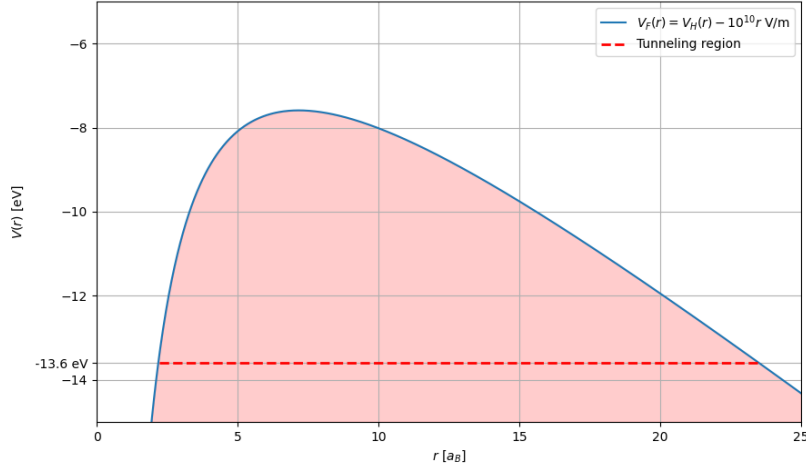

Figure 1.2: The potential of hydrogen atom modified by external field: $E_\text{ext} = -10^{10} r\,\text{V/m}$. r is shown the radial coordinate normalized to Bohr radius a_B. Energy of ground state of electron in hydrogen atom $E_0 = 13.6\,\text{eV}$ is highlighted.

The stronger is the external field, the shorter is the tunnelling distance for the electron to escape and the the higher is the probability that this can happen. The

field can even be so strong that the potential barrier will have its peak below the ground state energy. In that case, the electron is instantly considered to be free [39].

It is possible to estimate, which mechanism is more dominant cause of ionization by calculating so called *Keldysh parameter* $\gamma_K = \frac{\omega_0}{\omega_t}$, where ω_0 is the frequency of the laser and $\omega_t = \frac{eE_{ext}}{\sqrt{2m_e E_i}}$, where E_i represents the energy the electron needs to receive to be ionized [39].

The ionization processes can be explored in greater depth, but the fundamental concepts have already been adequately outlined. Henceforth, we will assume the plasma being targeted by the laser is fully ionized and will focus on how it can absorb additional energy from the laser.

1.5   Absorption of ultra-short, ultra-intense lasers

Modern lasers can generate pulses with durations of only few femtoseconds and extremely high intensities (up to 10^{22} W.cm^{-2}) [44, 46]. The interaction of such pulses with dense plasma produces hot electrons [23]. There are several processes of energy transfer from the laser's electromagnetic field to the electrons. Let us examine the most significant of these processes.

Collisional absorption

The principles of collisional absorption are similar to those of collisional ionization. In this process, an electron oscillates due to the influence of the laser field and transfers part of its kinetic energy to other ions through collisions. However, since the frequency of ion-electron collisions scales as $\nu_{ie} \propto E^{-3/2}$, this absorption mechanism is primarily significant for laser intensities below 10^{15} W.cm^{-2} [36]. Given that this thesis focuses on intensities above this threshold, further discussion on collisional absorption is unnecessary.

Resonance absorption

The first non-collisional absorption process we will describe is the *resonance absorption*. This phenomenon can occur during the propagation of a p-polarized light wave through a density gradient. By p-polarized, we refer to a wave that is linearly polarized with its polarization vector lying in the plane of incidence. The complete analytical description is difficult, but after few simplifications it is possible to obtain reasonable idea of the principle [24].

Laser light will reach density $n_t = n_{cr} \cos^2 \theta$ (from Snell's law [44]), where θ is the angle of incidence [24]. At this turning point, some light energy will tunnel through the critical density and the electron plasma will be resonantly excited at frequency of the laser ω_0. The resonant wave is then capable of accelerating electrons and is

defined by:

$$\frac{E_d}{\epsilon} = \frac{E_L}{\sqrt{2\pi\omega_0 L_n/c}}\phi\left(\tau\right) \tag{1.18}$$

where ϵ is plasma dielectric function, L_n is the density length scale and the parameter $\tau = (\omega_0 L_n/c)^{1/3}\sin\theta$ and $\phi\left(\tau\right) \propto \exp\left(-2\tau^3/3\right)$ [24, 44].

The angle of optimum resonance absorption for exponential density profile θ_{opt} can then be estimated as a function of L:

$$\theta_{opt}\left(L\right) = \arcsin\left(0.68(2\pi L)\right) \tag{1.19}$$

where L is normalized to the laser wavelength [36].

Vacuum heating

The second non-collisional absorption process (and no less important than the first one) is called *Vacuum heating* or sometimes *Brunel effect* or even *"not-so-resonant"* *resonance absorption* [7]. It was proposed by Brunel in 1987 it was later confirmed by many experiments [44].

Like before, p-polarized laser pulse is needed. At the angle of incidence θ the laser is hitting the target with steep density profile (a big gradient). A part of the pulse is reflected. The incoming laser wave $\boldsymbol{E}_{\mathrm{L}}$ and reflected wave $\boldsymbol{E}_{\mathrm{R}}$ are in superposition perpendicular to the target surface and the resulting field has a perpendicular component with amplitude $E_0 = 2E_{\mathrm{L}}\sin\theta$, under approximation that $E_{\mathrm{R}} = E_{\mathrm{L}}$. Poisson's equation at the surface gives us [44]:

$$\Delta E = -4\pi e\int_{x=-\Delta x}^{x=0} n_{\mathrm{e}}\mathrm{d}x = 4\pi e n_{\mathrm{e}}\Delta x. \tag{1.20}$$

The electrons are pulled out from plasma by the field E_0. If $n = N/\left(A\Delta x\right)$ with N/A being number of electrons pulled out into vacuum per unit area A and if $\Delta E = E_0$, we get [44]:

$$\frac{N}{A} = \frac{2E_0\sin\theta}{4\pi e}. \tag{1.21}$$

The energy absorbed E_{abs} by the electrons is then [44]:

$$E_{\mathrm{abs}} = \frac{1}{2}Nm_{\mathrm{e}}v_{\mathrm{e}}^2. \tag{1.22}$$

After calculating the power absorbed by unit area and after substituting v_{e} with *quiver velocity* v_{osc} defined by: $\frac{v_{osc}}{c} = \frac{eE_0}{m_{\mathrm{e}}c\omega_0}$, we get the absorbed fraction of the power $f_{\mathrm{VH}} = I_{\mathrm{abs}}/I_0$ [44]:

$$f_{\mathrm{VH}} = 8\frac{v_{\mathrm{osc}}}{c}sin^3\theta \tag{1.23}$$

Note that we made a simplification by letting $E_{\mathrm{R}} = E_{\mathrm{L}}$. Another option would be to directly write $E_0 = \left(1 + R^{1/2}\right)E_{\mathrm{L}}\sin\theta$, where R is the reflectivity [36]. We also neglected that the electron in vacuum is very fast and therefore relativistic

correction has to be made. It is possible to generally follow a more rigorous path found for example in [23] and [36]. Then the formula for f_{VH} is expanded to:

$$f_{\text{VH}} = \frac{\eta}{2\pi} \frac{1}{a_0} \frac{sin\theta}{cos\theta} \left(1 + R^{1/2}\right) \left\{ \left[1 + \left(1 + R^{1/2}\right)^2 a_0^2 \sin^2\theta\right]^{1/2} - 1 \right\}, \qquad (1.24)$$

where $\eta = 1.74$ and $a_0 = eE_{\text{L}}/(m_{\text{e}}\omega_0 c)$.

$\boldsymbol{J} \times \boldsymbol{B}$ heating

The last absorption mechanism we want to discuss is usually called $\boldsymbol{J} \times \boldsymbol{B}$ *heating*. In the sections above, we discussed the heating of plasma due to electron motion in the direction of the oscillating $\boldsymbol{E}$ component of the laser beam. That is of course caused by the $e\boldsymbol{E}$ part of the Lorentz force. The other part - $\boldsymbol{j} \times \boldsymbol{B}$ - can be neglected in non-relativistic cases. However, for laser intensities higher than 10^{17} W.cm^{-2} it is not possible to explain all absorption using the classical limit and another consideration has to be made [8].

Let ϕ and $\boldsymbol{A}$ be scalar and vector potential ($\boldsymbol{E} = -\nabla\phi - \frac{\partial \boldsymbol{A}}{\partial t}$ and $\boldsymbol{B} = \nabla \times \boldsymbol{A}$) satisfying Coulomb gauge $\nabla \cdot \boldsymbol{A} = 0$. Also, we can separate transverse and longitudinal part of electron momentum $\boldsymbol{p} = \boldsymbol{p}_{\text{t}} + \boldsymbol{p}_{\text{l}}$ Then the equations of motion for electron can be written as [8]:

$$\frac{\partial \boldsymbol{p}_{\text{t}}}{\partial t} = \frac{e}{c} \frac{\partial \boldsymbol{A}}{\partial t} \qquad (1.25)$$

$$\frac{\partial \boldsymbol{p}_{\text{l}}}{\partial t} = e\nabla\phi - m_{\text{e}}c^2 \nabla(\gamma - 1) \qquad (1.26)$$

where $\gamma = \sqrt{1 + \frac{a_0^2}{2}}$ is the relativistic factor for linearly polarized light [44]. a_0 is the same as defined in equation 1.24, but here is has meaning of a constant normalizing vector field $\boldsymbol{A}$. To be more precise $a_0 = |\boldsymbol{a}|$ and:

$$\boldsymbol{a} = \frac{e\boldsymbol{A}}{m_e c^2}. \qquad (1.27)$$

The second term of the equation 1.26 is the relativistic ponderomotive force and we can write the ponderomotive potential U_{p} as:

$$U_{\text{p}} = (\gamma - 1)m_{\text{e}}c^2. \qquad (1.28)$$

There is a force with frequency $2\omega_0$ which will affect the electrons in longitudinal direction. One of the interpretations of this force can sound like this: Twice every laser period, streams of electrons are pushed into the the target [8]. This causes the production of fast electrons.

It is important to note, that at higher intensity, the electron density can rise as a consequence of the zero-frequency pondermotive force. The change in density can cause the $2\omega_0$ force component to be inefficient for hot electron generation. The

density is also related the scale length of the target. For example for exponential density profile, the density decreases with the increase of scale length [8]. This means the $\boldsymbol{J} \times \boldsymbol{B}$ heating is expected to play a greater role for bigger scale lengths. In other words, the initial plasma conditions need to be known to estimate the effects of $\boldsymbol{J} \times \boldsymbol{B}$ heating.

One last note regarding $\boldsymbol{J} \times \boldsymbol{B}$ heating. The relativistic factor γ has a different form in a case of circularly polarized laser. That leads to suppressing this kind of electron heating altogether [8].

The three mentioned mechanisms are by no means exhaustive when it comes to laser absorption. They are the three most relevant in the context of this work. There are other physical processes contributing to heating up the plasma especially when the parameters of the experiment change [36]. We will now move on to describe the particular motivation of the thesis.

1.6   Motivation - X-ray K_α emission

Because hot electrons accelerated by ultra-intense ultra-short laser pulse can carry significant (keV-MeV) energy, they can penetrate deeper into the unionized part of the target, where they can generate characteristic X-rays by K-shell ionization [33]. The X-ray spectrum is consisting of spectral lines (e.g. K_α) and continuous part X-ray from Bremsstrahlung. The uniqueness of this method of producing X-rays rests in the monochromatic spectrum of high energy X-ray photons within a short pulse synchronized with the laser pulse. The source is typically very small [28].
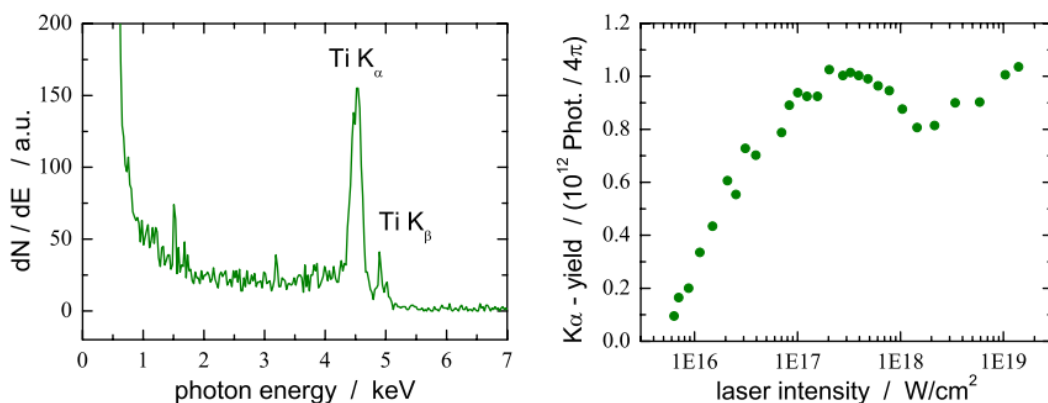

Figure 1.3: *Left:* The spectrum of laser generated K_α and K_β radiation of titanium. *Right:* The scaling of K_α - yield in relation to laser intensity. [35]

Let us now briefly look into the origin K_α-line. In figure 1.3, there is an energy spectrum of titanium x-ray photons and the K_α - yield scaling with the laser intensity. The generation of the K_α radiation clearly depends on laser intensity through the hot electrons temperature. According to [35], the yield drops at the laser intensity around 10^{18} W.cm^{-2} *"because the interaction time with the atom decreases with higher electron velocity"*. The following rise in the yield is then attributed to

the relativistic effect where the electric fields of the fast hot electrons are contracted and therefore have greater effect [35].

The total yield N can be expressed analytically as [33]:

$$N_{\mathrm{yield}} = \int n_{\mathrm{hot}} f_{\mathrm{hot}}(E) N_{\mathrm{gen}}(E) f_{\mathrm{em}}(E) \mathrm{d}E \qquad (1.29)$$

where N is the number of emitted photons, n_{hot} is the total number of hot electrons, and $f_{\mathrm{hot}}(E)$ is their energy distribution, $N_{\mathrm{gen}}(E)$ is the number of K_α photons generated by an electron of incidence energy E, and $f_{\mathrm{em}}(E)$ is the fraction of these photons that escapes from the solid [33].

Having a reliable numerical model for n_{hot} and $f_{\mathrm{hot}}(E)$ from equation 1.29 based on the parameters of the laser and the plasma could allow us to optimize the K_α yield. Namely, the angle of incidence, the laser intensity and the plasma length scale are the most important parameters for laser absorption. The research conducted by Reich et al. [33] could be followed up by examining a wider range of parameters beyond just laser intensity. This is crucial because, as shown by Cui et al. [10], the temperature function of the electrons has a complex and non-trivial shape. The complexity comes from the complex nature of the physical processes causing the electron heating.

In this thesis, we presents results from hundreds of PIC simulations while scanning through the mentioned parameters. Then we try to present and defend a way of how the hot electron temperature can be modelled and how to make the model more precise. The implications with respect to previous works and to equation 1.29 are discussed.

1.7 PIC simulations

As previously mentioned, we are using Particle-In-Cell (PIC) simulations to obtain the data necessary for our model. PIC codes have been under development since the advent of computers in the 1960s, and advancements in computer technology over the past 30 years have enabled us to run simulations on a much larger scale. One significant advantage of simulations is that they allow theoretical predictions to be verified in greater detail than is possible with real plasma experiments [11]. To illustrate the progress made over the decades, let us mention, that in 1962 Dawson and Buneman simulated the motion of 1000 plasma particles. Today, we can simulate the motion of more than 10^{10} particles [41].

We are not developing our own simulation code. Instead, we are using code freely available for academic purposes, specifically the 2D version of simulation code EPOCH [2]. EPOCH has been widely used in numerous publications within the laser-plasma field and adequately meets our requirements. Below, we will present a brief overview of the key principles of the PIC method, upon which EPOCH is also based.

Macro-particles

It is not possible to have as many particles in a simulation as in a real plasma, even in very small scales, because of the computational cost. Because of that, the simulations usually work with macro-particles which represent clouds of many real particles. These particles have finite sizes (as opposed to infinitesimal), so that there are no divergent forces in case of collisions. The forces in simulations go to zero for small distances between macro-particles. For large distances, they comply with the Coulombic behaviour. In plasma simulations, it is possible to do this without sacrificing accuracy, because of the collective behaviour of plasma[34].

Computational cycle

The simulation runs in a cycle. In each step, we solve for electromagnetic fields created by the charged particles. Then we evaluate the equations of motion for the particles, which are influenced by the Lorentz force [5]. The laser pulse is included as an external source of electromagnetic radiation at the boundary.

Typically, the finite-difference time-domain method (FDTD) is used for numerically solving Maxwell's equations, which fully describe the electromagnetic field. *Finite-difference* means that the electric field $\boldsymbol{E}$ and magnetic field $\boldsymbol{B}$ are specified in the points of a grid - usually a *Yee grid*. A comprehensive description of a Yee grid can be found in the original article by Yee [47]. The critical concept of a Yee grid is illustrated in figure 1.4. - the magnetic field components are calculated in the center of the faces of an imaginary cube (cell) while the electric field components are calculated in the center of the edges. The cube represents one cell of the 3-dimensional grid. We can approximate derivatives of electric field with centered finite differences [2]:

$$\left(\frac{\partial E_y}{\partial x}\right)_{i+\frac{1}{2},j,k} = \frac{E_{y,i+1,j,k} - E_{y,i,j,k}}{\Delta x} \tag{1.30}$$

Note, that this approximation is second order accurate at the cube point where we calculate $B_{z,i,j,k}$, because the formula is centered. Moreover, because of one of the Maxwell's equations, $\nabla \times \boldsymbol{E} = \frac{\partial \boldsymbol{B}}{\partial t}$, this derivative is exclusively used to calculate time-derivative of $B_{z,i,j,k}$. A similar relationship can be found when calculating all components of $\boldsymbol{B}$ from $\boldsymbol{E}$ and vice versa. Therefore, all used numerical derivatives are second order accurate [2].

In EPOCH, as in other PIC codes, fields are updated at both the half time-step and full time-step. The first part - time-step from n to $n+1/2$ - uses currents calculated at n:

$$\boldsymbol{E}^{n+1/2} = \boldsymbol{E}^n + \frac{\Delta t}{2}\left(c^2\nabla \times \boldsymbol{B}^n - \frac{\boldsymbol{J}^n}{\epsilon_0}\right) \tag{1.31}$$

$$\boldsymbol{B}^{n+1/2} = \boldsymbol{B}^n - \frac{\Delta t}{2}\left(\nabla \times \boldsymbol{E}^{n+1/2}\right) \tag{1.32}$$

where $\boldsymbol{J}$ is the current density and Δt is a size of a full time-step [2].
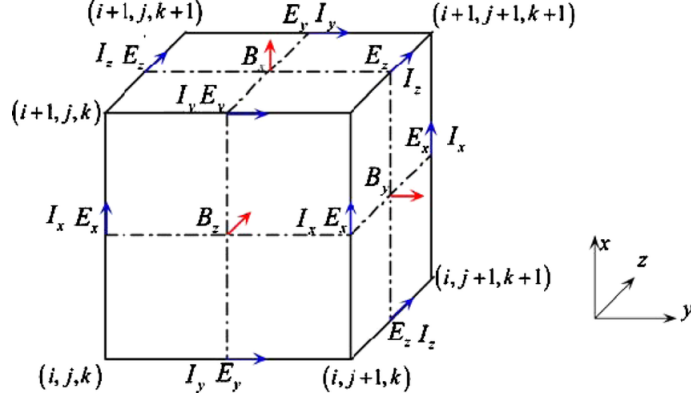
Figure 1.4: An illustration of a Yee grid [43]

.

In the second step, we use the updated currents at $n + 1$:

$$\boldsymbol{B}^{n+1} = \boldsymbol{B}^{n+1/2} - \frac{\Delta t}{2} \left(\nabla \times \boldsymbol{E}^{n+1/2} \right) \tag{1.33}$$

$$\boldsymbol{E}^{n+1} = \boldsymbol{E}^{n+1/2} + \frac{\Delta t}{2} \left(c^2 \nabla \times \boldsymbol{B}^{n+1} - \frac{\boldsymbol{J}^{n+1}}{\epsilon_0} \right) \tag{1.34}$$

The motion of particles and the resulting currents are consequences of the fields. Generally, the equations of motion are:

$$\frac{\mathrm{d}\boldsymbol{x_l}}{\mathrm{d}t} = \boldsymbol{v_l} \text{ and } \frac{\mathrm{d}\boldsymbol{p_l}}{\mathrm{d}t} = \boldsymbol{F_l} \tag{1.35}$$

where vectors $\boldsymbol{x_l}$, $\boldsymbol{v_l}$ and $\boldsymbol{p_l}$ represent the position, velocity and momentum of the l-th macro-particle. $\boldsymbol{F_l} = \boldsymbol{F_l}(t, \boldsymbol{x_l}, \boldsymbol{v_l}, \boldsymbol{E}, \boldsymbol{B})$ is the force. Fields $\boldsymbol{E}$ and $\boldsymbol{B}$ are functions of the positions and time [41]. In this context, the right-hand side of the second equation represents the Lorentz force, and the time-step formula for the momentum is: [2]:

$$\boldsymbol{p_l}^{n+1} = \boldsymbol{p_l}^n + q_l \Delta t \left[\boldsymbol{E}^{n+1/2} \left(\boldsymbol{x_l}^{n+1/2} \right) + \boldsymbol{v_l}^{n+1/2} \times \boldsymbol{B}^{n+1/2} \left(\boldsymbol{x_l}^{n+1/2} \right) \right] \tag{1.36}$$

where q_l is the charge of the l-th particle. The velocity can be calculated from the momentum using:

$$\boldsymbol{v_l} = \frac{\boldsymbol{p_l}}{\gamma_l m_l} \tag{1.37}$$

where m_l is the mass of the particle and $\gamma_l = [p_l^2/(m_l^2 c^2) + 1]^{1/2}$ is the corresponding gamma-factor [2].

The particle position update is calculated from the velocity, but this is also done in multiple steps. First, we calculate movement of half time-step from the old velocity as we need it to update the momentum in equation 1.36 [2]:

$$\boldsymbol{x_l}^{n+1/2} = \boldsymbol{x_l}^n + \frac{\Delta t}{2} \boldsymbol{v_l}^n \tag{1.38}$$

In similar way, we can then calculate $\boldsymbol{x}_l{}^{n+1}$ and $\boldsymbol{x}_l{}^{n+3/2}$ [2], which are needed for calculating the currents.

The currents necessary for updating the fields can be calculated using methods such as the one presented by Esirkepov in [12]. Modern approaches to calculating currents are generally based on solving the discrete form of the continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \boldsymbol{J} = 0 \tag{1.39}$$

where ρ is the charge density and $\boldsymbol{J}$ is the electric current. This can be discretized as:

$$\frac{\rho_{i+1/2,j+1/2,k+1/2}^{n+1} - \rho_{i+1/2,j+1/2,k+1/2}^{n}}{\Delta t} + \frac{J_{x,i+1,j+1/2,k+1/2}^{n+1/2} - J_{x,i,j+1/2,k+1/2}^{n+1/2}}{\Delta x} + \cdots = 0. \tag{1.40}$$

Since macro-particles in the simulation represent numerous real particles, a *weight* is assigned to each macro-particle. Although the exact particle distribution within a macro-particle is unknown, a representative function, known as a *shape function*, is chosen [2].

Any function with a unit integral and compact support can be used as a shape function. An even distribution of particle in volume $\Delta x \times \Delta y \times \Delta z$ is referred to as the *top hat* shape function. Using higher order shape functions are one of the improvements programmers were able to make thanks to more powerful computers. A higher order shape functions are for example triangular shape functions with volume of $2\Delta x \times 2\Delta y \times 2\Delta z$. The weight is then calculated as a convolution of shape function with the 'top hat' function [2].

Working with macro-particles instead of individual particles neglects some effects, particularly those effective over distances shorter than Δx. EPOCH uses a fully relativistic, energy-conserving binary collision model, which favors small-angle scattering to improve simulation behavior with a limited number of particles per cell. For our study involving laser intensities above 10^{10}, collisions are not necessary. Other effects, such as ionization and quantum phenomena like photon emission and pair production, are often included when relevant, with detailed descriptions available in [2].

Chapter 2

Temperature fitting

The results of the EPOCH simulations provide weighted momenta for a subset electrons $\boldsymbol{p}_{\mathrm{e}}$, where the weight represents the number of electrons with that momentum. They are not histograms yet, because more than one macro-particle can have the same value of $\boldsymbol{p}_{\mathrm{e}}$. The energies of electrons can then be calculated from $\boldsymbol{p}_{\mathrm{e}}$ using the relativistic formula:

$$E_{\mathrm{e}} = m_{\mathrm{e}} \cdot c^2 \left(\sqrt{1 + \left(\frac{p_{\mathrm{e}}}{m_{\mathrm{e}} \cdot c} \right)^2} - 1 \right), \tag{2.1}$$

where $p_{\mathrm{e}} = \sqrt{\boldsymbol{p}_{\mathrm{e}} \cdot \boldsymbol{p}_{\mathrm{e}}}$ is the size of $\boldsymbol{p}_{\mathrm{e}}$, $m_{\mathrm{e}} = 9.109 \times 10^{-31}\,\mathrm{kg}$ is the electron rest-mass and $c = 3 \times 10^8\,\mathrm{m.s^{-1}}$ is the speed of light [26].
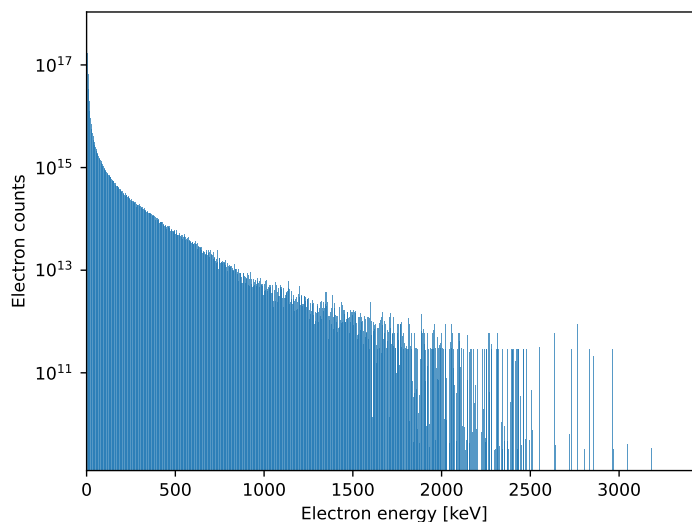
Figure 2.1: An example of electron energy distribution of 2D EPOCH simulation with intensity of laser $I = 10^{19}\,\mathrm{W.cm^{-2}}$, characteristic scale of the exponential preplasma profile $L = 0.1\,\mu\mathrm{m}$ and angle of incidence with respect to target normal direction $\alpha = 10°$.

One can easily create the histogram from electron energies and their counts. An example of such histogram can be seen in the figure 2.1. There are several things that need to be discussed.

Firstly, it is important to note that the y-axis is presented on a logarithmic scale to enhance readability. The extensive range of electron energies necessitates the use of relativistic formula. However, around the energies $E_k \approx 1900\,\text{keV}$ and more, the histogram exhibits irregularities - specifically, there are empty bins and several bins contain identical electron counts. These anomalies are attributed to the resolution of the simulation. Consequently, this portion of the spectrum is rendered questionable. This issue will be further addressed in the section dedicated to temperature fitting.

Last but not least, note the apparent exponential relationship between N_i and E_k in the energy spectrum between $E_k \approx 500\,\text{keV}$ and $E_k \approx 1500\,\text{keV}$:

$$N_i = N_0 \cdot \exp\left(-\frac{E_i}{T}\right), \tag{2.2}$$

where N_i is count in i-th bin and E_i is the corresponding energy and T is temperature in units discussed in section 1.1. The relationship is, of course, linear after the logarithmic transformation. The equation of 2.2 resembles exponential distribution with $1/T$ missing on the right-hand side. It is also not normalized, because it represents counts that have to add up to total number of electrons with that temperature. For the purpose of this thesis, we will call it the *Boltzmann distribution*, even though it is not completely accurate.

Fitting the correct part of the histogram using equation 2.2 and estimating T and N_0 is the first bigger part of this thesis. There are several obstacles, all of which are discussed in following sections.

2.1 Boltzmann vs. Maxwellian distribution

First of all, in section 1.1, where we introduced temperature as a quantity describing plasma, we said that the distribution of energies is usually considered to be Maxwellian. The question, whether we can fit our histogram using Boltzmann distribution, is therefore valid and needs to be addressed.

The equations for the Maxwellian $f_{\text{M}}(E)$ and Boltzmann $f_{\text{B}}(E)$ distributions can be after few simplifications regarding units defined as:

$$f_{\text{M}}(E)\text{d}E = \frac{\sqrt{E}}{(T)^{3/2}} \exp\left(-\frac{E}{T}\right) \text{d}E \tag{2.3}$$

and

$$f_{\text{B}}(E)\text{d}E = \frac{1}{T} \exp\left(-\frac{E}{T}\right) \text{d}E. \tag{2.4}$$

By ignoring that both are defined by the differential, taking logarithm of both $f_M(E)$ and $f_B(E)$ we get:

$$\bar{f}_M(E) = \ln\left(f_M(E)\right) = \ln\left(\frac{\sqrt{E}}{(T)^{3/2}}\right) + \left(-\frac{E}{T}\right) \tag{2.5}$$

and

$$\bar{f}_B(E) = \ln\left(f_B(E)\right) = \ln\left(\frac{1}{T}\right) + \left(-\frac{E}{T}\right). \tag{2.6}$$

These two equations describe the the distributions in a logarithmic scale. The first derivatives are then:

$$\frac{d\bar{f}_M}{dE} = -\frac{1}{2}\frac{T^{3/2}}{\sqrt{E}}\frac{1}{T^{3/2}\sqrt{E}} - \frac{1}{T} = -\frac{1}{2E} - \frac{1}{T} \tag{2.7}$$

and

$$\frac{\bar{f}_B}{dE} = -\frac{1}{T}. \tag{2.8}$$

The equations 2.7 and 2.8 suggest that, in the logarithmic scale, the slopes of these distributions differ by $-\frac{1}{2E}$, which decreases with increasing E. In other words, the fit of T can be replaced by fitting slope $s = -1/T$ of the histogram after logarithmic transformation. For large energies, the slopes of both discussed distributions are approximately equal.

Moreover, the assumption of Boltzmann distribution allows us to fit not only the temperature, but also N_0, which is the total number of electrons in that distribution of hot electrons. If we keep the form of the distribution as in equation 2.2, the total energy absorbed by the group of hot electrons E_{tothot} with temperature T_{hot} can be calculated as:

$$E_{\text{tothot}} = N_0 \cdot T_{\text{hot}}. \tag{2.9}$$

2.2 Exponential-sum fitting

Exponential-sum fitting has been a recognized challenge for many years. Forman S. Acton, a professor at Princeton University, addressed this issue in his 1970 book Numerical Methods That Work. Among various topics, he included an essay titled "What Not to Compute," wherein he described the fitting of sums of exponential functions as "extremely ill-conditioned" [1]. Despite the inherent difficulty of this problem, this thesis endeavors to determine electron temperatures, necessitating the development of an effective solution to this challenging computational task.

The task is to find the parameters $m, a_0, a_1, ..., a_m, b_1, ...b_m$ so that the function

$$f(x) = a_0 + \sum_{i=1}^{m} a_i e^{b_i x} \tag{2.10}$$

generalizes the data as good as possible. Formally, for a set of data (n data points) $((x_1, y_1), ..., (x_n, y_n))$ we are minimizing error function

$$e = \sum_{i=1}^{n} w_i (y_i - f(x_i))^2 \qquad (2.11)$$

where w_i are the weights.

Modern programming libraries provide access to well-known and generally efficient methods, such as non-linear least squares. However, due to the previously mentioned ill-conditioning, these methods do not yield satisfactory results in this context. The convergence of these methods is highly dependent on the quality of the initial guess, which poses a significant challenge. To date, significant efforts have been made to develop reliable methods for solving this problem.

Notably, Prony's method [30] and its variations, as discussed in sources such as [29], have been proposed. It was originally intended for signal approximation - specifically, the determination of complex parameters in exponential functions, similar to Fourier transformation.

Another, much simpler method is known as *successive subtraction* [45]. This method is frequently employed for exponential decays (negative parameters b_i), which closely resemble our problem of exponential distributions. The core concept involves fitting the end of the decay with a single exponential function, subtracting this result from the data, and iteratively identifying all exponential components. This simplicity makes it a promising candidate for our problem, especially since our primary interest lies in determining the temperature of the hottest electrons—the tail. However, automating this method can be challenging, as it is not straightforward to select an interval that can be fitted as a tail in each iteration. Moreover, the tail of our histograms are not reliable as was discussed with regard of the example histogram in figure 2.1.

Another method worth considering is presented in [45]. This method, called exponential sum fitting of transmissions (ESFT), was developed for fitting radiative transmission functions in the context of atmospheric research. ESFT too is an iterative method and might serve as an interesting alternative for future works.

We only mentioned a few methods. A summary with a deeper explanation of these and other methods can be found in [45], [16] or in [15].

If the goal is to fit only one of the exponentials, such as the parameters of the hot electrons distribution, the standard approach involves manually selecting the "correct" segment of the distribution and fitting it with a single exponential function. For instance, in [10], where researchers study a problem similar to ours, they appear to use this technique to determine the temperature of hot electrons. The term "correct" is somewhat vague because it can be subjective and we do not have any metric that would quantify how well the segment was chosen. We can only work with the metrics describing the fit itself.

The last mentioned option currently provides the most reliable approach to fitting the temperature of hot electrons. Even for hundreds of simulations, with the right tool, it is possible to fit manually every histogram within few hours. Having such dataset of histograms and temperatures is valuable for evaluating any method that is

trying to automate the fitting process. A tool with an UI was developed specifically for this purpose as a part of this work.

That being said, we still attempted to automate the fitting. The method we used is explicit an non-iterative.

The 'Jacquelin' method

We will call the method we chose *Jacquelin method*, because we will be referencing an article by J.Jacquelin who developed it for his own purpose [18]. This method is explicit and non-iterative which is its main advantage. The original derivation presents a universal strategy how to explicitly approximate any function that is a solution to some integral or differential equation. Let us start with derivation of the numerical algorithm for a function:

$$y(x) = a_0 + a_1 e^{b_1 x} + a_2 e^{b_2 x} \tag{2.12}$$

We start by calculating these integrals:

$$
\begin{aligned}
S(x) &= \int_{x_0}^{x} y(t) \mathrm{d}t \\
&= \int_{x_0}^{x} a_0 + a_1 e^{b_1 t} + a_2 e^{b_2 t} \mathrm{d}t \\
&= a_0(x - x_0) + \frac{a_1}{b_1}(e^{b_1 x} - e^{b_1 x_0}) + \frac{a_2}{b_2}(e^{b_2 x} - e^{b_2 x_0})
\end{aligned}
$$

$$
\begin{aligned}
SS(x) &= \int_{x_0}^{x} S(t) \mathrm{d}t \\
&= \int_{x_0}^{x} \int_{x_0}^{t} y(u) \mathrm{d}u \mathrm{d}t \\
&= \int_{x_0}^{x} a_0 t + \frac{a_1}{b_1} e^{b_1 t} + \frac{a_2}{b_2} e^{b_2 t} - \left(\frac{a_1}{b_1} e^{a_1 x_0} + \frac{a_2}{b_2} e^{a_2 x_0} + a_0 x_0 \right) \mathrm{d}t \\
&= \frac{1}{2} a_0 x^2 - \left(\frac{a_1}{b_1} e^{a_1 x_0} + \frac{a_2}{b_2} e^{a_2 x_0} + a_0 x_0 \right)(x - x_0) - \frac{1}{2} a_0 x_0^2 \\
&\quad + \frac{a_1}{b_1^2} \left(e^{b_1 x} - e^{b_1 x_0} \right) + \frac{a_2}{b_2^2} \left(e^{b_2 x} - e^{b_2 x_0} \right)
\end{aligned}
$$

Now, if we reorganize the terms it can be shown that there are constants A, B, C, D and E so that $y(x)$ can be written as:

$$y(x) = A \cdot SS(x) + B \cdot S(x) + Cx^2 + Dx + E. \tag{2.13}$$

By comparing coefficients before the exponential terms $e^{b_1 x}$ and $e^{b_2 x}$ we get:

$$a_1 = A \frac{a_1}{b_1^2} + B \frac{a_1}{b_1} \tag{2.14}$$

$$a_2 = A \frac{a_2}{b_2^2} + B \frac{a_2}{b_2} \tag{2.15}$$

From that it is possible to see that b_1 and b_2 are roots of the quadratic equation:

$$b^2 - bB - A = 0. \tag{2.16}$$

b_1 and b_2 are then:

$$b_1 = \frac{1}{2}\left(B + \sqrt{B^2 + 4A}\right) \tag{2.17}$$

$$b_2 = \frac{1}{2}\left(B - \sqrt{B^2 + 4A}\right) \tag{2.18}$$

The equation 2.13 is linear in the unknown parameters $A, .., E$ and after discretization it can be rewritten as:

$$\boldsymbol{y} = \boldsymbol{X} \cdot \boldsymbol{b} \tag{2.19}$$

where $\boldsymbol{y}$ is vector:

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{2.20}$$

Denoting $S_k = S(x_k)$ and $SS_k = SS(x_k)$ we can write:

$$\boldsymbol{X} = \begin{pmatrix} SS_1 & S_1 & x_1^2 & x_1 & 1 \\ SS_2 & S_2 & x_2^2 & x_2 & 1 \\ SS_3 & S_3 & x_3^2 & x_3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ SS_n & S_n & x_n^2 & x_n & 1 \end{pmatrix} \tag{2.21}$$

and

$$\boldsymbol{b} = \begin{pmatrix} A \\ B \\ C \\ D \\ E \end{pmatrix}. \tag{2.22}$$

We can use the well-known least squares method to calculate the parameters $A, ..., E$. We sort the data so that if $i < j$, then $x_i < x_j$ for $\forall i, j \in \{1, .., n\}$. The integrals are computed numerically as:

$$S_i = \begin{cases} 0 & i = 0 \\ S_{i-1} + \frac{1}{2}(y_i + y_{i-1})(x_i - x_{i-1}) & i \in \{2, ...n\} \end{cases} \tag{2.23}$$

and

$$SS_i = \begin{cases} 0 & i = 0 \\ SS_{i-1} + \frac{1}{2}(S_i + S_{i-1})(x_i - x_{i-1}) & i \in \{2, ...n\} \end{cases} \tag{2.24}$$

Now, the solution to the linear equation 2.19 can be written as [**lin-reg**]:

$$\boldsymbol{b} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \cdot \left(\boldsymbol{X}^T \boldsymbol{y}\right) \tag{2.25}$$

which in our case can be expanded to:

$$
\begin{pmatrix} A \\ B \\ C \\ D \\ E \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} SS_i^2 & \sum_{i=1}^{n} SS_iS_i & \sum_{i=1}^{n} SS_ix_i^2 & \sum_{i=1}^{n} SS_ix_i & \sum_{i=1}^{n} SS_i \\ \sum_{i=1}^{n} SS_iS_i & \sum_{i=1}^{n} S_i^2 & \sum_{i=1}^{n} S_ix_i^2 & \sum_{i=1}^{n} S_ix_i & \sum_{i=1}^{n} S_i \\ \sum_{i=1}^{n} SS_ix_i^2 & \sum_{i=1}^{n} S_ix_i^2 & \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} SS_ix_i & \sum_{i=1}^{n} S_ix_i & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} SS_i & \sum_{i=1}^{n} S_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & n \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} SS_iy_i \\ \sum_{i=1}^{n} S_iy_i \\ \sum_{i=1}^{n} x_i^2y_i \\ \sum_{i=1}^{n} x_iy_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}
$$

(2.26)

After we compute b_1 and b_2 from $\boldsymbol{b}$, we still have 3 unknown parameters a_0, a_1 and a_2. For those, we will take the original equation 2.12. As it is already linear in the parameters a_0, a_1 and a_2, we only need to pre-compute the values of $\alpha_k = \mathrm{e}^{b_1 x_k}$ and $\beta_k = \mathrm{e}^{b_2 x_k}$. One can see that we get a very similar equation as 2.26, that can be written in this form:

$$
\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} n & \sum_{i=1}^{n} \alpha_i & \sum_{i=1}^{n} \beta_i \\ \sum_{i=1}^{n} \alpha_i & \sum_{i=1}^{n} \alpha_i^2 & \sum_{i=1}^{n} \alpha_i\beta_i \\ \sum_{i=1}^{n} \beta_i & \sum_{i=1}^{n} \alpha_i\beta_i & \sum_{i=1}^{n} \beta_i^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} \alpha_iy_i \\ \sum_{i=1}^{n} \beta_iy_i \end{pmatrix}
$$

(2.27)

In [18], only the case without the constant a_0 is presented, but as we have shown the derivation of the generalized version with constant is not difficult. An extension of Jacquelin method for more than two exponential terms is also not very complicated. Note that adding one term adds two parameters, which leads to matrix of size 7×7. However, it always leads to solving for the roots of the polynomial created by the first N elements of the vector $\boldsymbol{b}$, where N is the number of exponential terms. In case of two exponential terms, equations 2.17 solve for the roots of the polynomial of Equation 2.16. In case of three exponential terms, we would need to calculate $SSS(x)$ from $SS(x)$. There would also be a term x^3. If we set $\boldsymbol{X}$ as $[SSS(x), SS(x), S(x), x^3, x^2, x^1, 1]$, we would solve for the roots of:

$$
b^3 - Cb^2 - Bb - A = 0.
$$

(2.28)

It is necessary to note, that this algorithm is prone to numerical defects coming from the fact that there is a cumulative error related to the calculation of the partial sums. For that reason the resulting matrices can be singular or the polynomial can have imaginary roots.

In [27], Mokhomo et al. did numerical experiments where they have shown that similar method (derived from differential instead of integral equations) can be used for fitting, but does not yield precise estimation results even for noiseless data.

Nevertheless, it is an explicit algorithm for approximation of parameters which then can be used as initial guess for more advanced iterative techniques.

When it comes to uncertainty of the coefficients, there are two possibilities. First, it is possible to propagate the experimental error from y_i all the way to the final coefficients. The derivation can be found in [22]. However, we do not have the standard error of the histogram bins. In principle, this can be obtained by repeating the simulation with the same input parameters but with a different seed for the random number generator which provides initial velocities for particles based on Maxwellian distribution.

The second option is to take the formula for linear regression and calculate the error estimation directly from that from that. In that case, we calculate estimates ΔA and ΔB as standard deviations for A and B. Then, using the error propagation formula for the quadratic roots we get:

$$\Delta b_1 = \sqrt{\frac{\Delta A^2}{(B^2 + 4A)} + \frac{\Delta B^2}{4}\left(1 + \frac{B}{\sqrt{B^2 + 4A}}\right)^2} \qquad (2.29)$$

and

$$\Delta b_2 = \sqrt{\frac{\Delta A^2}{(B^2 + 4A)} + \frac{\Delta B^2}{4}\left(1 - \frac{B}{\sqrt{B^2 + 4A}}\right)^2}, \qquad (2.30)$$

where Δb_1 and Δb_2 are the estimates of standard deviations for b_1 and b_2. The standard deviations of a_0, a_1 and a_2 are calculated directly from the second linear regression.

We will not follow same approach for more exponential terms, but it could be done.

Chapter 3

Surrogate hot electron temperature models

In this chapter, we will talk about the models suitable for hot electron temperature modelling. It was already explained in the first chapter that there are many physical processes contributing to the absorption of laser. As a consequence, there is a non-linear relationship between the initial parameters and the hot electron temperature. It would be very difficult to propose an analytical model where the temperature is explicitly dependant on all the initial parameters. There have been works studying how the temperature scales with respect to intensity [4, 10, 14, 20, 25], angle [10] or length scale and even pulse durations [25], but none of them models the dependency on all parameters at once. Needless to say, these works are valuable and they will be discussed with respect to predictions of our final model in chapter 5.3.

Given the complexity and multi-dimensionality of our dataset, few machine learning regression methods offer a promising alternative to analytical models. We will now discuss general principle of regression and several particular regression methods as well as their suitability for this thesis.

3.1   Regression methods

Regression is a fundamental technique in machine learning used for predicting continuous outcomes [6]. Formally, for the set of data $D = \{(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)\} \in \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ is d-dimensional space of inputs, we want to find a function $f(\boldsymbol{x}, \boldsymbol{w})$ that represents the relationship between $\boldsymbol{x_i}$ and y_i. $\boldsymbol{w} = (w_0, .., w_m)$ is the vector of parameters of f. In the observations, we assume a Gaussian noise with zero mean and variance σ^2 so we can write:

$$y_i = f(\boldsymbol{x_i}, \boldsymbol{w}) + \epsilon_i, \tag{3.1}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is the noise.

Moreover, we want the function $f(\boldsymbol{x}, \boldsymbol{w})$ to be able to generalize - to give accurate predictions for inputs not present in the training dataset. A good regression process

balances the trade-off between fitting the training data accurately and maintaining predictive performance on unseen data. This condition transforms most regression problems to an optimization problem, where one tries to find the best model under certain conditions.

A good example is generalized linear regression where the model is given by:

$$y = w_0 + \sum_{i=1}^{m} \phi_i(\boldsymbol{x})w_i = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{3.2}$$

where $\phi_i(\boldsymbol{x})$, $i = 1, .., m$ are transformation functions of the inputs also called *basis functions* and $\phi_0(\boldsymbol{x}) = 1$ [6]. A reasonable metric of the goodness of this model (sometimes also called *loss*) can be the sum of square of errors (SSE) calculated on dataset:

$$SSE(\boldsymbol{w}) = \frac{1}{2} \sum_{i=0}^{n} \left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x_i}) - y_i \right)^2 \tag{3.3}$$

Minimizing SSE with respect to weights in simple problems with correctly chosen $\boldsymbol{\phi}$ is often enough to get a reliable predictor. This usually requires a good knowledge about the data. If the model is over-fitted - SSE is small for training data but large for new data - modifying the error function might help. For example, one can add *regularization* term $\frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$ to the SSE. This introduces *hyper-parameter* λ, which controls the importance of weights being small. The loss function to minimize is:

$$SSE_{\text{reg}}(\boldsymbol{w}) = \frac{1}{2} \sum_{i=0}^{n} \left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x_i}) - y_i \right)^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{3.4}$$

Minimizing SSE_{reg} instead of SSE sometimes helps complex models to improve their predictive performance [6].

The optimization problem can be reversed to so called maximum likelihood estimation (MLE), where instead of minimizing an error function, one tries to find model maximizing the probability that we observe data given the set of parameters. In case of Gaussian noise, this approach is equivalent to minimizing SSE [6].

The appropriate selection of the feature mapping $\boldsymbol{\phi}$ enables one to effectively capture non-linear relationships. However, in situations where determining the exact form of $\boldsymbol{\phi}$ is ambiguous, various methodologies have been developed to accommodate modelling without explicit knowledge of the underlying model structure. In the following sections, we look into advanced techniques for non-linear regression. We begin with Support Vector Regression (SVR), followed by a brief overview of Neural Networks, and finally, a more detailed exploration of Gaussian Processes.

3.2   Support vector regression

Support vector regression (SVR) is powerful regression technique capable of modelling non-linear relationship without any previous assumption about the model. From its formulation it is directly set up to balance complexity and prediction error [48]. In several following paragraphs, we will present the most important principles of SVR.

Linear ϵ-SVR model

Assuming simple linear model we can simplify equation 3.2 to $y = \bar{\boldsymbol{w}}^T \boldsymbol{x} + w_0$ where $\bar{\boldsymbol{w}} = (w_1, .., w_d)$. In ϵ-SVR, there are constraints put on the predictions in such way that the prediction $\bar{\boldsymbol{w}}^T \boldsymbol{x}_i + w_0$ cannot be further from y_i than ϵ. We can imagine that there is a *tube* with radius 2ϵ around the training points. We also want the *tube* defined by ϵ to be as *flat* possible, which can be expressed as minimizing $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$ [48].

In reality, having rigid boundary at distance ϵ from y_i is not practical because of outliers. It is therefore natural to introduce two *slack* variables ξ_i and ξ_i^*, which are used to widen the tube at the points of observation. The optimization problem can be formulated as:

$$\text{minimize} \qquad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{3.5}$$

$$\text{subject to} \qquad \begin{cases} y_i - (\bar{\boldsymbol{w}}^T\boldsymbol{x}_i + w_0) & \leq \epsilon + \xi_i \\ (\bar{\boldsymbol{w}}^T\boldsymbol{x}_i + w_0) - y_i & \leq \epsilon + \xi_i^* \\ \xi_i,\, \xi_i^* & \geq 0 \end{cases} \tag{3.6}$$

where $C > 0$ is a hyper-parameter determining the trade-off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated [37].

Kernel SVR

We can leave the assumption that the relationship between $\boldsymbol{x}$ and y is linear. We do this by so-called *kernel trick* - transforming features $\boldsymbol{x}_i$ to a higher-dimensional kernel space $\mathcal{F}$. Let us note the mapping $\phi(\cdot) : \mathcal{X} \to \mathcal{F}$. $\boldsymbol{x}_i$ in 3.6 is replaced by $\phi(\boldsymbol{x})$. Moreover, the new optimization problem can be written in dual form [37]:

$$\max_{\alpha_i, \alpha_i^*} \left\{ -\frac{1}{2}\sum_{i,j}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \epsilon\sum_{i}^{n}(\alpha_i + \alpha_i^*) + \sum_{i}^{n}y_i(\alpha_i - \alpha_i^*) \right\} \tag{3.7}$$

$$\text{subject to} \qquad \sum_{i}^{n}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C] \tag{3.8}$$

where $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j)$ is the kernel function and α_i and α_i^* are parameters of the model in the dual problem. α_i and α_i^* are defined by [37]:

$$\boldsymbol{w} = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\phi(\boldsymbol{x_i}). \tag{3.9}$$

There are multiple conditions which the kernel function must satisfy and they can be found in [37]. Popular kernels include linear, polynomial, radial basis function and others [48]. Note that $\boldsymbol{w}$ is no longer explicitly present. Another important fact is that the flatness is now relevant in the new space $\mathcal{F}$ and not the original input space $\mathcal{X}$ [37].

To determine the hyper-parameters ϵ and C (or other parameters associated with the kernel itself), it is customary to perform a *cross-validation* where we randomly divide the dataset to two parts - training set and test set - and evaluate the predictive performance only on the latter which is not used during the optimization [48].

3.3   Neural networks

Neural networks have a different approach. They still assume almost nothing about the modelled relationship, but as opposed to SVR, where the goal is to minimize the confidence interval by searching for a flat solution, neural networks usually search directly for minimum error on the training data [42].

The simplest neural network is probably the Multilayer Perceptron (MLP). A MLP is a type of neural network that consists of an input layer, one or more *hidden* layers, and an output layer. Each layer is composed of neurons, and each neuron in a layer is connected to every neuron in the subsequent layer. Neuron linearly combines all the outputs from the previous layer using its own set of weights and then transforms the result to its output using some function. The mathematical formulation of an MLP can be described as follows:

1. **Input Layer:** Let the input vector be $\boldsymbol{x}$.

2. **Hidden Layers:** Consider a *single* hidden layer with m neurons. The output of the j-th neuron in the hidden layer, denoted as h_j, is given by:

$$h_j = \sigma \left(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)} \right) \tag{3.10}$$

where $w_{ij}^{(1)}$ is the weight connecting the i-th input to the j-th hidden neuron, $b_j^{(1)}$ is the bias term for the j-th hidden neuron, and σ is the *activation function*, that introduces non-linearity.

3. **Output Layer:** Let the output layer have 1 neuron. The output of the l-th neuron in the output layer, denoted as y_l, is given by:

$$y_l = \phi \left(\sum_{j=1}^{m} w_{jl}^{(2)} h_j + b_l^{(2)} \right) \tag{3.11}$$

where $w_{jl}^{(2)}$ is the weight connecting the j-th hidden neuron to the l-th output neuron, $b_l^{(2)}$ is the bias term for the l-th output neuron, and ϕ is the activation function for the output layer (for regression ϕ is the identity) [6].

Note that for each layer weights are well represented as matrix. Number of hidden layers and the number of neurons has to be chosen. However, even with only two layers with correct non-linear activation functions, any continuous function on a compact domain can be approximated to arbitrary precision if the total number of neurons is large enough [6].

Training the MLP

The training of neural network includes usually calculating gradients of the loss function with respect to the weights, propagating the gradients back through the network and updating the weights - also known as *gradient descent*. It is usually the case that if the loss and the activations are differentiable functions (w.r. to the weights), because their derivative is needed for the error back-propagation [6].

Gradient descent alone can find local minimum of the error function for the training data but it says nothing about the ability to recognize patterns. There are multiple *regularization* strategies that help neural network generalize but in contrast to SVR they have to be fine-tuned. Common ones are early stopping (not letting the weights to come to the local minimum completely), weight decay (e.g. using SSE_{reg}), batch normalization (the gradients are averaged for multiple training points), or dropout (in each learning process iteration we exclude few neurons from the network) [6, 38].

When training complex non-linear relationship on small datasets (up to 1000 samples), it can be the case that the complexity cannot be captured by network with less weights than the number of training points. This is called *over-parametrization*. However, it is sometimes still valid to train such network, because the effective number of degrees of freedom can be lowered by regularization [3, 17]. Tuning over-parametrized NNs is more difficult, but seeing its predictions side by side with other models can give us a better insight into the properties of our data.

3.4 Gaussian Processes

Another viable regression method is Gaussian Process regression. Its approach is completely different compared to both previous methods. Following paragraphs are explaining the basics of Gaussian Processes.

Gaussian Process (GP) can be understood as an extension of multivariate Gaussian distribution over vectors to distribution over functions. Informally, we replace $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is mean and $\boldsymbol{\Sigma}$ is the covariance matrix, with Gaussian Process $\mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$, where $m(\boldsymbol{x})$ is mean function and $k(\boldsymbol{x}, \boldsymbol{x}')$ is covariance function. $m(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}')$ fully specify a Gaussian process. For every input $\boldsymbol{x}$ there is now a random variable $f(\boldsymbol{x})$ for which we can write [31]:

$$f \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')). \tag{3.12}$$

Function k is commonly called *kernel* function as in SVR. Sampling from f can then be done easily, if we know k and m and if we realize that we can usually represent sample function by finite amount of points.

For sampling, we first calculate covariance matrix $\Sigma_{i,j} = k(\bar{x}_i, \bar{x}_j)$ for points $\bar{\boldsymbol{x}} = (\bar{x}_1, .., \bar{x}_s)$, at which we want to plot the sampled functions. Each sample is then a vector of size s given by multivariate Gaussian distribution with covariance matrix $\boldsymbol{\Sigma}$. As an example, 5 samples from Gaussian Process with $m(x) = 0$ and $k(x, x') =$
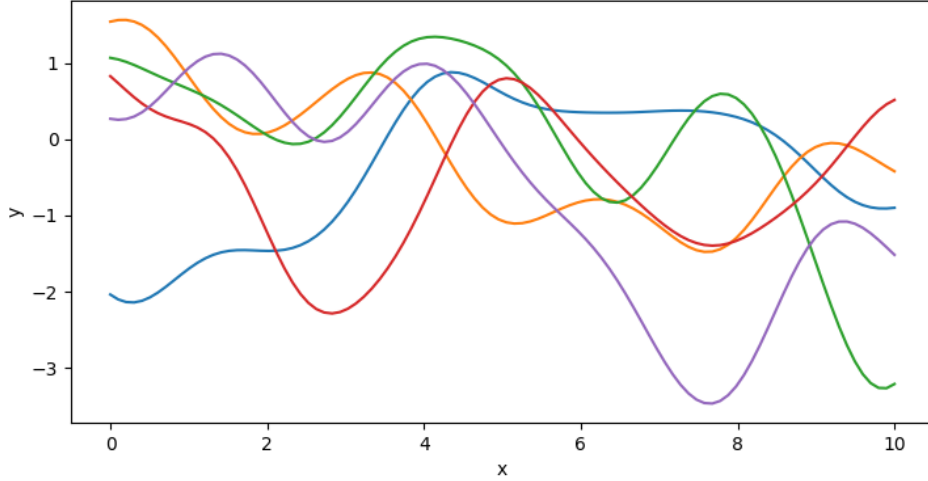
Figure 3.1: 5 sample functions from $\mathcal{GP}(0, \exp(-\frac{(x-x')^2}{2}))$. It was sampled at 100 points between $x = 0$ and $x = 10$.

$\exp(-\frac{(x-x')^2}{2})$ can be seen in figure 3.1. Note that the smoothness is a consequence of the choice of the kernel function.

The next step is to update the $\mathcal{GP}$ with the measured data. In other words, we want to condition f from 3.12 on dataset D. We can write [31]:

$$f|D \sim \mathcal{GP}(m_D, k_D)$$
$$m_D(\boldsymbol{x}) = m(\boldsymbol{x}) + \boldsymbol{\Sigma}_*(\boldsymbol{X}, \boldsymbol{x})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{f} - \boldsymbol{m}) \tag{3.13}$$
$$k_D(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{\Sigma}_*(\boldsymbol{X}, \boldsymbol{x})^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}_*(\boldsymbol{X}, \boldsymbol{x}'),$$

where $\boldsymbol{\Sigma}_*(X, x)$ is a vector of covariances between every training case $\boldsymbol{X}$ and $\boldsymbol{x}$. Applying equation 3.13 to our previous example, we can generate new samples from the posterior distribution of functions. The term *posterior* comes from the Bayesian inference. If we let for example the training dataset consist of five points $D = \{(1, 2), (3, 2), (5, 3), (7, 4), (9, 2)\}$, the 5 samples from the updated distribution can be seen in Figure 3.2.

One can see that Figure 3.2 has sample functions that go through the points in D. This is because we set D as the condition. A less strict condition would be to set a non-zero noise to the dataset. This allows the samples not to go strictly through all points in the dataset. Mathematically speaking, adding noise is represented by adding a diagonal matrix to the covariance matrix or even more general [31]:

$$y = f + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$
$$f \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \qquad y \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}') + \delta_{xx'}\sigma^2), \tag{3.14}$$

where $\delta_{xx'} = 1$ for $x = x'$ and $\delta_{xx'} = 0$ for $x \neq x'$. In general, σ can be different for each training point.

The understanding of the training principle might be easier if one sees how we created Figure 3.2. The Python code used to generate this graph can be seen in Appendix B.

36
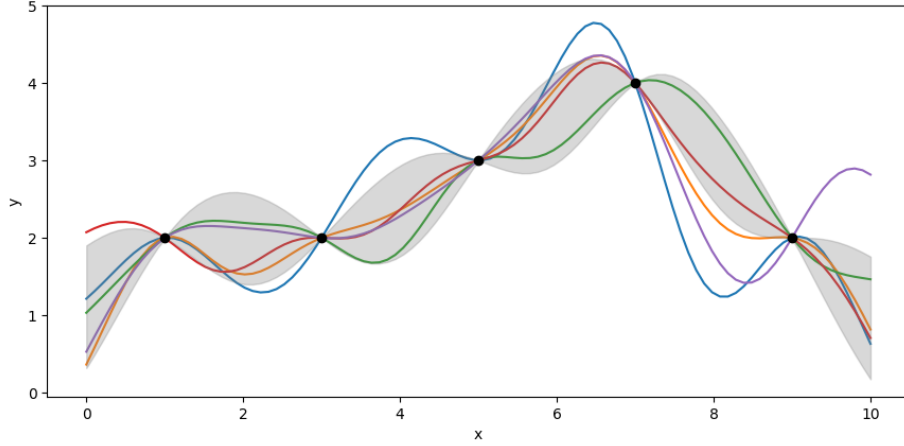
Figure 3.2: 5 sample functions from $\mathcal{GP}(m_D, k_D)$). It was sampled at 100 points between $x = 0$ and $x = 10$. The grey area depicts one standard deviation from the mean.

Apart from RBF kernel $k_{\text{RBF}}(x - x') = \sigma^2 \exp(-\frac{(x-x')^2}{2l})$ another common choice are kernels from the Matern class [32]:

$$k_{\text{Matern}}(r) = \sigma_k^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^\nu K_\nu \left(\frac{\sqrt{2\nu}r}{l}\right), \tag{3.15}$$

where σ_k ν and l are positive paremeters, $r = |\boldsymbol{x} - \boldsymbol{x}'|$, $\Gamma(\nu)$ is a gamma function and K_ν is a modified Bessel function. Common choices of ν are $\frac{1}{2}, \frac{3}{2}$ and $\frac{5}{2}$. For $n \to \infty$, $k_{\text{Matern}}(|\boldsymbol{x} - \boldsymbol{x}'|)$ degenerates to $k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}')$ [32].

Gaussian process regression is a non-parametric model, meaning the training data cannot be discarded after the training, because they are needed for the computation of $\boldsymbol{\Sigma}_*$. The memory complexity of the algorithm is $O(n^2)$ and the time complexity is $O(n^3)$, because of the costly matrix inversion needed for computation of the posterior distribution [31]. For small datasets, however, this is manageable by standard computers.

Apart from that, the tuning of the hyper-parameters such as σ, the choice of the covariance function or other parameters is usually done using maximum-likelihood methods [31].

To the contrast of previous two models, which assume nothing about the data, in GP it is helpful to specify the prior mean function. It can be specified parametrically and the parameters can be tuned during hyper-parameter tuning. Having prior mean $m(\boldsymbol{x}) = 0$ makes the model more simple but can introduce bias in the predictions if it is not handled. Common technique of dealing with this, implemented for example in the GPy library, is to normalize y values so that they actually have zero mean before [40].

For us, it is particularly interesting to have the possibility of calculating the uncertainty of the prediction as can be seen in Figure 3.2. Of course, it is mainly

related to the density of the dataset in the studied domain. However, knowing the model uncertainty as function of input parameters can be leveraged for selection of parameters for new simulations with the goal of making the model more precise by expanding the training data. We will come back to this in Chapter 5.

Chapter 4

Dataset

In this chapter we will describe in detail, what simulations did we run and and what we found was the best strategy to do the automated temperature fitting. In the last section, we dive discuss the dataset obtained by temperature fitting.

4.1   EPOCH Simulations

The EPOCH simulation was run in following setting:

- The laser wavelength is 1 micron.

- The laser is p-polarized and focused to a Gaussian spot of size 3.2 mircons.

- The density in the target ranges from about $0.01n_c$ to $3.\gamma_{osc}n_c$, where n_c is critical density of laser radiation [10] and γ_{osc} is defined in [10].

- The initial temperature of plasma is 100 eV.

- The target is composed of electrons and protons. They are represented by 30 macro-particles per cell.

- The resolution of spatial grid is 33nm and the time step satisfies the CFL condition [2].

- The intensities of the laser: $I = 10^{17}$ W.cm^{-2}, $I = 10^{18}$ W.cm^{-2} and $I = 10^{19}$ W.cm^{-2}.

- The angle of incidence with respect to target normal direction: $\alpha \in \{0°, 1°, 2°, 3°, 4°, 5°, 10°, 20°, 30°, 40°, 45°, 50°, 60°\}$.

- The characteristic scale length of the preplasma: $L \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5\}$ in microns.

The simulations have been performed on the Q3 node of the Quantum Hyperion cluster at FNSPE. The input file that used to start the EPOCH simulation can be seen in appendix A.

4.2 Temperature fitting

The results of the simulations are transformed into histograms as it was described in the beginning of chapter 2. We fixed the energy histogram size to 1000 bins with their width scaling with the maximum electron energy. In several following pages, we will describe a strategy we developed and used to find the temperature from the vast majority of the energy histograms. The effectiveness of this strategy varies for different histograms, because it depends on multiple properties.

To get the best possible hot electron temperature fit, three things are important. First, because of the imperfections of the histogram related to the beginning and the end of the energy spectrum, it is helpful to narrow down the fitted region. Secondly, it is still necessary to perform a good fit automatically ideally without many issues. Last but not least, the fitting strategy has to be able to deal with the fact that the energy range and temperatures can for different histograms from the same dataset vary by several orders of magnitude. We will now present the strategy developed for the purpose of this thesis.

Fitting strategy

Before fitting the data, it is essential to prepare it appropriately. In practice, this involves removing a few bins in the beginning of the electron energy spectrum. The lower energy bins can be removed, because they can be empty or at least be affected by error, as the low energy electrons need more time to reach the virtual detector and the simulation can end before that happens. This is usually the case for histograms from simulations with lower laser intensity. Also, we made an approximation by considering Boltzmann distribution which does not work for small energies very well. Cutting the beginning is therefore justified.

We also cut the end of the spectrum for reasons discussed in chapter 2. We cut it in a place where there is a first empty bin. Apart from the main reason, it is also not practical to work with empty bins in the logarithmically transformed version of the histogram. In special cases, it might be possible to work with histograms cut with less strict approach. However, these cut-offs help the analysis to focus on the relevant parts of the spectrum.

The result of the cut-off of histogram 2.1 can be seen in figure 4.1. Notice that the data show non-symmetrical noise in the part of the spectrum with the highest energies. This can be attributed to the logarithmic scale which can skew and seemingly magnify the noise for lower electron counts.

For the purpose of the fitting, it is suitable to approximate the histogram data as points with x values equal to the centre of the corresponding histogram bin and y value equal to count.

The Jacquelin method was implemented in Python programming language as a class with number of exponential terms as a parameter. However, the lack of the numerical stability for more exponential terms usually causes issues for more than three terms
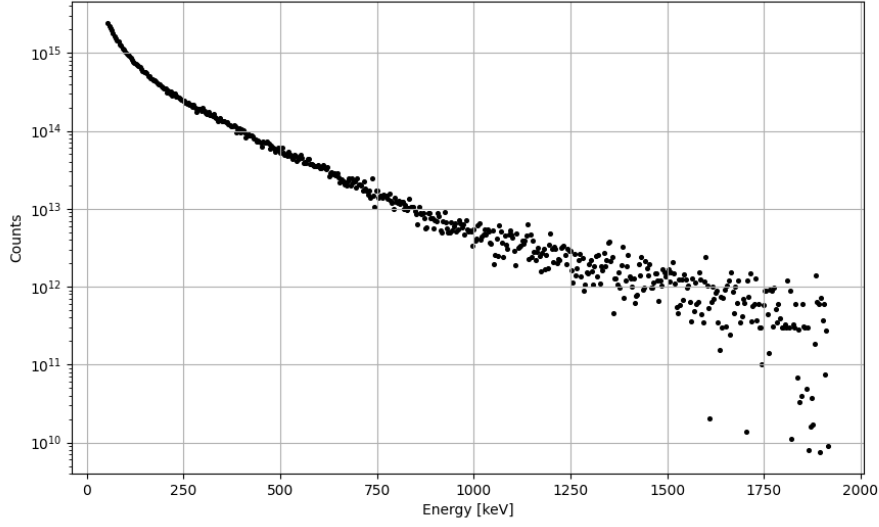
Figure 4.1: An example of trimmed histogram with simulation parameters $I = 10^{19}\,\mathrm{W.cm^{-2}}$, $L = 0.1\,\mu\mathrm{m}$ and $\alpha = 10°$.

because of reasons discussed at the end of chapter 2. One can also choose whether the constant term should be included.

As mentioned previously, the distributions other than the hot electron distribution are not of primary concern. It is reasonable to assume that the quality of the fit improves if the fitted region is minimized, provided it still encompasses the section dominated by hot electrons. By definition, in an ideal case where the region is selected perfectly, the fit would be influenced only by noise and the other distributions, which decrease exponentially much faster as E increases.

That being said, the next step aims to cut (or rather ignore) the beginning histogram once more, but now in a more sophisticated way. We found, that the majority of the histograms can be fit by the three-exponential Jacquelin method without the constant term. Of the three, there is one exponential distribution dominant for small energies. The corresponding electron distribution temperature is the lowest of all three distributions. Even though some histograms do not have three distributions present, the fit usually does not fail and the lowest temperature distribution can be (roughly) identified. We can quantify the energy E_{crit} at which the two most dominant terms are equal using equation:

$$E_{\mathrm{crit}} = \frac{\ln\left(a_1/a_2\right)}{b_2 - b_1} \tag{4.1}$$

and use it as threshold for the cut-off.

The basic illustration of the idea we just described can be seen in figure 4.2. Note that E_{crit} intersects the fitted curve approximately in the middle of the arc where one exponential term is equal to the other. To make the lowest temperature electron distribution insignificant, the cut-off is made even a little further right. The hot
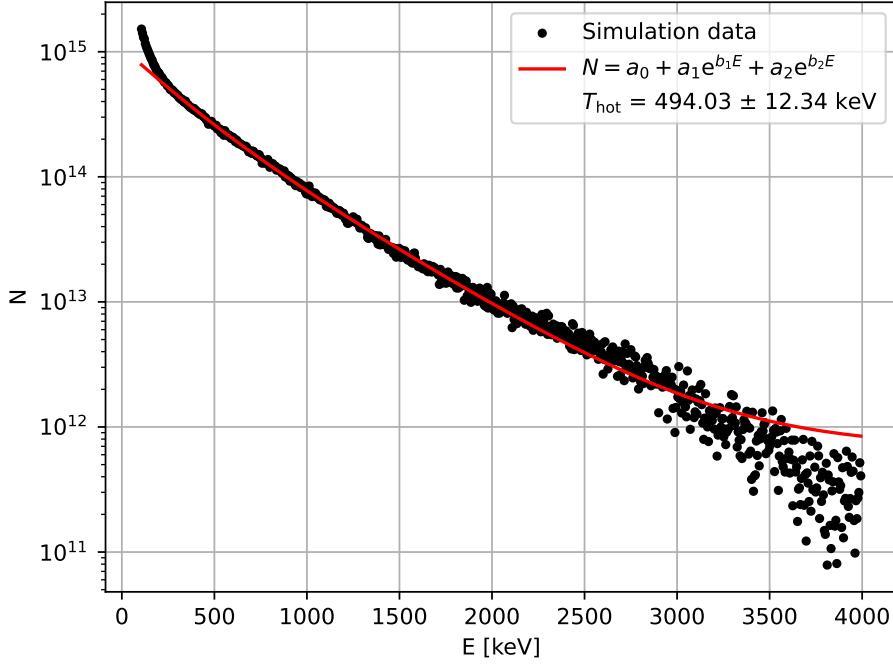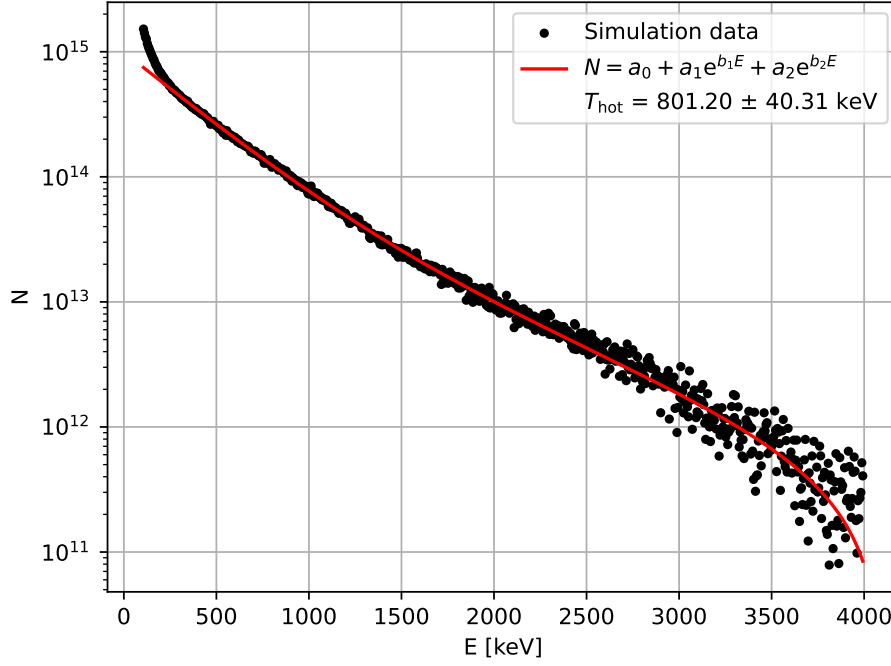
41
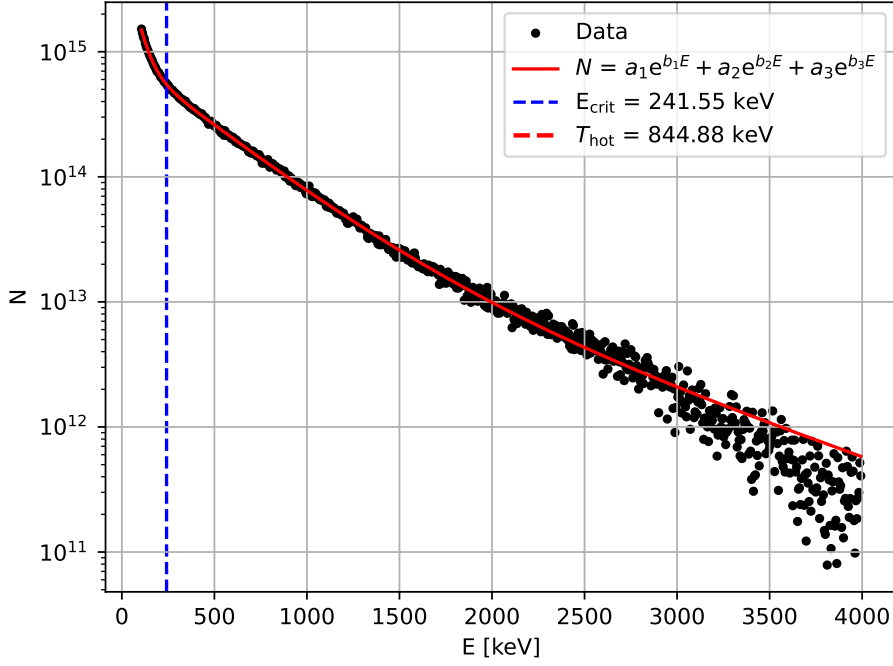
Figure 4.2: An example of histogram with simulation parameters $I = 10^{19} \, \text{W.cm}^{-2}$, $L = 0.6 \, \mu\text{m}$ and $\alpha = 30°$ fitted with three-exponential Jacquelin method. The energy E_{crit} is highlighted.

electron temperature T_{hot} in figure 4.2 calculated by:

$$T_{\text{hot}} = -\frac{1}{b_3}, \tag{4.2}$$

where b_3 is the highest negative exponential coefficient of the whole fit. The attribute *negative* is emphasized for practical reasons, because it is not guaranteed that all coefficients are. To be more precise, in some fits, the absolute value of one of the factors a_i ($i = 1..3$) is small. Then the corresponding b_i can be positive without significant impact on the fitted section of the histogram. In the limit of energy going to infinity, this does not have a valid physical interpretation, but in this intermediate step that does not have to bother us. In the other cases of the fit failing altogether, it often helps to narrow down the energy range even more from both ends (e.g. by 5 to 10 bins from each side) and try it again. If that fails enough times there will be nothing left and one should rather fit it differently.

We found, that in most cases after the second cut, it is now possible to perform the two-exponential Jacquelin method. The cut-off reduced the cumulative error of the partial sums, but what is more important, it justifies the lower number of the exponential terms. From our experience, the two-exponential Jacquelin fit is generally also a little bit more stable if performed on a histogram that resembles a two-exponential distribution.

The "removal" of the lowest temperature electron distribution we just defined is

Figure 4.3: An example of histogram with simulation parameters $I = 10^{19}\,\mathrm{W.cm^{-2}}$, $L = 0.6\,\mu\mathrm{m}$ and $\alpha = 30°$ fitted with two-exponential Jacquelin method.

similar to *successive subtraction* described in chapter 2 in section about fitting multi-exponential functions. Here we do a similar step but from the opposite end.

The two exponential fit performed on the cut version of the histogram in figure 4.2 with the cut-off part plotted but not fitted is shown in figure 4.3. It is possible to see that T_{hot} estimate is lower than it was before. Whether two-exponential Jacquelin method on cut histogram provides better T_{hot} estimates compared to three-exponential Jacquelin method on the original histogram will be a part of a discussion at the end of this section. The estimate of the standard error of T_{hot} is calculated as:

$$\Delta T_{\mathrm{hot}} = \frac{\Delta b}{b^2} = T_{\mathrm{hot}}^2 \Delta b, \tag{4.3}$$

where b is the corresponding exponential coefficient and Δb is its standard deviation calculated using equation 2.29 or 2.30.

As the last step of our strategy, we now use an non-linear least squares (NLSQ) algorithm to find a fit using the last result as the initial guess. It can be argued that this step is not truly necessary, because we do not have a good reason to assume that the ill-conditioning of this problem disappeared by a few cuts. However, we found, that it many cases the good initial guess provided by the explicit Jacquelin method can guide the optimizer to find a better solution.

In general, this technique is prone to failing as it is iterative and the convergence is not guaranteed. In either case, it provides an additional insight to the quality of the fit and with . The example of the histogram from figure 4.3 fitted with iterative
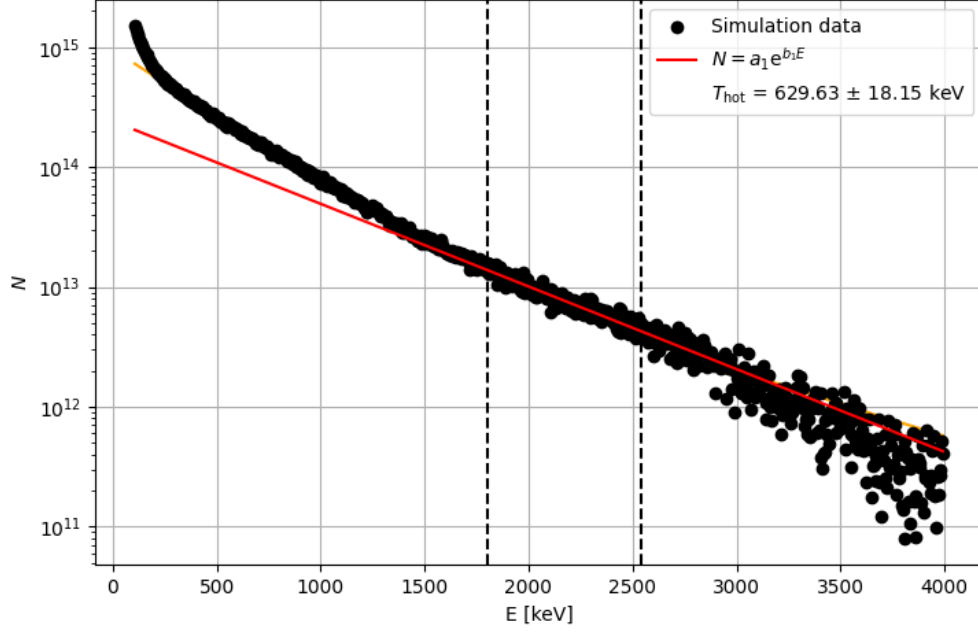
Figure 4.4: An example of histogram with simulation parameters $I = 10^{19}\,\mathrm{W.cm^{-2}}$, $L = 0.6\,\mu\mathrm{m}$ and $\alpha = 30°$ fitted with weighted non-linear least-squares.

non-linear square method using *scipy.optimize.curve_fit* function in Python can be seen in figure 4.4.

This was the last step of our strategy. We will discuss how the quality of the intermediate T_{hot} estimates changes with each step. The comparison is done with respect to *manually* fitted histograms. *Manually* means, that we select the region where distribution of hot electron dominates and perform one-exponential fit. We use an iterative weighted NLSQ method using coefficients from linear regression as initial guess. It is true, that the selection is subjective, but as of now, it is the most reliable way of finding the hot electron temperature.

As we said in the chapter 2, a tool was developed to make the manual fitting of T_{hot} more effective. The effectiveness has roots in the possibility to load the whole folder with histograms and for each select the most reasonable region with two clicks of the mouse. The tool was developed in Python using PyQt6 framework.

There is no need to show the functionality of the tool in detail. It is basic tool tailored to make the fitting easier while also working automatically creating the dataset. We are also using custom format to save the dataset so it might be in the future extended to support different fitting strategies. Currently the only option is to select the range and click "Fit". The tool also provides an option to show the original fit obtained by automated process.

Example of a fit performed manually can be seen in the figure 4.5. First thing to notice is the selected region. In this example it is quite narrow, but maybe it could

Figure 4.5: An example of histogram with simulation parameters $I = 10^{19}\,\mathrm{W.cm^{-2}}$, $L = 0.6\,\mu\mathrm{m}$ and $\alpha = 30°$ fitted manually with the fitting tool.

have been maybe a little wider while still only enclosing the right region. Other than that, it is now clear that the automated fit of temperature is in this series of examples off by a big margin despite the fit the figure 4.4 is seemingly behaving well.

Quantitative analysis of the fitting strategy and its T_{hot} estimates will be done in the following way. For each histogram for which the NLSQ method converged (359 out of total 373), we take all three T_{hot} estimates of 3-exponential method $T_{\mathrm{hot,3exp}}$, 2-exponential method $T_{\mathrm{hot,2exp}}$ and NLSQ estimate $T_{\mathrm{hot,NLSQ}}$ and calculate their relative error to the manual fit estimate $T_{\mathrm{hot,manual}}$ using:

$$\mathrm{Relative\,error} = \frac{T_{\mathrm{hot,2exp}} - T_{\mathrm{hot,manual}}}{T_{\mathrm{hot,manual}}} \tag{4.4}$$

and for $T_{\mathrm{hot,NLSQ}}$ and $T_{\mathrm{hot,3exp}}$ analogously. A comparison of mean square relative error for all three steps of the strategy is shown in Table 4.1. From this table, it is evident that in general it can be said that the fit improves during the strategy, because the relative error is most likely better for NLSQ than for 2-exponential fit.

Method	MSRE
3exp	16.61
2exp	0.32
NLSQ	0.06

Table 4.1: Mean Squared Relative Errors for fitting methods.

We plot the relative errors with respect to $T_{\text{hot,manual}}$ for each evaluated simulation. The scatter plot containing the relative errors can be seen in the figure 4.6. We are not plotting the 3-exp results. From this graph, we can conclude that the majority of the T_{hot} estimates is within the relative error of 0.5. That is probably acceptable for small values of T_{hot} (up to 20 keV), because the absolute value of this error is not significant. For higher energies, having relative error 0.5 is disappointing to say the least, but it again points to the fact that the automation of the fitting is very challenging.
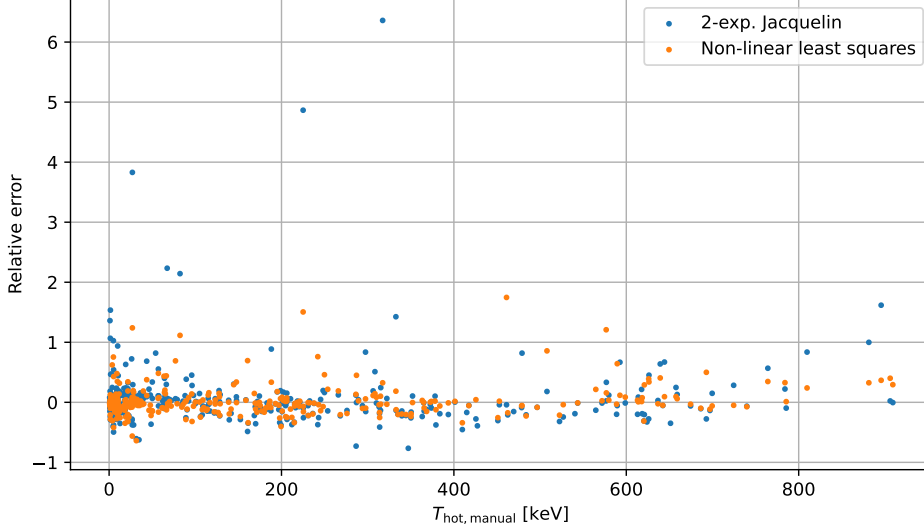
Figure 4.6: Relative errors of the T_{hot} for two fitting methods.

Despite various attempts, we were unable to make further progress from this point. One potential approach is to repeat the cut-off process. However, this introduces several complications. First, it is challenging to perform the cut-off without excessively reducing the region dominated by the T_{hot} distribution. During the initial cut-off, removing a larger portion of the second distribution is not problematic. However, in this case, it significantly impacts the results, likely because the end of the histogram does not belong to any of the distributions and therefore it is skewing the fit. Secondly, the one-exponential fit is more sensitive to data that do not belong to the fitted distribution. This sensitivity means that if the histogram is cut in a way that leaves a trace of the distribution intended to be removed, the one-exponential fit produces worse estimates of T_{hot} than the two-exponential fit.

A metric for comparing two fits of the same histogram side by side is also not trivial. Presenting such metric was one of the goals of this thesis. A comparison of the automated fits can be done by using weighted mean squared error:

$$WMSE = \frac{\sum_{i=1}^{n} w_i (y_i - \bar{y}_i)^2}{\sum_{i=1}^{n} w_i} \tag{4.5}$$

where $\bar{y}_i$ is the predicted value and w_i are weights. The weights can be for instance set equal to standard deviation estimated as using Poisson statistics $\sigma_i = 1/y_i$. The

idea is to compensate for the the bigger variance for smaller energies. After using our fitting strategy, this metric correctly labels 245 fits out of 359 to be better compared with respect to manually fitted value.

In conclusion, there definitely is a room for improvement of this process, if one wants to rely on fitting thousands of histograms at once. The presented strategy can produce acceptable results, but in current state it probably shouldn't be completely unsupervised. The failed fits have to be handled separately and without briefly checking each histogram, the reliability is questionable. Needless to say, it may serve as starting point for further improvement.

4.3   Full dataset

In this section, we will present the dataset used for modelling. Naturally, the results from 378 simulations contain a lot of information most if which will be discarded. As we concluded at the end of the previous section, the automation of the temperature fitting brings many challenges, some of which were not exhaustingly solved. From now on, we will be using the manually labelled dataset as such data are considered the most reliable. The main goal of this section is to visualize the interesting aspects of the data we obtained so far.

The visualization of the dataset itself can give us valuable insight to the behaviour of the function of $T_{\mathrm{hot}} = T_{\mathrm{hot}}(I, l, \alpha)$. Because there are three quantities by which T_{hot} is being modelled, the visualization can be tricky. The issue can be approached from multiple points of view.

Let us start by showing the dependency of the fitted temperature on simulation parameters with three fixed intensities $I = 1 \times 10^{17}\,\mathrm{W.cm^{-2}}$, $I = 1 \times 10^{18}\,\mathrm{W.cm^{-2}}$ and $I = 1 \times 10^{19}\,\mathrm{W.cm^{-2}}$. This way we can "slice" through the three dimensional space of parameters and get a good idea what we are modelling. By the way, this approach is almost completely corresponding to the way the parameter space was sampled for the simulations. Most of the parameters were chosen as a points of grid with certain steps in α and logarithmic steps in l. This is not true for all points and because of that, these graphs were made using linear interpolation to the grid of similar density. In other words, the values should not be considered exact and the graphs serve only as illustration. The graphs can be seen in figure 4.7.

The initial observation from these three graphs can for instance be the increasing temperature scales on the right side of each graph, which corresponds to the increasing intensity. This demonstrates the fundamental relationship: as the energy carried by the laser pulse increases, the temperature of the hot electrons rises too. Next thing one might notice is that size of the temperatures range also changes with intensity. For both $I = 1 \times 10^{17}\mathrm{W.cm^{-2}}$ and $I = 1 \times 10^{18}\mathrm{W.cm^{-2}}$, there is a region (or simulation point) with electron energy close to zero. For intensity $I = 1 \times 10^{19}\mathrm{W.cm^{-2}}$ there is no such phenomenon.
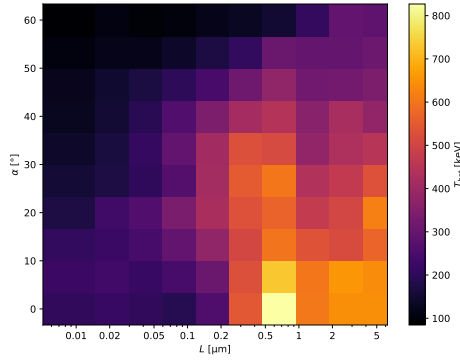
It is also possible to make some qualitative comments about the dataset in the light of the absorption theory which was briefly discussed in chapter 1. We will come
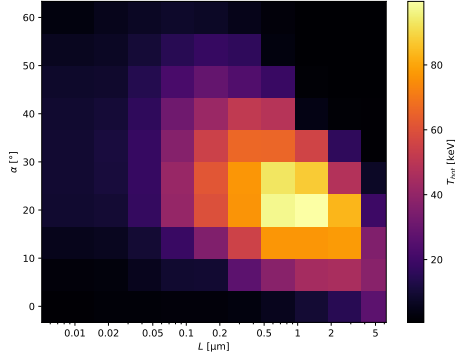
(a) Dataset with $I = 1 \times 10^{17}$W.cm^{-2}.
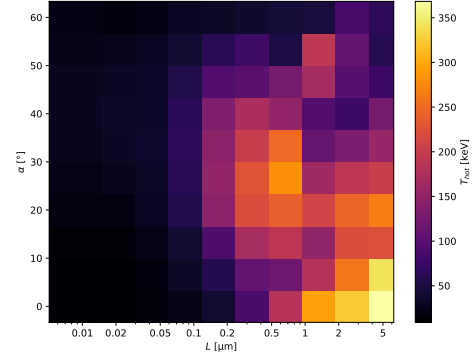
(b) Dataset with $I = 1 \times 10^{18}$W.cm^{-2}.

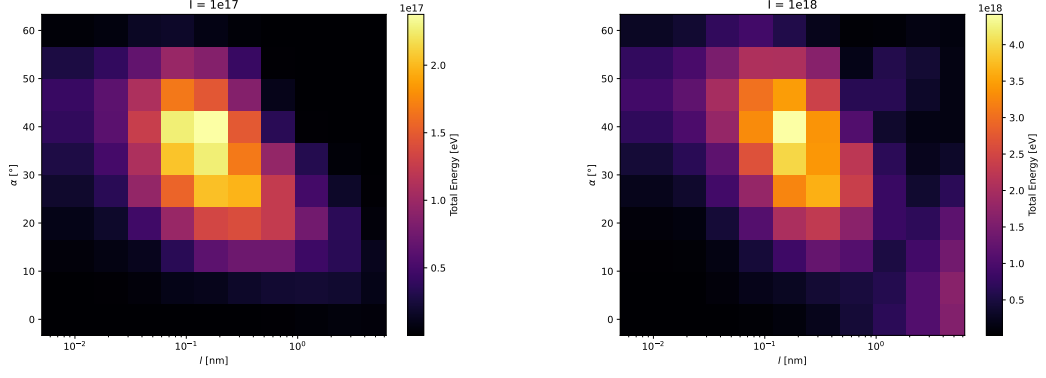(c) Dataset with $I = 1 \times 10^{18}$W.cm^{-2}.

Figure 4.7: Illustration of dataset shown for three simulated intensities.

back to this later.

It is important to realize, that these graphs do not represent the general absorption patterns. To compare total absorption across the domain of simulation points, it is necessary to also include the energy absorbed by the rest of the electrons. We will visualize it in similar way but instead of temperature, we look at the overall energy carried by all the electrons. This can be integrating each of the histograms. For the same reasons as in the previous section, we restrict ourselves to electron energies higher than threshold of 10 keV. The total absorbed energy for the same three intensities is displayed in the graphs in figure 4.8.

It is possible to see, that the maximum of overall absorption is happening in cases of bigger incidence angles. This can be attributed to ???. Also for the intensity $I = 1 \times 10^{19}$W.cm^{-2} there is relatively larger absorption happening for small angles and large scale lengths. All this is in agreement with what was presented in chapter 1.

Moving on, let us shift our attention back to the the study of the actual dataset of temperatures. Without interpolation, we can plot the temperature with respect to scale and angle of incidence so that the actual values of temperature are more clear. Three graphs corresponding to the to the same three intensities are show in the figure 4.9.

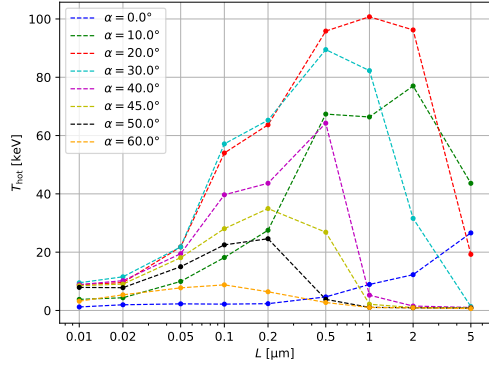(a) Dataset with $I = 1 \times 10^{17} \text{W.cm}^{-2}$.

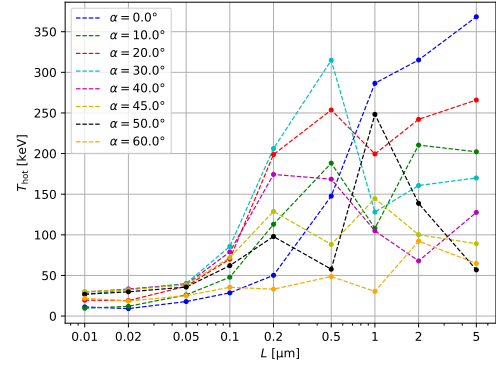(b) Dataset with $I = 1 \times 10^{18} \text{W.cm}^{-2}$.
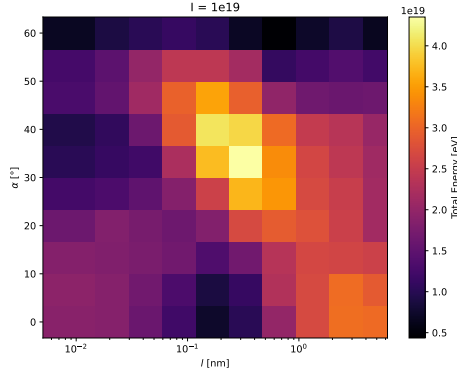
(c) Dataset with $I = 1 \times 10^{19} \text{W.cm}^{-2}$.

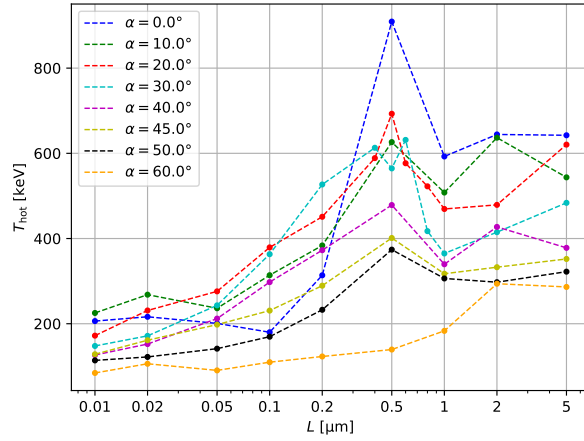Figure 4.8: Illustration of total energy absorption shown for three simulated intensities.

Here, what might be particularly interesting is the evident rise of temperatures for $l = 0.5\mu m$ for the intensity $I = 1 \times 10^{19} \text{W.cm}^{-2}$, especially for small incidence angles. The total absorption is for these parameters relatively small, as can be seen in figure 4.8c. (add explanation ???). For $l = 0.5\mu m$, in both of the smaller intensities the rise is apparent only for larger angles.

(a) Dataset with $I = 1 \times 10^{17} \text{W.cm}^{-2}$.

(b) Dataset with $I = 1 \times 10^{18} \text{W.cm}^{-2}$.

(c) Dataset with $I = 1 \times 10^{19} \text{W.cm}^{-2}$.

Figure 4.9: Illustration of total energy absorption shown for three simulated intensities.

Chapter 5

Hot electron temperature modelling

In this chapter, we will present models of the hot electron absorption trained on the dataset discussed in the previous chapter. The models will be compared and we will also discuss how well the best model represents the behaviour of T_{hot} compared to other studies. We will also present a simple graphing UI tool which allows to load any supported model and plot the predictions in any axis.

We do not implement the models ourselves, but rather we use optimized open-source libraries that offer rich possibilities of configuration. We will compare the models with each other using mostly *RMSE* metric and R^2 statistic. We do not claim that all three models are exhaustingly optimized. We also do not claim that the model we label as best is the only correct solution. The goal of this chapter is to show, how the models work in practice, what are their strengths an weaknesses in this application and what can maybe done in the future to improve them.

We found that the models perform poorly if the data is not transformed. Transforming data is a common step before training a machine learning model. We decided to scale all three parameters to a closed interval [0,1] (L and I logarithmically). If one would skip this step, in SVR and GP, the problem quickly arises because of the absolute value of the distance between two data points needed for the kernel function. For example, if the intensity scale is $10^{17} - 10^{19}$ and scale of angles is $0 - 60$, the dataset is too sparse for these models to learn the relationship because the distance in the intensity dimension is unproportionally bigger. Moreover, in the parameter space, the distance between 10^{17} and 10^{18} is roughly ten times smaller than distance between 10^{18} and 10^{19}. The logarithmic transformation is a justified data manipulation based on the visual inspection and the expected nature of the modelled relationship.

In the actual implementation, the classes which represent all three models follow an abstract structure that allows us to work with different models flexibly. For example, the instance of a transformer class used in the training is saved to a member variable of the model class. In general, the parameters of the transformer can differ for different models. Also, after training using transformation the data have to be

transformed in the same way before the prediction as well.

For each model, we will now provide a brief introduction regarding the implementation. Then we compare the performance of these models using 8-fold cross-validation. In the end, we also compare the models by visual inspection of the prediction graphs sampled on for the same parameters as in the presentation of the dataset.

5.1 Models

For training the SVR model we used *SVR* class from Python *scikit-learn* library. For kernel we selected the RBF kernel. There are multiple hyper-parameters to be optimized - ϵ and C from SVR itself and γ from the RBF kernel which is here defined as $\text{RBF}(\boldsymbol{x}, \boldsymbol{x}') = \exp(\gamma|\boldsymbol{x} - \boldsymbol{x}'|)$. During the cross-validation, where we calculate *RMSE* and R^2 on test set for each fold, the hyper-parameters are selected via grid-search using an additional cross-validation on the training set.

The neural network model was trained using the *PyTorch* library in Python. The architecture is a fully-connected neural network with two hidden layers, each containing 64 neurons with *ELU* activation functions. A dropout rate of 0.2 and a weight decay of 0.1 were applied to compensate the over-parametrization. The model was optimized using the *Adam* optimizer over 2500 epochs with a learning rate of 0.02 minimizing the *MSE* loss function. These hyper-parameters were chosen based on multiple attempts to find a good NN model.

Gaussian processes regression was done using *GPRegression* from *GPy* library also in Python. *GPy* is a very robust and numerically stable tool for Gaussian process regression and classification. We used the *Mattern* kernel defined by 3.15 with $\nu = 3/2$. The hyper-parameters are σ (observation noise), σ_k^2 (kernel variance) and l (kernel scale). The GPy class *GPy.models.GPRegression* has its own optimizer, which can find the optimal hyper-parameters using maximum likelihood approach very effectively.

K-Fold crossvalidation of the models

For all three selected models, we performed an 8-fold cross-validation. This approach allows for a robust and reliable comparison between the models. For each fold, we calculated the root mean square error (RMSE) and the R^2 statistic on the test data. The detailed results for each fold, along with the mean values, are presented in Table 5.1.

As shown in the table, the mean square errors are of the same order for all three models. Some folds tend to produce worse results in general, which is expected. Specifically, it is guaranteed that in one of the folds, the dataset's minimum or maximum value might be missing from the training set. This can result in a higher RMSE due to such an *unlucky* division. This is precisely why we perform 8-fold cross-validation, as on average, these random factors cancel out. The high values of

Table 5.1: Cross-validation of SVR, NN and GP models.

	SVR_{rmse}	SVR_{r2}	NN_{rmse}	NN_{r2}	GP_{rmse}	GP_{r2}
Fold 1	35.13	0.96	30.97	0.97	27.55	0.98
Fold 2	47.61	0.93	40.82	0.95	37.28	0.96
Fold 3	33.18	0.98	31.51	0.98	23.94	0.99
Fold 4	79.91	0.90	43.35	0.97	40.80	0.97
Fold 5	36.95	0.97	30.09	0.98	31.05	0.98
Fold 6	69.68	0.90	43.04	0.96	37.00	0.97
Fold 7	41.01	0.95	37.71	0.96	33.78	0.97
Fold 8	60.28	0.94	41.18	0.97	38.96	0.97
Mean	50.47		37.33		33.80	

R^2 hint that the models are able to capture the modeled relationship at least roughly. The SVR model demonstrates the worst predictive performance. In contrast, the NN and GP models exhibit similar results, though the NN is better only in fold 5. These two models will be compared in greater detail in the following sections.

Neural Network model

The predictions of NN model trained with the same technique as before but now on the whole dataset can be seen in the figure 5.1. The parameter space is the same as in previous chapter when we were presenting the dataset. Now we increased the density by which the model is sampled to grid 50 by 50 (also logarithmic in L axis).

With the NN model, achieving smoothness is quite challenging because the fitting process does not explicitly account for the relationship between two data points. Regularization techniques such as weight decay and dropout do provide some improvement, though likely not as much as desired. The presence of sharp lines or ridges in the graphs indicates that the model does not generalize the relationships well enough. While further regularization could potentially increase smoothness, the results seen in the prediction graphs are probably expected given the small size of the dataset.

Despite these issues, the maxima appear to be in the correct parameter regions, and the maximum hot electron temperature is reasonably consistent with the values present in the dataset. However, it is important to note that the neural network can produce negative predictions, which do not have any physical interpretation. To address this, we decided to convert all negative predictions to zero.

Generally, it is possible to claim that the over-parametrized model can be smoothened to acceptable condition. However, it is still not completely clear, whether this architecture can capture the true physical relationship or not. The ridges, which are visible for predictions with $I = 10^{11}\,\mathrm{W.cm^{-2}}$ suggest either that we need more data, or that the total number of neurons is a little low.
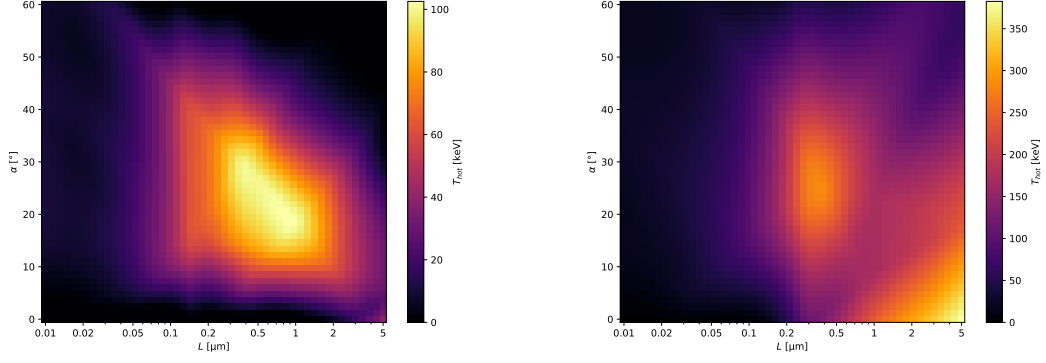
(a) Predictions of NN model for $I = 1 \times 10^{17}$ W.cm^{-2}.

(b) Predictions of NN model for $I = 1 \times 10^{18}$ W.cm^{-2}.
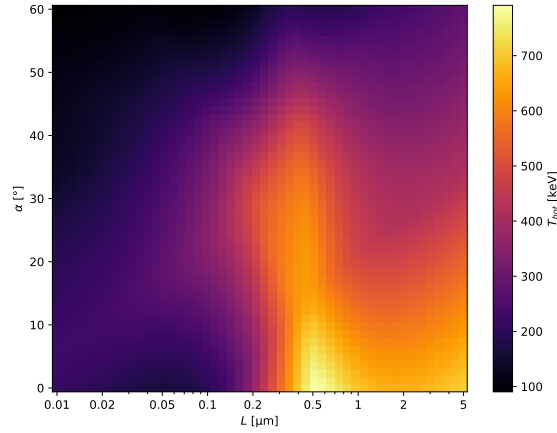
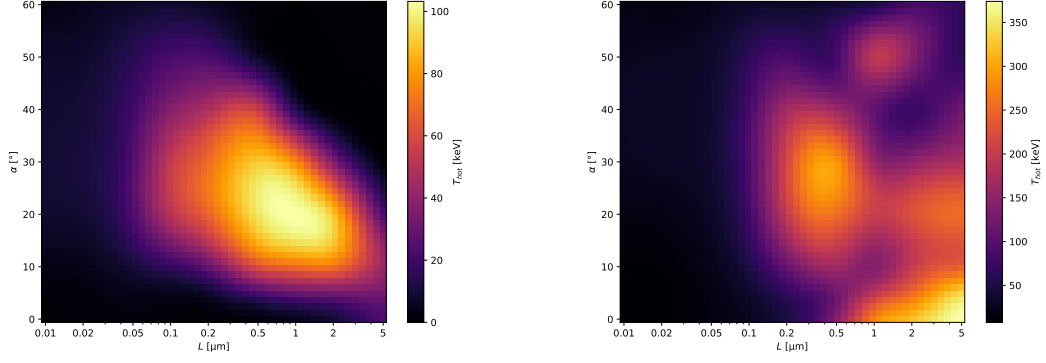

(c) Predictions of NN model for $I = 1 \times 10^{19}$ W.cm^{-2}.
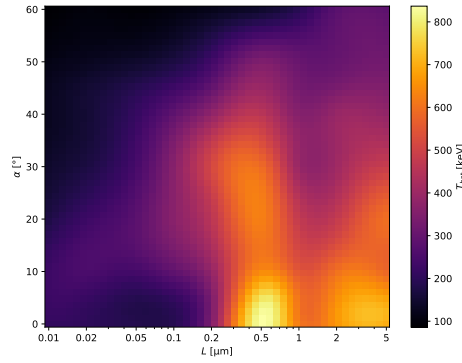
Figure 5.1: Predictions of NN model.

Gaussian process regression model

The GP regression model has shown the best performance on the cross-validation test. Let us remind, that once the optimal parameters of the kernel are found, the predictions are calculated from the posterior distribution as it was presented in section 3.4. The kernel parameters optimized on the whole dataset are $\sigma_k^2 = 29179.49$, $l = 0.36$ and $\sigma^2 = 665.06$. The predictions shown in similar way as for NN model can be seen in figure 5.2.

Immediately one can see, that these predictions are much smoother than those of NN model although in general, they are very similar. Probably the biggest qualitative difference can be found in the predictions for $I = 1 \times 10^{19}$ W.cm^{-2}, where the maximum temperature is lower for NN, whereas in GP the peak has almost exactly the same value as the dataset. Also, in graph for $I = 1 \times 10^{18}$ W.cm^{-2}, there is a local maximum for $\alpha = 50°$ and $L \approx 1\,\mu$m, which cannot be observed in the predictions of NN model.

(a) Predictions of GP model for $I = 1 \times 10^{17}\,\mathrm{W.cm^{-2}}$.

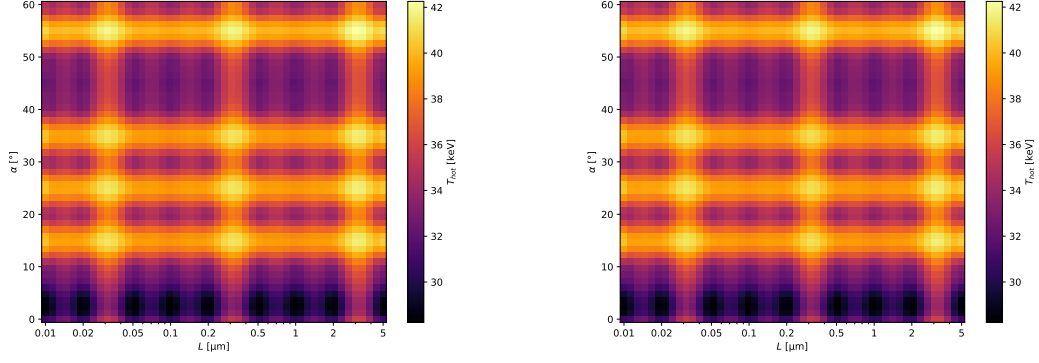(b) Predictions of GP model for $I = 1 \times 10^{18}\,\mathrm{W.cm^{-2}}$.

(c) Predictions of GP model for $I = 1 \times 10^{19}\,\mathrm{W.cm^{-2}}$.
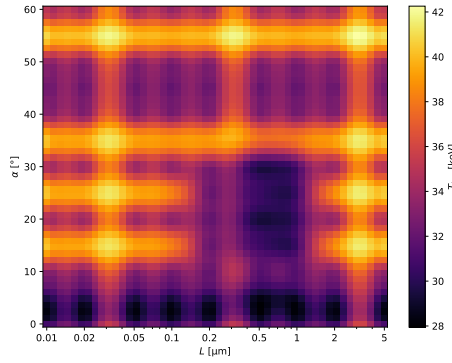
Figure 5.2: Predictions of the GP model.

Now let us come back to one of the biggest strengths of the GP model. As it was stated in the section 3.4, the GP model consists of mean function and covariance function. The predictions in 5.2 are the mean evaluated in given points. The variance of these points is calculated as diagonal elements of a matrix corresponding to the covariance function in the testing points. These values are scaled with optimized parameter of GP - noise variance σ^2. The interpretation of this parameter is therefore straightforward.

It is possible to create same plots but for the standard error σ. Such plots are shown in the figure 5.3. Naturally, the uncertainty is smaller where the training data is denser and larger in areas with sparse data. The denser the dataset, the smaller the error. The dataset was primarily created using a grid distribution of the simulation parameters. The plots hint that the dataset is denser for small angles (up to 10°), which is true. Additionally, there is a higher density of data for the intensity $I = 10^{17}\,\mathrm{W.cm^{-2}}$ around $L = 0.5\mu m$ and for α in range of 15° to 30°, as confirmed by the graphs.

(a) Uncertainty of GP model for $I = 1 \times 10^{17}$ W.cm^{-2}.

(b) Uncertainty of GP model for $I = 1 \times 10^{18}$ W.cm^{-2}.

(c) Uncertainty of GP model for $I = 1 \times 10^{19}$ W.cm^{-2}.

Figure 5.3: Uncertainty of the GP model.

5.2 Prediction UI tool

For the purpose of visual comparison of the models, another tool was developed. The main goal is to allow user to be able to look at the predictions of different models in different axes than those which we used until now. Plotting of custom looks can be tedious if one is writing each time a new script. Also, working with 4 dimensional data is unpleasant, because it is not possible to draw everything in one graph without sacrificing readability.

The tool provides an user interface for several basic options that are relevant for this analysis and a graph of the predictions based on the configuration. A screenshot from this application is shown in the figure 5.4.

The tool is quite simple, but it is tailored for this application, so it almost exhaustingly fulfils the need for special tool. It might be useful even beyond the scope of this thesis for example if someone decides to improve the models we presented or if he just wants to inspect them more closely.

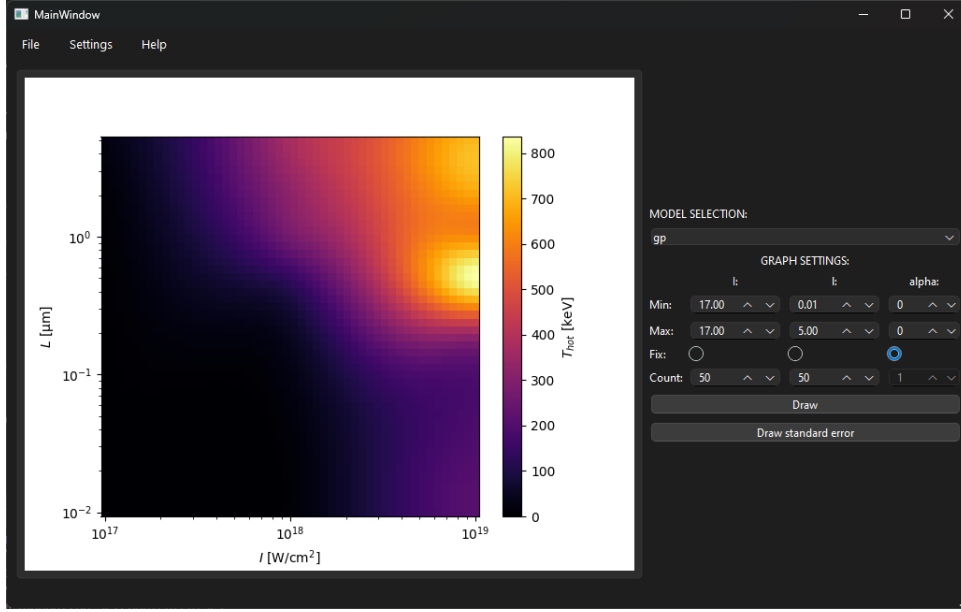The application is implemented in Python in *PyQt6* framework with the additional

Figure 5.4: A screenshot from the graphing tool with graph of GP model for $\alpha = 0°$.

use of class *FigureCanvasQTAgg* from *matplotlib* library. The core of the functionality is relying on an abstraction, where the models are represented as a class with particular members - functions *load* and *predict* and already mentioned member variable of custom type *Transformer*. There are also few more classes which make it easier to work with the prediction grid and the graphing canvas. Adding a support for more, completely different models is only a matter few lines of code as long s the abstraction is followed. Currently only NN, SVR and GP are supported.
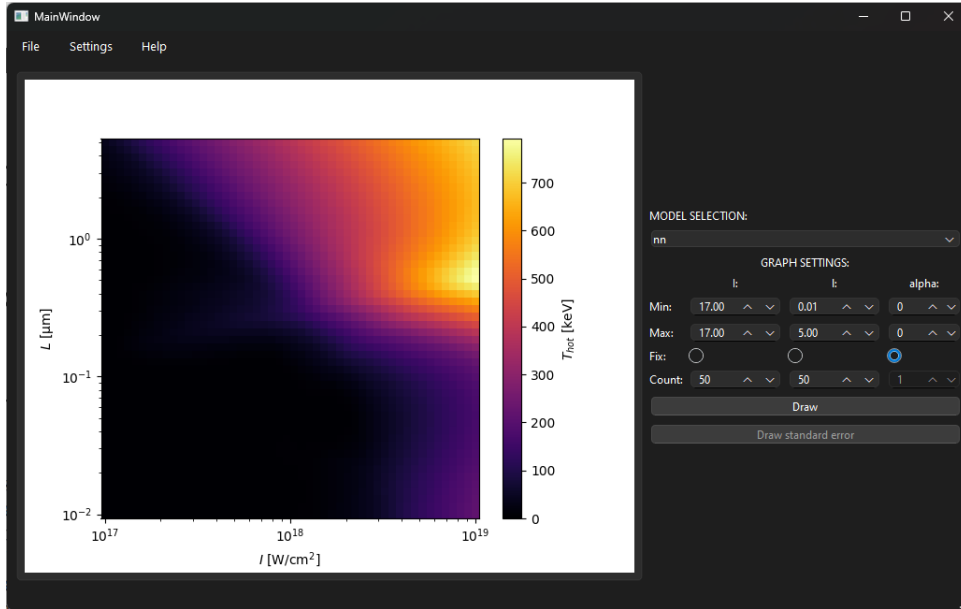

Figure 5.5: A screenshot from the graphing tool with graph of NN model for $\alpha = 0°$.

After choosing a model, the user can choose, which axis he wants to *fix* and at which value. Before, we were always fixing the intensity to three values and sampled L and

α. Then, the user chooses the ranges (min, max) of the remaining two axes alongside with the density of the sample ($count$). Prediction of the same configuration as before but for NN model is shown in the figure 5.5. In both examples, notice the smooth transition between the intensities. Remember that there are almost no training point off the three intensities, so the predicted temperatures could not be very accurate.

Drawing of the error is currently supported only for the GP model. By clicking the *Draw standard error* button the prediction standard error is drawn. The screenshot with standard error can be seen in figure 5.6.
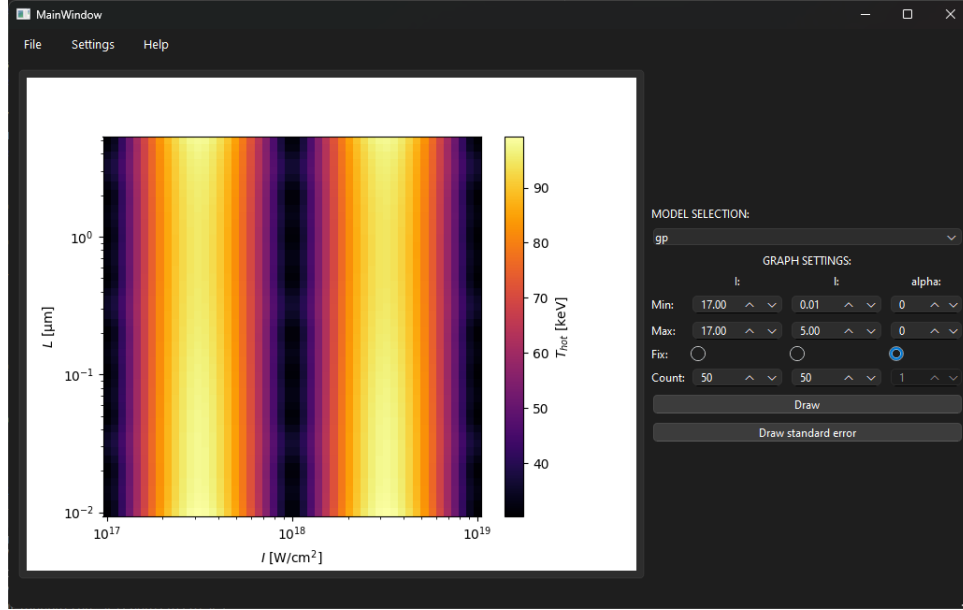

Figure 5.6: A screenshot from the graphing tool with graph of standard error of GP model for $\alpha = 0°$.

After previous discussion, it shouldn't surprise us that the uncertainty is larger for intensities that are not a power of 10. Similar error pattern is visible when L is the fixed axis, instead of α.

5.3   Comparison to other temperature scaling

In this last section, the models are compared to a few of the previous works which studied the scaling of hot electron temperature.

First, the comparison will be done with respect to work of *Cui et al.*[10], where they model scaling of hot electron temperature with intensity as:

$$T_{\text{hot}} = \left(\frac{I\lambda^2}{10^{18}\, W\, \text{cm}^{-2}\, \mu\text{m}^2} \right)^{1/3} \times 1.01\, \text{MeV}, \tag{5.1}$$

which, as they have shown, approximately holds for the optimized scale length and angle of resonance absorption ($\lambda = 1$ and $\alpha = 21.6$) given by 1.19.

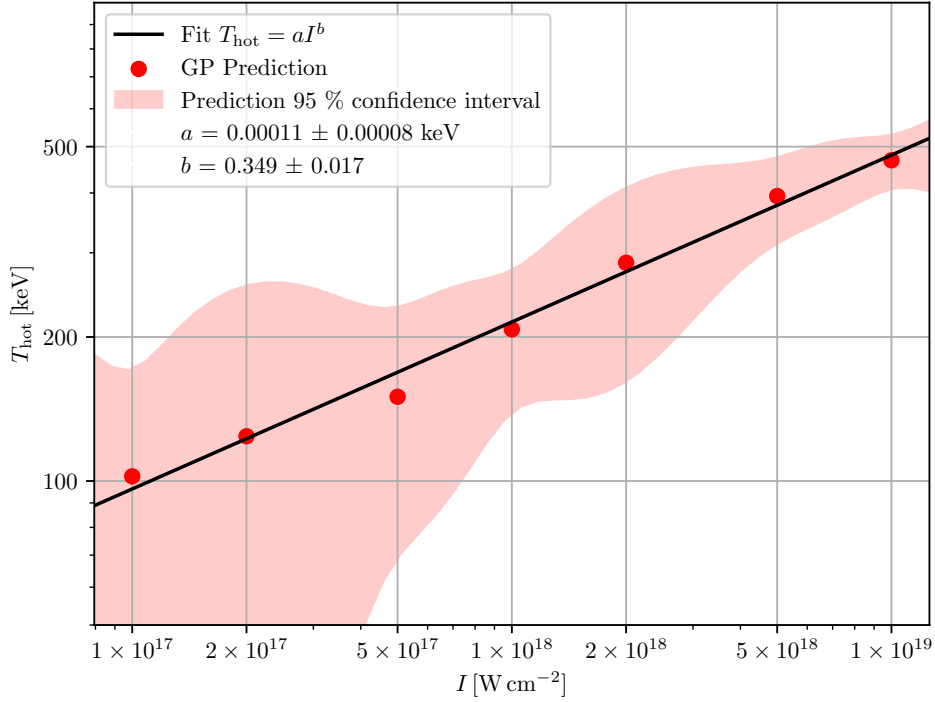The GP model trained by us gives predictions that can be seen in the figure 5.7.

Figure 5.7: The predictions of GP model for optimized resonance absorption $\lambda = 1$ and $\alpha = 21.6°$. The equation of the fit as well as the fitted parameters is shown in the legend. The red area depicts the 95% confidence interval of the predictions.

After fitting the predictions of the GP model with a function $T_{\mathrm{hot}} = aI^b$, we get the estimates $a = 0.00012 \pm 0.00008\,\mathrm{keV}$ and $b = 0.349 \pm 0.017$. The fitted curve is a straight line in the graph because of the choice of scales.

For numerical comparison, the coefficient a has to be scaled with $a^* = a \times 10^6$ because of the unit conversion used bu Cui et al.. This gives us $a^* = 120 keV$ which is still roughly 10 times larger than the constant 1.01 MeV from equation 5.1. The difference is probably caused by the fact that in [10], they used hot electron temperatures from the time of interaction. As they show themselves, the temperature drops quite rapidly with time, which would explain why our temperatures are lower. Other than that, the exponent estimate $b = 0.349 \pm 0.017$ is within one standard error in a agreement with scaling $T_{\mathrm{hot}} \sim I^{1/3}$.

If the angle of incidence is not satisfying the condition 1.19, the relationship seems to become more complicated. In [10], they looked for power coefficient for $T_{\mathrm{hot}} \approx I^b$ at $\lambda = 1$ and $\alpha = 45°$ and estimated it as $b = 0.64$. They used two datapoints $I = 10^{17}\,W\,\mathrm{cm}^{-2}$ and $I = 10^{19}\,W\,\mathrm{cm}^{-2}$ and for these two points we get similar estimate b. However, if we sample the space more densely, our model predicts non-trivial relationship which cannot be easily fitted with simple power function. The predictions together with function $T_{\mathrm{hot}} \sim I^{0.64}$ can be seen in figure 5.8.
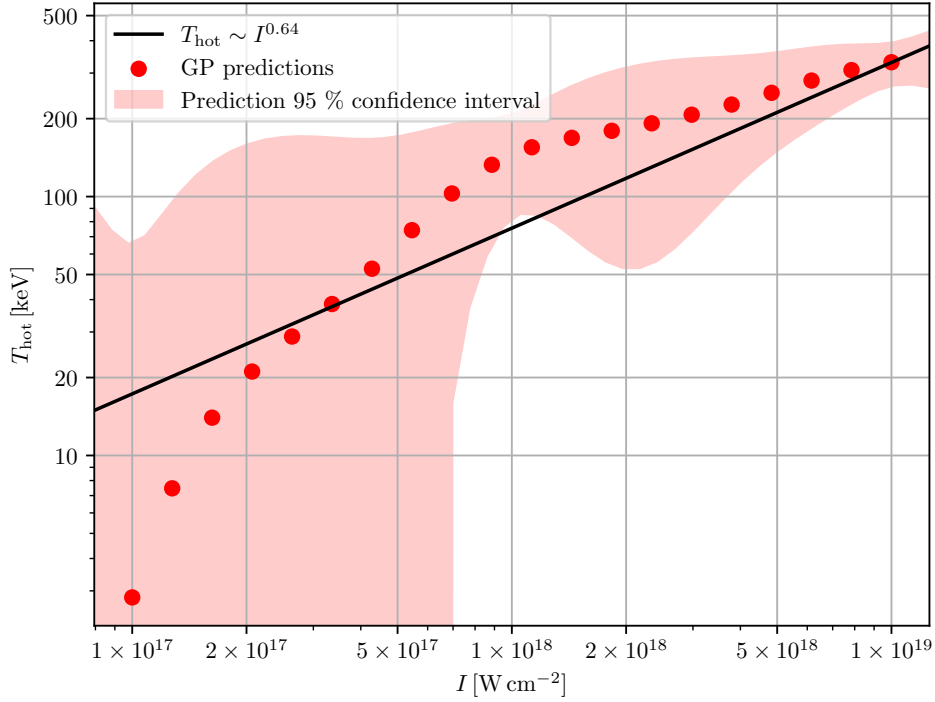
Figure 5.8: The predictions of GP model for $\lambda = 1$ and $\alpha = 45°$. The red area once again shows the 95% confidence interval of the predictions.

One can note that the predictions for the intensity $I = 10^{18}\,\mathrm{W\,cm^{-2}}$ do not match the power rule $T_{\mathrm{hot}} \sim I^{0.64}$. Keep in mind, that the model was mostly trained for three intensities in majority of the grid with few exceptions. It might be interesting that the shape of the predictions in the figure 5.8 much resemble to other graph (figure 2) from [10], which depicts *absorption rate* instead of the temperature.

5.4   Strategy for choosing the next simulations

This section is dedicated to the last unresolved issue of this thesis. We need a strategy for expanding the dataset with the goal of improving the model accuracy. The inherent capability of GP models to provide not only predictions but also an estimate of the uncertainty associated with those predictions makes them well-suited for this task.

The first step in this strategy is to examine the GP model's uncertainty estimates to identify regions with high predictive uncertainty. These regions are typically visualized using standard error plots. Areas with larger uncertainties are prime candidates for further exploration, as additional data from these regions can significantly reduce the overall uncertainty of the model. By focusing on these high-uncertainty regions, we can ensure that resources are used efficiently, targeting the most informative parts of the parameter space.

Once high-uncertainty regions have been identified, the next step is to select specific input parameters within these regions for new simulations. Naturally, resulting data is used to update the GP model. This iterative process is repeated, gradually refining the model and reducing uncertainty across the parameter space.

One of the major strengths of this strategy is its efficient use of resources. By focusing on areas of high uncertainty, we ensure that each new experiment or simulation provides high information gain. Additionally, the GP model's uncertainty estimates are dynamic, allowing for real-time adjustments in the exploration strategy as new data becomes available.

However, there are also some weaknesses to this strategy. The effectiveness of the approach heavily depends on the initial GP model's ability to reasonably approximate the underlying data distribution. If the initial model is poorly constructed, the uncertainty estimates may be misleading. Additionally, if the original model was constructed using parameters selected from grid, it is likely that the uncertainties will be highest in points from which the training inputs are furthest.

When comparing Figure 5.6 to Figure 5.3, one can see that the uncertainties are much larger for the stripes of unexplored intensities. We performed six simulations with parameters selected from the region of high uncertainties to demonstrate how this strategy works in practice. Results of these simulations can be seen in the table 5.2.

Table 5.2: Parameters of new simulations.

$I\,[W\,\mathrm{cm}^{-2}]$	$L\,[\mu\mathrm{m}]$	$\alpha\,[°]$	$T_{\mathrm{hot}}\,[\mathrm{keV}]$	$\Delta T_{\mathrm{hot}}\,[\mathrm{keV}]$
5×10^{17}	0.01	0	3.03	0.11
5×10^{17}	1.00	0	80.55	15.89
5×10^{17}	5.00	0	212.01	15.64
5×10^{18}	0.01	0	85.86	1.87
5×10^{18}	1.00	0	416.01	3.78
5×10^{18}	5.00	0	393.06	5.28

Whether adding new data points improved the model cannot be known using the same K-fold method as we used when choosing the best model. Imagine we added points in regions with the biggest data density. The model describes these regions well even without the new information and therefore the mean test error across the fold would likely be smaller.

In principle, our strategy does the opposite. It prefers the regions with lower data density because the uncertainty is dependant on how far the neighbouring points are (in the transformed parameter space). If we perform the same k-fold with the expanded dataset as in the beginning of this chapter, the models generally produce *bigger* values of mean RMSE than before. This does not mean that a smaller dataset is better than a bigger one. It probably means that the initial dataset was either small or was not chosen optimally, which only motivates us more to explore these regions. It is possible that once the dataset is dense enough everywhere, the mean RMSE from the K-fold starts to decrease with new data.
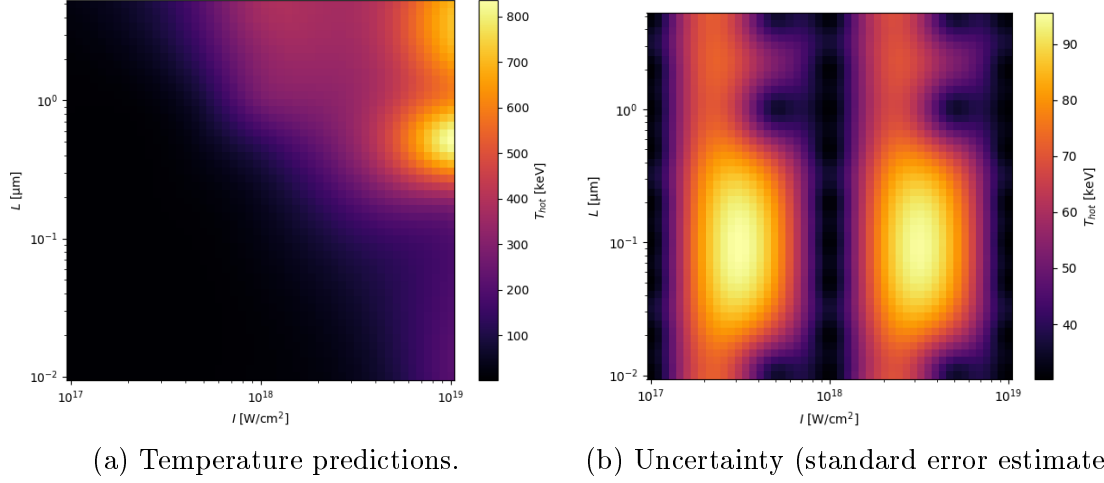
(a) Temperature predictions.　　(b) Uncertainty (standard error estimate).

Figure 5.9: Predictinos of the GP model for $\alpha = 0$ trained with the expanded dataset.

In Figure 5.9 one can see the predictions of the updated GP model. The predictions have slightly changed compared to Figures 5.5 and 5.6. Both predictions and the uncertainties are lower where the new data is. The fact that the temperatures were for some parameters overestimated might have contributed to bigger values of RMSE in the K-fold. We would need much more new simulations to reduce the uncertainty globally but let us have two final remarks to the dataset expansion.

This strategy is very similar to the intuitive approach where we add point to regions with few points. However, neither this nor our strategy is including the information about the shape of the predictions. If the number of new simulations is small, one should select the regions that are particularly interesting to him after examining the predictions visually. The idea is, that where the temperature is changing more steeply, the smoothening might not be as useful as elsewhere. The error introduced by too strict smoothening can be reduced by new data in steep regions.

Let us not forget, that all of the models we used are regression models and they can sometimes treat the maximum values of dataset to be outliers. This can also be compensated for example with improving the dataset density in regions with highest predictions.

Conclusion

This thesis developed a comprehensive approach to modeling laser absorption in plasma, aimed at optimizing the production of K_α X-ray pulses from very small sources, with pulse durations similar to that of the laser (30 fs).

We gathered data for the model by running around 400 particle-in-cell simulations. Initially, we introduced a method for automatically processing these results using a multi-step algorithm to fit sums of exponential distributions. Creating a reliable method for fitting many energy spectra without supervision presents some challenges. We discussed the limitations of our approach, highlighting that finding a metric to describe the quality of the hot electron temperature estimate is not straightforward. This is due to the difficulty in identifying the portion of the electron energy spectrum dominated by hot electrons. We concluded that for larger datasets like ours, further improvements to our method would be beneficial.

The dataset for the model comprised three input parameters from the simulations: laser intensity, the scale length of the exponential preplasma profile, and the angle of incidence. The output was the hot electron temperature. To ensure the dataset's reliability, each simulation result was inspected and fitted separately using a specialized tool developed for this purpose. This dataset can be used in the future to compare automated fitting methods.

We then presented, tuned, and trained three different models: Support Vector Regression (SVR), Neural Network (NN), and Gaussian Process (GP) regression. Comparing these models using RMSE and R^2 metrics, we concluded that GP regression offers the best generalization. NN did not perform much worse in the test and with bigger dataset, it is probably still a valid option. Additionally, we developed a tool for visual analysis of the models, essential due to the complexity of visualizing predictions with 3-dimensional input and 1-dimensional output. The tool features a simple framework that can be easily extended to support other models.

The GP model not only estimates the hot electron temperature but also provides an estimate of the standard error. This uncertainty is valuable for expanding the dataset. We presented a strategy that leverages the standard error estimates, revealing that the initial dataset might not have been optimally selected, as adding more points increased the mean training error.

The original dataset was not selected in an equidistant manner, which is difficult to define in this context. During the training process, the inputs were transformed using hyper-parameters. We opted for a cubic transformation, but fine-tuning these

hyper-parameters and appropriately expanding the dataset could enhance the model.

Finally, we compared our model to previous works. While few studies provide exact analytical models of hot electron temperature scaling, we validated the power law $T_{\text{hot}} \sim I^{1/3}$ for the optimized angle of resonance absorption, as investigated by Cui et al. [10]. We also found that deviating from that angle alters the relationship between I and T_{hot}, with the power law no longer holding true. Our model indicates a more complex relationship, especially since relativistic effects become significant at intensities above $I = 10^{18}, W, \text{cm}^{-2}$.

In summary, we demonstrated that the complex physical processes of laser absorption in dense plasma can be simplified into a surrogate model using Gaussian Processes. Significantly expanding the dataset is feasible and would enhance model accuracy, assuming improvements in fitting the hot electron temperature can be achieved.

Bibliography

[1] F. S. Acton, *Numerical Methods That Work.* The Mathematical Association of America, 1990, pp. 245–257, ISBN: 0-88385-450-3.

[2] T. D. Arber *et al.*, "Contemporary particle-in-cell approach to laser-plasma modelling", *Plasma Physics and Controlled Fusion*, vol. 57, 11 Sep. 2015, ISSN: 13616587. DOI: 10.1088/0741-3335/57/11/113001.

[3] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network", *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 44, pp. 525–536, 2 1998. DOI: 10.1109/18.661502.

[4] F. N. Beg *et al.*, "A study of picosecond laser-solid interactions up to $10^{\{19\}}$ w $\{cm\}^{\{-2\}}$", *Physics of Plasmas*, vol. 4, pp. 447–457, 2 Feb. 1997, ISSN: 1070-664X. DOI: 10.1063/1.872103.

[5] C. K. Birdsall and A. B. Langdon, *Plasma physics via computer simulation.* McGraw-Hill, 1985, ISBN: 0070053715.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*, M. Jordan, J. Kleinberg, and B. Scholkopf, Eds. Springer, 2006, ISBN: 0-387-31073-8.

[7] F. Brunel, "Not-so-resonant, resonant absorption", *Physical Review Letters*, vol. 59, pp. 52–55, 1 Jul. 1987, ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.59.52.

[8] H.-b. Cai, W. Yu, S.-p. Zhu, and C.-y. Zheng, "Short-pulse laser absorption via j×b heating in ultrahigh intensity laser plasma interaction", *Physics of Plasmas*, vol. 13, 11 Nov. 2006, ISSN: 1070-664X. DOI: 10.1063/1.2372463.

[9] F. F. Chen, *Introduction to plasma physics and controlled fusion.* Plenum Press, 1984, ISBN: 0306413329.

[10] Y. Q. Cui, W. M. Wang, Z. M. Sheng, Y. T. Li, and J. Zhang, "Laser absorption and hot electron temperature scalings in laser-plasma interactions", *Plasma Physics and Controlled Fusion*, vol. 55, 8 Aug. 2013. DOI: 10.1088/0741-3335/55/8/085008.

[11] J. Dawson, "One-dimensional plasma model", *Physics of Fluids*, vol. 5, pp. 445–459, 4 1962. DOI: 10.1063/1.1706638.

[12] T. Esirkepov, "Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor", *Computer Physics Communications*, vol. 135, pp. 144–153, 2001. DOI: 10.1016/S0010-4655(00)00228-9. [Online]. Available: www.elsevier.nl/locate/cpc.

[13] P. Gibbon, "Introduction to plasma physics", CERN, 2014, pp. 51–65, ISBN: 9789290834250. DOI: `10.5170/CERN-2016-001.51`.

[14] M. G. Haines, M. S. Wei, F. N. Beg, and R. B. Stephens, "Hot-electron temperature and laser-light absorption in fast ignition", *Physical Review Letters*, vol. 102, 4 Jan. 2009, ISSN: 0031-9007. DOI: `10.1103/PhysRevLett.102.045008`.

[15] J. M. Hokanson, *Numerically stable and statistically efficient algorithms for large scale exponential fitting*, Aug. 2013. [Online]. Available: `https://hdl.handle.net/1911/102224`.

[16] K. Holmström and J. Petersson, "A review of the parameter estimation problem of fitting positive exponential sums to empirical data", *Applied Mathematics and Computation*, vol. 126, pp. 31–61, Feb. 2002. DOI: `10.1016/S0096-3003(00)00138-7`. [Online]. Available: `https://doi.org/10.1016/S0096-3003(00)00138-7`.

[17] S. Ingrassia and I. Morlini, "Neural network modeling for small datasets", *Technometrics*, vol. 47, pp. 297–311, 3 Aug. 2005. DOI: `10.1198/004017005000000058`.

[18] J. Jacquelin, *Regressions et equations integrales*, 2014. [Online]. Available: `https://www.scribd.com/doc/14674814/Regressions-et-equations-integrales`.

[19] C. J. Joachain, N. J. Kylstra, and R. M. Potvliege, *Atoms in Intense Laser Fields*. Cambridge University Press, Dec. 2011, ISBN: 9780521793018. DOI: `10.1017/CBO9780511993459`.

[20] T. Kluge, T. Cowan, A. Debus, U. Schramm, K. Zeil, and M. Bussmann, "Electron temperature scaling in laser interaction with solids", *Physical Review Letters*, vol. 107, 20 Nov. 2011, ISSN: 0031-9007. DOI: `10.1103/PhysRevLett.107.205003`.

[21] W. L. Kruer, *The Physics Of Laser Plasma Interactions*. CRC Press, Aug. 2019, ISBN: 9781003003243. DOI: `10.1201/9781003003243`.

[22] P. Lecca, M. Lecca, C. A. Maestri, and M. Scarpa, "Biexponential fitting for noisy data with error propagation", *Mathematical Methods in the Applied Sciences*, vol. 44, pp. 10 154–10 171, 13 Sep. 2021, ISSN: 0170-4214. DOI: `10.1002/mma.7396`.

[23] A. Macchi, *A Superintense Laser-Plasma Interaction Theory Primer*, 1st ed. Springer Dordrecht, 2013, ISBN: 978-94-007-6124-7. DOI: `10.1007/978-94-007-6125-4`. [Online]. Available: `http://www.springer.com/series/8902`.

[24] P. Michel, *Introduction to Laser-Plasma Interactions*. Springer International Publishing, 2023, ISBN: 978-3-031-23423-1. DOI: `10.1007/978-3-031-23424-8`.

[25] K. G. Miller, D. R. Rusby, A. J. Kemp, S. C. Wilks, and W. B. Mori, "Maximizing mev x-ray dose in relativistic laser-solid interactions", *Physical Review Research*, vol. 5, 1 Mar. 2023, ISSN: 2643-1564. DOI: `10.1103/PhysRevResearch.5.L012044`.

[26] P. J. Mohr, D. B. Newell, and B. N. Taylor, "Codata recommended values of the fundamental physical constants: 2014", *Reviews of Modern Physics*, vol. 88, 3 Sep. 2016. DOI: 10.1103/RevModPhys.88.035009.

[27] M. Mokhomo and N. J. Matjelo, "Fitting sum of exponentials model using differential linear regression", *International Research Journal of Engineering and Technology*, 2021, ISSN: 2395-0072. [Online]. Available: www.irjet.net.

[28] T. Pfeifer, C. Spielmann, and G. Gerber, "Femtosecond x-ray science", *Reports on Progress in Physics*, vol. 69, pp. 443–505, 2 Feb. 2006, ISSN: 0034-4885. DOI: 10.1088/0034-4885/69/2/R04.

[29] D. Potts and M. Tasche, "Parameter estimation for exponential sums by approximate prony method", *Signal Processing*, vol. 90, pp. 1631–1642, 5 May 2010. DOI: 10.1016/j.sigpro.2009.11.012.

[30] G. R. B. Prony, "Essai exp´erimental et analytique sur les lois de la dilatabilit´e de fluides ´elastiques et sur celles de la force expansion de la vapeur de l'alcool, 'a diff´erentes temp´eratures.", *Journal de l'Ecole Polytechnique (Paris)*, vol. 1, pp. 24–76, 1795.

[31] C. E. Rasmussen, "Gaussian processes in machine learning", in Springer, 2004, pp. 63–71. DOI: 10.1007/978-3-540-28650-9_4.

[32] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005, ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001.

[33] C. Reich, P. Gibbon, I. Uschmann, and E. Förster, "Yield optimization and time structure of femtosecond laser plasma k-alpha sources", *Physical Review Letters*, vol. 84, pp. 4846–4849, 21 May 2000, ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.84.4846.

[34] F. Ricardo, "Computational challenges in laser-plasma interactions", in D. A. Jaroszynski, R. Bingham, and R. Cairns, Eds. CRC Press, Mar. 2009, ISBN: 9780429146954. DOI: 10.1201/9781584887799.

[35] H. Schwoerer, "Generation of x-rays by intense femtosecond lasers", in Sep. 2004, pp. 235–254. DOI: 10.1007/978-3-540-39848-6_17.

[36] Z.-M. Sheng *et al.*, "Absorption of ultrashort intense lasers in laser–solid interactions", *Chinese Physics B*, vol. 24, 1 Jan. 2015, ISSN: 1674-1056. DOI: 10.1088/1674-1056/24/1/015201.

[37] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression", *Statistics and Computing*, vol. 14, pp. 199–222, 3 Aug. 2004, ISSN: 0960-3174. DOI: 10.1023/B:STCO.0000035301.49549.88.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 1 2014.

[39] H. Takabe, *The Physics of Laser Plasmas and Applications - Volume 1*. Springer, 2020. [Online]. Available: http://www.springer.com/series/15614.

[40] The, GPY, and Authors, {*Gpy*}: *A gaussian process framework in python*, 2014. [Online]. Available: http://github.com/SheffieldML/GPy.

[41] D. Tskhakaya, K. Matyash, R. Schneider, and F. Taccogna, "The particle-in-cell method", *Contributions to Plasma Physics*, vol. 47, pp. 563–594, 8-9 2007. DOI: 10.1002/ctpp.200710072.

[42] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer New York, 2000, ISBN: 978-1-4419-3160-3. DOI: 10.1007/978-1-4757-3264-1.

[43] J. Wang *et al.*, "Three-dimensional parallel unipic-3d code for simulations of high-power microwave devices", *Physics of Plasmas*, vol. 17, 7 Jul. 2010, ISSN: 1070664X. DOI: 10.1063/1.3454766.

[44] S. Wilks and W. Kruer, "Absorption of ultrashort, ultra-intense laser light by solids and overdense plasmas", *IEEE Journal of Quantum Electronics*, vol. 33, pp. 1954–1968, 11 1997, ISSN: 00189197. DOI: 10.1109/3.641310.

[45] W. J. Wiscombe and J. W. Evans, "Exponential-sum fitting of radiative transmission functions", *JOURNAL OF COMPUTATIONAL PHYSICS*, vol. 24, pp. 416–444, 4 1977. DOI: 10.1016/0021-9991(77)90031-6.

[46] V. Yanovsky *et al.*, "Ultra-high intensity- 300-tw laser at 0.1 hz repetition rate", *Optics Express*, vol. 16, p. 2109, 3 2008, ISSN: 1094-4087. DOI: 10.1364/OE.16.002109.

[47] K. S. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media", *IEEE Transactions on Antennas and Propagation*, vol. 14, pp. 302–307, 3 May 1966. DOI: 10.1109/TAP.1966.1138693.

[48] F. Zhang and L. J. O'Donnell, "Support vector regression", in Elsevier, 2020, pp. 123–140. DOI: 10.1016/B978-0-12-815739-8.00007-9.

Appendix A

EPOCH input file

```
begin:constant
konstanta1 = 1e19
konstanta2 = 2.0
konstanta3 = 60

las_lambda     = 0.8 * micron
ll             = las_lambda
las_intensity_focus_w_cm2 = konstanta1
a0             = ll / micron * sqrt(las_intensity_focus_w_cm2/1.37e18)
gamma0         = sqrt(1.0+a0^2)
pulse_length   = 30.0 * femto
spot_size      = 3.2 * micron
cpwl           = 30
simtime        = 800.0 * femto
theta          = konstanta3 / 180.0 * pi
ct             = cos(theta)
st             = sin(theta)
n_el_over_nc   = 3.0*gamma0
min_density    = 0.01
preplasma_length = konstanta2 * micron
foil_thickness = 4.0 * micron
profile_thickness = - loge(min_density) * preplasma_length
boundary_thickness = 1.0 * micron
x1 = 60 * micron - foil_thickness -
    profile_thickness - boundary_thickness
x2 = x1 + profile_thickness
x3 = x2 + foil_thickness
xfocus         = x2
yc             = - xfocus * tan(theta) * 0.5
sigma_to_fwhm  = 2.0 * sqrt(loge(2.0))
w0             = spot_size / sigma_to_fwhm
zR             = pi * w0^2 / ll
lfocus         = xfocus / ct
lfzr           = lfocus / zR
wl             = w0 * sqrt(1.0+lfzr^2)
intensity_fac2d = 1.0 / sqrt(1.0+lfzr^2) * ct
las_omega      = 2.0 * pi* 299792458.0 / ll
las_time       = 2.0 * pi / las_omega
wt             = pulse_length / sigma_to_fwhm
```

```
n_crit          = critical(las_omega)
n_el            = n_el_over_nc * n_crit
y_full          = 54.0 * micron
end:constant

begin:control
x_min = 0.0 * micron
x_max = 60.0 * micron
y_min = -55.0 * micron
y_max = 55.0 * micron
nx = (x_max-x_min) / ll * cpwl
ny = (y_max-y_min) / ll * cpwl
t_end = simtime
particle_tstart = x1 / ct / 299792458.0
dt_multiplier = 0.97
dlb_threshold = 0.5
use_random_seed = T
stdout_frequency = 10
smooth_currents = T
field_order = 2
end:control

begin:boundaries
cpml_thickness = 6
cpml_kappa_max = 20
cpml_a_max = 0.15
cpml_sigma_max = 0.7
bc_x_min = cpml_laser
bc_x_max_field = cpml_outflow
bc_y_min_field = periodic
bc_y_max_field = periodic
bc_x_min_particle = heat_bath
bc_x_max_particle = heat_bath
bc_y_min_particle = heat_bath
bc_y_max_particle = heat_bath
end:boundaries

begin:laser
boundary = x_min
intensity_w_cm2 = las_intensity_focus_w_cm2 * intensity_fac2d
lambda = ll
t_profile = gauss(time, pulse_length, wt)
t_end = 3.0 * pulse_length
phase = -2.0*pi*(y-yc)*st/ll+((pi/ll)*(((y-yc)*ct)^2))/
    (lfocus*(1.0+(1.0/lfzr)^2))-atan((y-yc)*st/zR+lfzr)
profile = gauss((y-yc)*ct,0.0,wl)
end:laser

begin:species
name = electron
charge = -1.0
mass = 1.0
dump = T
npart_per_cell = 30
density = if((x gt x1) and (x lt x2) and (abs(y) lt y_full),
    n_el*exp((x-x2)/preplasma_length), 0.0)
```

```
density = if((x gt x2) and (x lt x3) and (abs(y) lt y_full), n_el,
    density(electron))
temp_ev = 100.0
identify:electron
end:species

begin:species
name = proton
charge = 1.0
mass = 1836.0
dump = T
npart_per_cell = 30
density = density(electron)
temp_ev = 100.0
end:species

begin:probe
name = electron_back_probe
point = (x3-2.0*micron, 0.0)
normal = (1.0, 0.0)
ek_min = 1.602e-16
ek_max = -1.0
include_species:electron
dumpmask = always
end:probe

#begin:output
#   dt_snapshot = 10.0*femto
#   name = fields
#   file_prefix = f
#   restartable = F
#   grid = always + single
#   ex = always + single
#   ey = always + single
#   bz = always + single
#end:output

#begin:output
#   dt_snapshot = 10.0*femto
#   name = plasma
#   file_prefix = n
#   restartable = F
#   grid = always
#   number_density = always + single
#   poynt_flux = always + single
#   absorption = always
#   total_energy_sum = always
#end:output

begin:output
dt_snapshot = 700.0*femto
name = prob
file_prefix = ppr
restartable = F
particle_probes = always
end:output
```

Appendix B

GP posterior distribution example in Python

```python
import numpy as np
import matplotlib.pyplot as plt

def k(a, b):
    return np.exp(-0.5 * np.subtract.outer(a, b)**2)

x = np.linspace(0, 10, 100)
x_train = np.array([1, 3, 5, 7, 9])
y_train = np.array([2, 2, 3, 4, 2])

K = k(x, x)
K_train = k(x_train, x_train) + 1e-6 * np.eye(len(x_train))
K_train_inv = np.linalg.inv(K_train)

K_x = k(x, x_train)

mu_post = K_x @ K_train_inv @ y_train
cov_post = K - K_x @ K_train_inv @ K_x.T

samples = np.random.multivariate_normal(mu_post, cov_post, 5)
std_dev = np.sqrt(np.diag(cov_post))

plt.scatter(x_train, y_train, color='black', zorder=5)
plt.plot(x, samples.T, zorder=1)
plt.fill_between(x, mu_post - std_dev,
                 mu_post + std_dev, color='gray', alpha=0.3, zorder=2)
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```