BACHELOR

Numerical simulations of heating mechanisms in ultracold plasmas

Sebregts, M.M.C.

Award date:
2012

Eindhoven University of Technology
Department of Applied Physics
Coherence and Quantum Technology (CQT)

Numerical Simulations of Heating Mechanisms in Ultracold Plasmas

Maarten Sebregts

CQT 2012-05

Supervisors:

ir. P.W. Smorenburg
prof. dr. ir. O.J. Luiten

Abstract

Numerical simulations were performed to study heating mechanisms in ultracold plasmas. Both disorder-induced heating and heating caused by three-body recombination are looked at for a plasma with zero initial temperature. The disorder-induced heating gives the expected result: rapid heating to a coupling of $\Gamma \approx 1$. Also the conventional theory for three-body recombination is tested. This model describes the simulation results quite well, since it gives the same results within a factor 3.

Another set of simulations applies a weak oscillating electric field to a weakly coupled plasma ($\Gamma \approx 0.1$-0.2). In this situation, the conventional theory for collisional heating is tested. It is found that this theory predicts the observed heating really well in this situation.

Contents

1 Introduction **1**

2 Theory **2**

2.1 Ultracold plasmas 2

2.2 Plasma parameters 2

2.2.1 Plasma frequency 2

2.2.2 Coupling parameter 2

2.2.3 Collision frequency 3

2.3 Plasma without an external oscillating field 4

2.3.1 Scaled equations of motion 4

2.3.2 Disorder induced heating 4

2.3.3 Three-body recombination 5

2.4 Plasma in an external oscillating field 6

2.4.1 Scaled equations of motion 6

2.4.2 Single electron 6

2.4.3 Neutral plasma motion 6

2.4.4 Collisional heating 7

3 Simulation **8**

3.1 Simulation program 8

3.2 Simulation set-up 8

3.3 Accuracy of the Runge-Kutta solver 9

3.4 Approximate Coulomb potential 10

3.5 Barnes-Hut approximation 10

4 Results and discussion **11**

4.1 Plasma without an external field 11

4.1.1 Independence of input parameters 11

4.1.2 Averaged result 11
4.1.3 Disorder induced heating 13
4.1.4 Three-body recombination 13
4.2 Plasma in an external field 14
4.2.1 Variation of simulation parameters 15

5 Conclusion and outlook **19**

References **20**

Appendix A Simulation parameters **21**

Appendix B Simulation code **22**

B.1 GPT input files 22
B.2 GPT custom elements 24
B.3 GDFA custom programs 26
B.4 Matlab scripts 32

1 Introduction

Ultracold Plasmas (UCPs) have many interesting properties. As the name implies, they have an extremely low temperature compared to traditional plasmas. This low temperature is useful for creating cold ion or electron beams, which have the advantage over 'hot' beams that they can be focused into smaller spots. Ultracold beams are usually created by placing an acceleration structure around the UCP set-up. This will extract the ions on one side and the electrons on the other, which can be accelerated to create the beams.

Another idea to accelerate the plasma is using an electromagnetic wave. This wave will cause the electrons inside the plasma to oscillate, whereas the ions will be approximately stationary since their mass is much larger. If the wavelength is larger than the size of the plasma, all electrons will oscillate coherently. These oscillating electrons will emit dipole radiation, and because the radiated energy is extracted from the incoming wave, the wave will lose momentum. Due to the law of conservation of momentum, the electrons will have to gain momentum. The electrons will then 'pull' the ions with them, which results in the acceleration of the complete plasma.

This acceleration approach requires to still have a cold beam. It is therefore important to understand the heating processes in the plasma. For this purpose, several numerical simulations were performed. The first goal was to understand the heating mechanisms in an UCP when there is no external field. After that an oscillating electric field was applied to investigate its influence on plasma heating.

2 Theory

2.1 Ultracold plasmas

Ultracold Plasmas (or UCPs) differ in a lot of properties with conventional plasmas [1]. First of all their temperature is low: usually less than 100 Kelvin. This means that the electrostatic energy between the particles is much more important than in hot plasmas, since the thermal energy of the electrons is relatively much lower.

Secondly, because UCPs are created by photo-ionizing laser-cooled atom clouds, their size and number of particles is limited. With current techniques, a particle density up to $10^{17}\,\mathrm{m}^{-3}$, and a maximal size of a few millimetres can be reached. With these sizes and densities, boundary effects may be important and may even become dominant over volume effects.

Due to these differences, standard plasma theory for heating may not be valid any more. Except for the disorder-induced heating from section 2.3.2, all theory discussed here is developed for conventional plasmas. Part of the research done was therefore to check if this theory is also valid for an UCP.

2.2 Plasma parameters

2.2.1 Plasma frequency

If the electrons of a plasma are pulled out of their equilibrium position, they will start to oscillate due to the restoring space charge force. For standard plasmas the eigenfrequency of the electrons is the *electron plasma frequency*, ω_p, as derived in [2]:

$$\omega_p = \sqrt{\frac{n_e e^2}{m_e \epsilon_0}}, \tag{1}$$

where n_e is the electron density, e the elementary charge, m_e the electron mass and ϵ_0 the vacuum permittivity. However in a spherical plasma, which is considered here, the eigenfrequency of the plasma is the Mie frequency (ω_m), which differs from the plasma frequency as follows:

$$\omega_m = \frac{\omega_p}{\sqrt{3}}. \tag{2}$$

2.2.2 Coupling parameter

The Coulomb coupling parameter Γ is the ratio between the typical Coulomb energy and the thermal energy of electrons:

$$\Gamma = \frac{U_{Coulomb}}{U_{thermal}} = \frac{e^2}{4\pi\epsilon_0 a}/k_B T_e, \tag{3}$$

where k_B is the Boltzmann constant and T_e the electron temperature. The Wigner-Seitz radius a is the typical inter-particle distance defined as

$$a = \left(\frac{3}{4\pi n_0}\right)^{1/3}, \tag{4}$$

where $n_0 = n_i + n_{e,0}$ is the initial density of the plasma. n_i is the ion density, which will be constant on the time scales that are looked at in this report. n_e is the electron density, which will not be constant because electrons escape from the plasma. $n_{e,0}$ is the initial electron density, which is chosen to be equal to n_i in this report, so $n_0 = 2n_i$.

2.2.3 Collision frequency

In a plasma, a charged particle is at any instant undergoing Coulomb interactions with many other charged particles. Because of these interactions the direction of the particle will slowly change randomly, as if it had collided. The usual way to define the time between two 'collisions' is the average time it takes to change a particle's direction with 90°. The inverse of this time is ν_{eff}, the effective collision frequency. For weakly coupled plasmas with singly-ionized ions, Spitzer has derived the following effective *electron* collision frequency [3]:

$$\nu_{\text{eff}} = \frac{4}{3}\frac{\sqrt{2\pi}\, e^4\, n_i}{(4\pi\epsilon_0)^2\sqrt{m_e}}(k_B T_e)^{-3/2}\, \ln\Lambda. \tag{5}$$

Here n_i is the ion density and $\ln\Lambda = \lambda$ is the Coulomb logarithm. As summarized by Gericke [4], there are multiple ways to define it. The most common definition is:

$$\lambda = \ln\left(\frac{b_{max}}{b_{min}}\right), \tag{6}$$

where b_{max} is equal to the electron Debye length $\lambda_D = \sqrt{\epsilon_0 k_B T_e / e^2 n_e}$ and $b_{min} = \sqrt{\lambda\!\!\!^{-2} + \rho_\perp^2}$ is an interpolation between the de Broglie wavelength $\lambda\!\!\!^{-} = \hbar/2m_e v_{thermal}$ and the classical parameter $\rho_\perp = e^2/(4\pi\epsilon_0 k_B T_e)$. Here $\hbar$ is the reduced Planck constant and $v_{thermal}$ the thermal electron speed: $\frac{1}{2}m_e v_{thermal}^2 = 3k_B T_e/2$. This gives good results for $\lambda > 3$, however for lower values of λ this approach is wrong – λ may even become negative – and the following relation gives a much better result:

$$\lambda = 0.5 \ln\left(1 + \frac{b_{max}^2}{b_{min}^2}\right). \tag{7}$$

In the simulations in this report where the collision frequency was used, the Coulomb logarithm will be a bit larger than 3, so Eq. (6) is used.

Eq. (5) can also be written in terms of Γ. This yields the following result:

$$\nu_{\text{eff}} = \sqrt{\frac{2}{3\pi}}\frac{1}{\sqrt{2}}\omega_{p,0}\Gamma^{3/2}\, \ln\left(\Gamma^{-3/2}\sqrt{\frac{n_0}{3n_e}}\right). \tag{8}$$

As derived by Silin [5], the Spitzer collision frequency is valid in the weak field limit $(eE_0)/(m_e\omega_d) \ll v_{thermal}$, where E_0 is the electric field amplitude and ω_d the oscillation frequency of the electric field.

2.3 Plasma without an external oscillating field

2.3.1 Scaled equations of motion

For a particle i with mass m_i, charge q_i and position $\vec{r}_i$ in a plasma, the equation of motion is:

$$m_i \frac{\mathrm{d}^2 \vec{r}_i}{\mathrm{d}t^2} = \sum_{j \neq i} \frac{1}{4\pi\epsilon_0} \frac{q_i q_j (\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j|^3}. \tag{9}$$

Here, the sum gives the Coulomb forces due to the other particles $j \neq i$. To investigate how the behaviour of the plasma changes for varying parameters, length is scaled by the *initial* Wigner-Seitz radius a_0 and time is scaled by the *initial* plasma frequency $\omega_{p,0}$. Then we get the dimensionless position $\vec{x} = \vec{r}/a_0$ and time $T = \omega_{p,0} t$. In these scaled units, the equation of motion becomes

$$\frac{m_i}{m_e} \frac{\mathrm{d}^2 \vec{x}_i}{\mathrm{d}T^2} = \frac{1}{6} \sum_{j \neq i} \frac{q_i q_j (\vec{x}_i - \vec{x}_j)}{e^2 |\vec{x}_i - \vec{x}_j|^3}, \tag{10}$$

where Eq. (1) and Eq. (4) have been used.

Besides the equations of motion, also the coupling parameter is independent of the density. Using Eq. (3) and $3k_B T_e/2 = m_e \langle v^2 \rangle / 2$, the coupling parameter can also be written as

$$\begin{aligned} \Gamma^{-1} &= \frac{4\pi\epsilon_0 a}{e^2} \frac{1}{3} m_e \langle \left(\frac{\mathrm{d}\vec{r}}{\mathrm{d}t} \right)^2 \rangle \\ &= \frac{1}{2} \langle \left(\frac{\mathrm{d}\vec{x}}{\mathrm{d}T} \right)^2 \rangle. \end{aligned} \tag{11}$$

From this we can conclude that the physics of a plasma without an external field does not depend on the density of the plasma. Since the scaled equations of motion are the same for plasmas of different densities, the resulting behaviour will only differ in the time and length scales.

2.3.2 Disorder induced heating

When charged particles are created at random positions inside a volume with no initial speed ($U_{kin} = 0$), there will be an excess of potential energy (U_{pot}) in the system. Because of the random placement, there is no order in the system. Since the lowest potential energy state is reached in an ordered system (with a crystalline structure) and initially there is no kinetic energy, there is a surplus of potential energy. Because of equipartition, U_{kin} will rise from zero to $U_{kin} \approx U_{pot}$. This effect is called *disorder-induced heating*: the particles will heat due to the lack of order at the beginning.

This effect is also important in UCPs, as described by Killian [1]. In a time scale of the inverse plasma frequency ω_p^{-1}, the electron potential energy is transferred to kinetic energy. From Eq. (3), this leads to a coupling factor of $\Gamma \approx 1$, which has been observed in numerical simulations by Kuzmin and O'Neil [6].

2.3.3 Three-body recombination

Three-body recombination (TBR) is another heating process in plasmas. In this process an electron and an ion recombine, while the excess potential energy is transferred to another electron. Quantum mechanically, the recombined electron will be in a high-Rydberg state. In a classical approximation however, the electron will orbit around the ion.

In 1924, J.J. Thomson proposed a theory for the recombination rate, which is the number of recombinations per unit volume per unit time. The basic idea is as follows. Consider an electron and an ion in a plasma. When the electron is far from the ion, on the average it will have a kinetic energy of $3k_BT/2$. When the electron is attracted to the ion, its kinetic energy will increase at the expense of its potential energy. If the electron experiences a 'thermalizing' collision with another electron close to the ion, its kinetic energy will on the average decrease back to $3k_BT/2$. If this collision occurs within a distance r_0 from the ion, the total relative energy of the electron and the ion can become negative. The electron will then move in a bound trajectory about the ion and recombination has occurred.

The radius r for which a recombination can occur satisfies the inequality [7]

$$r < r_0 = \frac{e^2/4\pi\epsilon_0}{3k_BT/2}. \tag{12}$$

The recombination rate R is obtained by calculating the frequency of thermalizing collisions between two electrons, within a distance $r < r_0$ from an ion. The result of this calculation is [7]

$$R = 2.13 \times 10^{-20} \ \frac{n_i n_e^2}{T_e^{9/2}} \ln\Lambda \quad \mathrm{m^{-3}s^{-1}}, \tag{13}$$

where n_i and n_e are the ion and electron densities and $\ln\Lambda$ is the Coulomb logarithm, see 2.2.3. All units are SI.

The result from this relatively simple model is in good agreement with the result obtained with more rigorous arguments by Hinnov and Hirschberg [8]:

$$R = 1.09 \times 10^{-20} \ \frac{n_i n_e^2}{T_e^{9/2}} \quad \mathrm{m^{-3}s^{-1}}. \tag{14}$$

According to the Thomson model, an electron will have a kinetic energy of $3k_BT_e/2$ before recombination, and approximately no relative potential energy. After the electron has recombined, its kinetic energy will again be $3k_BT_e/2$ and its potential energy will be less than $-3K_BT_e/2$. This means that the energy released to the second electron will be at least $3k_BT_e/2$. The rate of change of total kinetic energy in the plasma will therefore be

$$\frac{\mathrm{d}U_{\mathrm{kin,total}}}{\mathrm{d}t} = RV\frac{3}{2}k_BT_e, \tag{15}$$

where $V = N_i/n_i$ is the volume of the plasma where ions are present. Using the rate from Eq. (14), Eq. (15) becomes:

$$\frac{\mathrm{d}U_{\mathrm{kin,total}}}{\mathrm{d}t} = C\frac{n_i n_e^2}{(k_BT_e)^{9/2}}\frac{N_i}{n_i}\frac{3}{2}k_BT_e, \tag{16}$$

where $C = k_B^{9/2} * 1.09 \times 10^{-20}\,\mathrm{m^6\,s^{-1}\,J^{9/2}}$. Using $U_{\mathrm{kin,total}} = N_e \langle U_{\mathrm{kin}} \rangle$, $N_e = N_i$ and scaling this relation in the same way as done in section 2.3.1, the following equation is obtained:

$$\frac{\mathrm{d}\Gamma^{-1}}{\mathrm{d}T} = C \sqrt{\frac{m_e \epsilon_0}{e^2}} \left(\frac{4\pi\epsilon_0}{e^2}\right)^{9/2} \left(\frac{3}{4\pi}\right)^{3/2} \frac{n_e^2}{2^{3/2} n_i^2} \Gamma^{7/2}. \tag{17}$$

2.4 Plasma in an external oscillating field

2.4.1 Scaled equations of motion

Analogously to section 2.3.1, the equations of motion can also be scaled when there is an external oscillating field $\vec{E}_{ext} = \vec{E}_0 \cos(\omega_d t)$. The equations of motion then become

$$m_i \frac{\mathrm{d}^2 \vec{r}_i}{\mathrm{d}t^2} = \sum_{j \neq i} \frac{1}{4\pi\epsilon_0} \frac{q_i q_j (\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j|^3} + q_i \vec{E}_0 \cos(\omega_d t). \tag{18}$$

If ω_d and $\vec{E}_0$ are substituted by respectively $f_d \omega_p$ and $\frac{e}{4\pi\epsilon_0 a^2} \vec{C}_E$ and the same scaling as in section 2.3.1 is applied, the scaled equation of motion becomes

$$\frac{m_i}{m_e} \frac{\mathrm{d}^2 \vec{x}_i}{\mathrm{d}T^2} = \frac{1}{6} \left[\sum_{j \neq i} \frac{q_i q_j (\vec{x}_i - \vec{x}_j)}{e^2 |\vec{x}_i - \vec{x}_j|^3} + \frac{q_i}{e} \vec{C}_E \cos(f_d T) \right]. \tag{19}$$

This means that also in a plasma with an external field, the scaled equations of motion and hence the behaviour of the plasma does not depend on the density of the plasma. Since the scaled equations of motion are the same for plasmas of different densities, the resulting behaviour will only differ in the time and length scales for the same values of f_d and $\vec{C}_E$.

2.4.2 Single electron

For a single electron in an oscillating electric field $\vec{E} = E_0 \cos(\omega_d t) \vec{e}_x$, the solution of the equation of motion is

$$\vec{r}(t) = \frac{eE_0}{m_e \omega_d^2} \cos(\omega_d t) \vec{e}_x + \vec{v}_0 t + \vec{r}_0. \tag{20}$$

When the electron is initially at rest, the kinetic energy of the electron as a function of the time is

$$U_{kin} = \frac{1}{2} m_e \left| \frac{\mathrm{d}\vec{r}}{\mathrm{d}t} \right|^2 = 2U_p \sin^2(\omega_d t), \tag{21}$$

where $U_p = \frac{e^2 E_0^2}{4 m_e \omega_d^2}$ is the ponderomotive energy.

2.4.3 Neutral plasma motion

A spherical neutral plasma in an oscillating field can be approximated by two spheres of charge: a positively charged sphere of ions and a negatively charged sphere of electrons. Let x be the separation of the centres of mass - and the centres of charge as well

- in the horizontal direction. When a field $\vec{E} = E_0 \cos(\omega_d t)\vec{e}_x$ drives this plasma and under the assumption that the ions are stationary (ion mass $\gg$ electron mass), x will satisfy the following differential equation:

$$\begin{cases} \ddot{x} + \omega_m^2 x = -\frac{eE_0}{m_e} \cos(\omega_d t) \\ x(0) = x_0 \\ \dot{x}(0) = v_0 \end{cases}, \tag{22}$$

where ω_m is the Mie frequency (2).

When this differential equation is solved, the following solution for x is obtained:

$$x = x_0 \cos(\omega_m t) + \frac{v_0}{\omega_m} \sin(\omega_m t) + \frac{eE_0}{m_e \omega_d^2} \frac{1}{1 - \omega_m^2/\omega_d^2} \Big(\cos(\omega_d t) - \cos(\omega_m t) \Big). \tag{23}$$

With this solution, the average electron kinetic energy caused by the electric field can be calculated as well:

$$\begin{aligned} U_{kin} &= \tfrac{1}{2} m_e \dot{x}^2 \\ &= \tfrac{1}{2}\, m_e \Big[-x_0 \omega_m \sin(\omega_m t) + v_0 \cos(\omega_m t) + \\ &\qquad \tfrac{eE_0}{m_e \omega_d^2} \tfrac{1}{1-\omega_m^2/\omega_d^2} \Big(\omega_m \sin(\omega_m t) - \omega_d \sin(\omega_d t) \Big) \Big]^2 \end{aligned} \tag{24}$$

2.4.4 Collisional heating

One of the most important heating mechanisms in plasmas in an external field is collisional heating. As shown in the book of Mulser and Bauer [9], an electron gains energy when colliding with an ion in an oscillating electric field $\vec{E} = E_0 \cos(\omega_d t)\vec{e}_x$. Under the assumption that the thermal velocity is much larger than the quiver velocity caused by the electric field, they show that the mean energy gain in one collision is $2U_p$, where U_p is the ponderomotive energy.

Combining the collision frequency and the energy gain per collision, the energy gain per unit time per electron will be

$$\frac{\mathrm{d}U}{\mathrm{d}t} = \nu \cdot 2U_p = \nu \cdot 2 \frac{e^2 E_0^2}{4 m_e \omega_d^2}. \tag{25}$$

The total energy gain of the plasma per unit time will therefore be

$$\frac{\mathrm{d}U_{total}}{\mathrm{d}t} = 2\nu U_p N_e, \tag{26}$$

where N_e is the number of electrons inside the plasma. Using for ν the Spitzer frequency from Eq. (8) and writing the result in scaled units yields

$$\frac{\mathrm{d}U_{total}}{\mathrm{d}T} = 2\sqrt{\frac{2}{3\pi}} \frac{n_i}{\sqrt{n_{e,0} n_0}} \Gamma^{3/2} \ln\left(\Gamma^{-3/2} \sqrt{\frac{n_0}{3n_e}} \right) \cdot U_p N_e. \tag{27}$$

3 Simulation

3.1 Simulation program

The General Particle Tracer[10], GPT, was used as the simulation program. GPT has the capability to solve the equations of motion for N charged particles simultaneously in the electric field caused by those particles. External electromagnetic fields can also be taken into account, making the program suitable to use in our classical simulations. Effectively GPT solves for particle i the following equation of motion:

$$\frac{d\vec{p}_i}{dt} = q_i \left(\sum_{j \neq i} \frac{q_j(\vec{r}_i - \vec{r}_j)}{4\pi\epsilon_0(|\vec{r}_i - \vec{r}_j|^2 + R^2)^{3/2}} + \vec{E}_{ext} \right), \tag{28}$$

where R is a Coulomb round-off parameter. This parameter is used to avoid singularities in the electric field.

GPT solves these equations of motion using a fifth-order embedded Runge-Kutta method with adaptive step-size. The accuracy of this method can be controlled by the user with an accuracy parameter A. The algorithm assures that for every particle the estimated error in $\gamma\beta$ will be less than 10^{-A}, where $\beta = v/c$ with v the particle's speed and c the speed of light, γ is the Lorentz factor $(1 - \beta^2)^{-1/2}$.

GPT has multiple methods to calculate the electric force on a particle due to all other particles. The two relevant methods are direct calculation and an improved Barnes-Hut algorithm. In direct calculation, the electric field is simply calculated by evaluating the sum of Eq. (28). This method is exact, but the computation time scales with N^2, where N is the number of particles. The Barnes-Hut algorithm [11] is more efficient. The computation time scales with $N \log N$, so it is more suited when having a large number of particles.

The Barnes-Hut algorithm used in GPT calculates the force by grouping particles together. It calculates the electric monopole, dipole and quadrupole moments of all groups and uses these moments to evaluate the electric field caused by the particles. Particles are grouped together when they are sufficiently far away. This depends on the ratio s/d, where s is the size and d the distance of the group. This ratio has to be lower than a threshold value θ. A lower value of θ means that less particles are grouped together and that the calculation is more exact, but it also means a longer computation time.

3.2 Simulation set-up

In all simulations an equal number of ions and electrons are placed inside a sphere, this is done using a uniform random number generator, so the electron distribution in the sphere is approximately uniform. The plasma density is used as an input parameter and this determines the radius of the plasma sphere, such that all electrons and ions fit inside it. Initially all ions are at rest and the velocity components of the electrons have a Gaussian distribution with variance $k_B T_0/m_e$, where T_0 is the initial electron temperature. The mass of the ions is taken to be the mass of a rubidium ion, since that is an often used element to create UCPs.

3.3 Accuracy of the Runge-Kutta solver

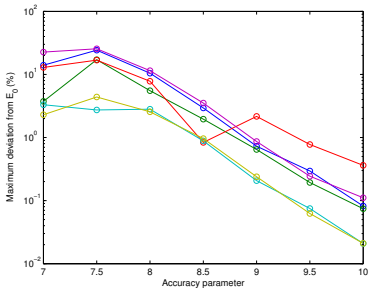
When no external field is applied to the plasma, the total energy of the plasma

$$U_{total} = \sum_i \frac{1}{2} m_i v_i^2 + \frac{1}{2} \sum_i \sum_{j \neq i} \frac{q_i q_j}{4\pi\epsilon_0 |\vec{r}_i - \vec{r}_j|} \tag{29}$$

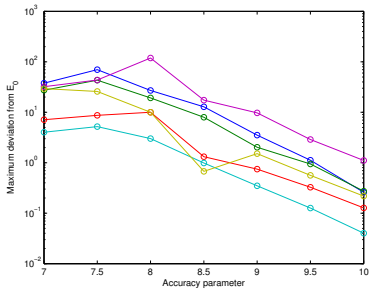
is conserved. Since the Runge-Kutta solver is not symplectic, i.e. the Hamiltonian is not a priori conserved, the total energy in the simulation will *not* be conserved exactly. A good measure for the accuracy of the simulation is therefore the maximum deviation of the total energy during a simulation.

In Fig. 1, the results from several simulations are plotted. The exact simulation parameters used in these simulations can be found in Appendix A. In these simulations, the Coulomb round-off R was extremely small and electron grouping with the Barnes-Hut algorithm was not used, to avoid effects from both of these approximations. As expected, the total energy is much better conserved when using a higher accuracy parameter.

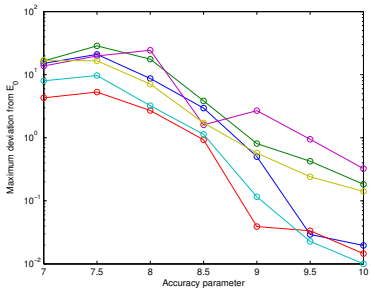
We consider an energy conservation of 1 % to be good enough. Therefore an accuracy parameter of 8.5 is used in further simulations.



(a) 128 electrons



(b) 192 electrons



(c) 256 electrons

Figure 1: Maximum relative deviation of the total energy from the total energy at the start (E_0) as a function of the accuracy parameter, for several numbers of electrons. In each graph, each line represents a different initial distribution, while all other parameters were kept the same.

3.4 Approximate Coulomb potential

From Eq. (28) it is clear that the Coulomb round-off parameter R is not important at large distances. It is included in the simulation to prevent singularities in the electric field. The most physically accurate would be to let $R \to 0$.

A larger R will reduce computation time for close encounters. However, the lesser accuracy only affects the simulation results if encounters closer than R occur frequently. To prevent this, R is taken to be ϵa, with a the Wigner-Seitz radius (4) and $\epsilon \ll 1$.

In the simulations done in the previous section, the minimum distance between two particles was almost never less than $a/100$. To give physically correct simulations, ϵ will be 10^{-3} in all simulations done hereafter.

3.5 Barnes-Hut approximation

From the large variation for different runs in Fig. 1, it was clear that small changes in the initial conditions result in completely different microscopic behaviour, which is a good example of chaos. The microscopic behaviour will probably also be different when using the Barnes-Hut algorithm to calculate the electric fields. However, in this report only the macroscopic behaviour is important. To test the accuracy of the Barnes-Hut approximation, the resulting macroscopic behaviour must be the same. Because the most interesting quantities in the simulation are the energies, the kinetic and total energy must behave the same in a simulation with and without using the Barnes-Hut algorithm.

As can be seen in Fig. 2, the average electron kinetic energy with and without using the Barnes-Hut algorithm differs only less than a few percent. Even for a relatively large θ of 1, the results are not much affected by the approximation. This is no different for the total energy. In both simulations the total energy was conserved to within 1%. Because, in addition, the computation time is considerably lower, the Barnes-Hut algorithm is used with $\theta = 1$ in all following simulations.

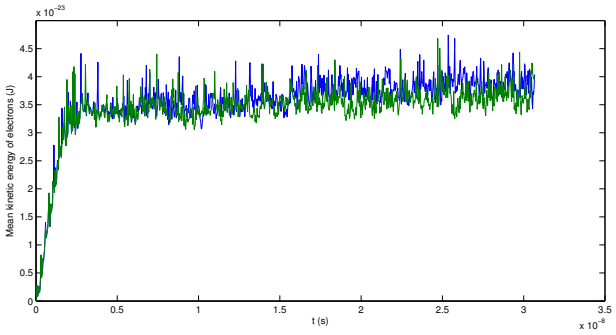


Figure 2: Results from a simulation not using the Barnes-Hut algorithm (blue line) and another simulation with the same parameters using $\theta = 1$ (green line).

4 Results and discussion

4.1 Plasma without an external field

Simulations were done to study the heating mechanisms in an UCP without an external field. In these simulations the initial electron temperature was taken to be zero. Because the heating of the plasma is studied, the scaled temperature is of interest here. In all results in this section, the inverse coupling parameter is plotted versus the scaled time in all results. As argued in section 2.3.1, the scaled equations of motion and thus the scaled behaviour of the plasma should be the same. As we will see later, this assumption is true for the performed simulations.

4.1.1 Independence of input parameters

In Fig. 3 the results from three simulations are plotted. A plasma of 500 electrons and 500 ions was simulated with three different densities. The other initial conditions were exactly the same. The positions of the particles were only scaled to give the plasma another density, which is realised by choosing the same random seed for each simulation run. The results are macroscopically the same, as was expected from section 2.3. Even microscopically the results are the same up to about $2\omega_p t$. This is also what's expected from Eq. (10). After that point the results begin to differ in the microscopic details, which is caused by rounding errors in the numerical simulation. This leads to minimal differences at first, but because the plasma is chaotic, these small differences cause a completely different microscopic behaviour.

The initial positions of the electrons and ions do matter for the microscopic behaviour, as can be seen in Fig. 4. The three simulations performed here all had different initial particle positions, but the other parameters were the same. Although there is microscopically a dependence on the initial placement, the results are macroscopically the same.

Also another number of particles does not make a difference in the development of the plasma. In Fig. 5 the inverse coupling parameter is plotted for three different numbers of particles. These results are just as much the same as the results plotted in Fig. 4.

4.1.2 Averaged result

Because the electron temperature is defined to be $3k_B T_e/2 = \langle U_{kin} \rangle$, large peaks can occur if one electron passes very close by an ion and will thus get a large kinetic energy for a small period of time. To reduce this noise, the results of 100 simulation runs were averaged. In Fig. 6, this average result is plotted. The simulations are similar to the ones performed by Kuzmin and O'Neil [6], except that they use a finite simulation box with reflective walls, while in our simulations there are no boundaries. Still the results are very comparable as can be seen in Fig. 6.

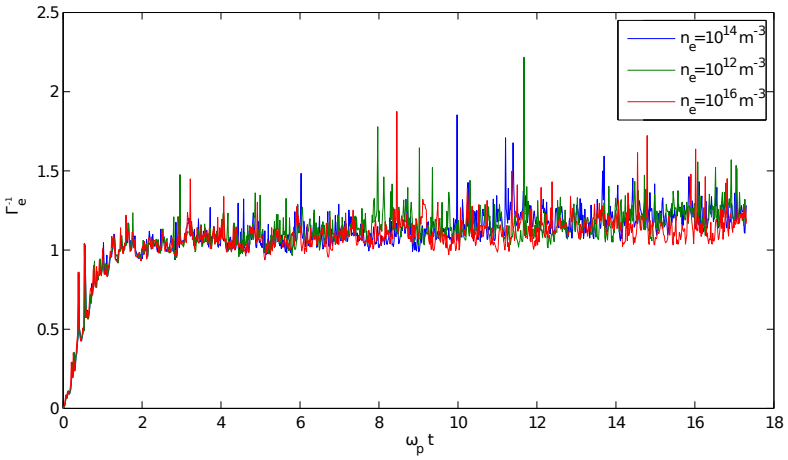


Figure 3: Scaled temperature versus scaled time for three different densities. All other parameters were kept the same for the three simulations.

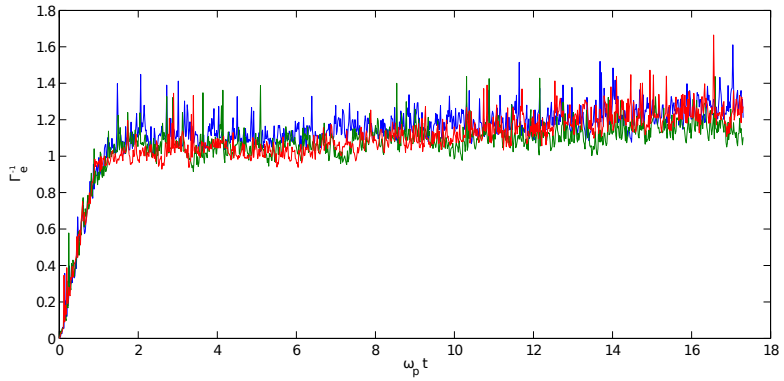


Figure 4: Scaled temperature versus scaled time for three different random seeds. Every parameter is the same in these simulations, except for the initial positions of the particles.

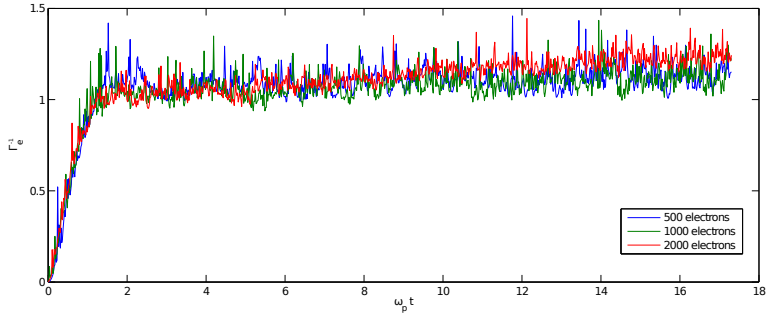


Figure 5: Scaled temperature versus scaled time for three different numbers of particles.

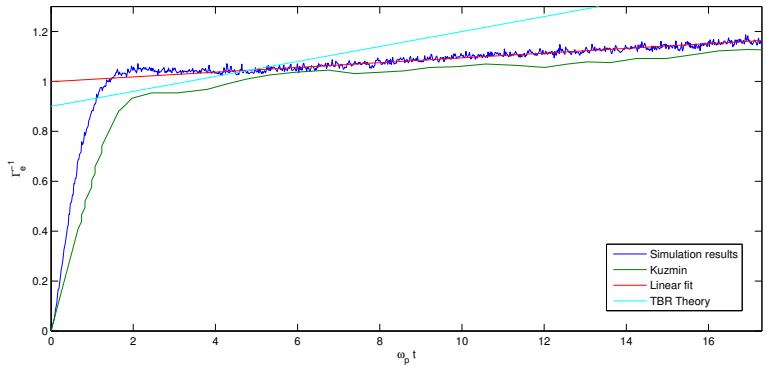


Figure 6: Comparison of an average of 100 simulations with the result obtained by Kuzmin and O'Neil [6]. Also a linear fit through the data for $\omega_p t > 4$ and the heating predicted by Eq. (17) *is plotted.*

4.1.3 Disorder induced heating

As expected, an initial rapid heating is observed in all performed simulations. As discussed in section 2.3.2 this is disorder-induced heating. In Fig. 6 it is clear that the heating takes place before $\omega_p t = 1$, which is expected since that corresponds to $t = 1/\omega_p$. Also an overshoot in the electron kinetic energy is visible. This phenomenon has also been observed in the disorder-induced heating of ions in experiments performed by Cummings et al. [12], and was attributed to strong-coupling effects. However, in Fig. 6 it is difficult to determine the precise length and amplitude of the overshoot. In addition, other processes in the plasma will probably occur at the time scale ω_p^{-1} as well. It is therefore uncertain whether the observed overshoot is due to strong-coupling effects.

4.1.4 Three-body recombination

After the disorder-induced heating phase, the electron temperature still rises in Figs. 3-6. This is expected to be caused by three-body recombination, see section 2.3.3. In Fig. 6 a line is fitted through the data in the TBR phase. The slope of this line is 9.5×10^{-3}. When evaluating the theoretically expected slope from Eq. (17), a numerical value of $3(1) \times 10^{-2}$ is found. The line corresponding to this slope is also plotted in Fig. 6. This result is remarkably good, since the predicted value – derived for conventional plasmas with $\Gamma < 10^{-3}$ –- differs only a factor 3 with the numerical result.

Also, according to Eq. (17), the amount of electrons that should have been recombined with an ion at time $\omega_p t = 17$, was more than half of the total number of electrons. Looking at the simulation data, it was clear that quite some electrons were orbiting around ions, but not even close to what the theory predicted.

A possible explanations why the $T^{-9/2}$ law of Eq. (14) starts to deviate for UCPs, is that the Thomson radius (12) is larger than the typical inter-ion distance if $\Gamma \approx 1$. The assumption that a thermalizing collision inside the Thomson sphere can cause an

electron to be recombined with that ion is not valid any more, because a large part of the Thomson sphere is closer to other ions. This means that according to the derivation a larger volume is assumed in which an electron can recombine with any ion, than the total volume of the actual plasma. This will cause the TBR rate to be overestimated.

4.2 Plasma in an external field

In this section, the validity of Eq. (27) for collisional heating as described in section 2.4.4 is tested. This equation is derived for the weak field regime, i.e. the quiver velocity due to the electric field is much lower than the average thermal velocity. Therefore the ponderomotive energy corresponding to the electric field is chosen to be about 1% of $k_B T$. Part of Eq. (27) is the Coulomb logarithm. To be sure that Eq. (6) for the Coulomb logarithm is valid, a plasma in the weakly coupled regime is tested. The electrons are given an initial temperature to get $\Gamma \approx 0.1$-0.2. See appendix A for the exact parameters used in the simulations.

Just as in the situation without an external field, the electrons are constantly exchanging potential and kinetic energy. Because the most interesting property in collisional heating is the energy gain, it is more informative to study the total energy of the plasma than the kinetic energy of the electrons. This removes the violent fluctuations, such as in Figs. 3-5, from the numerical results. In Fig. 7, the normalized energy, denoted with tildes, is plotted against the normalized time $\omega_p t$. The normalized energy is defined as the net mean energy gain per electron, normalized on the typical Coulomb energy $U_{norm} = e^2/(4\pi\epsilon_0 a)$:

$$\tilde{U} = \frac{U_{tot} - U_{tot,0}}{N_e} \frac{4\pi\epsilon_0 a}{e^2}, \tag{30}$$

where a is the Wigner Seitz radius from equation (4) and N_e is the total number of electrons.

The blue line in Fig. 7 is the result from the simulation and the green line is a plot of U_{theory}/U_{norm}, with:

$$\begin{aligned} U_{theory} &= U_{heating} + U_{quiver} \\ &= \int_0^{\omega_p t} \frac{1}{N_e} \frac{\mathrm{d}U_{total}}{\mathrm{d}(\omega_p t')} \mathrm{d}(\omega_p t') + 2U_p \sin^2(\omega_d t), \end{aligned} \tag{31}$$

where $\frac{\mathrm{d}U_{total}}{\mathrm{d}(\omega_p t')}$ is taken from Eq. (27) and $2U_p \sin^2(\omega_d t)$ is the expected quiver energy from Eq. (21). The integral from this equation is evaluated numerically, using Γ and the number of electrons N_e inside the plasma from the simulation data.

As can be seen, the theory and the simulation result match very well, both in the oscillation amplitude and the slow increase. It can therefore be concluded that Eq. (27) is valid for the parameter regime from this simulation.

The vacuum quiver energy is not always a good approximation, however. This can clearly be seen in Fig. 8. The simulation parameters were the same as for the result in Fig. 7, but the initial particle positions were different. These results can be explained when eigen-oscillations of the electron sphere are taken into account. These oscillations are described in section 2.4.3. By replacing the quiver energy component from Eq. (31) with Eq. (24), the green line from Fig. 8 is obtained. The two parameters x_0

and v_0 were obtained from the simulation data. By calculating the average x and v_x from the ions and the electrons separately and subtracting these, x_0 and v_0 are calculated.

Using this approach also gives a good result for the data from Fig. 7. The vacuum quiver energy was by accident valid as well, since x_0 and v_0 were such, that there was hardly a contribution from the eigen-oscillation.

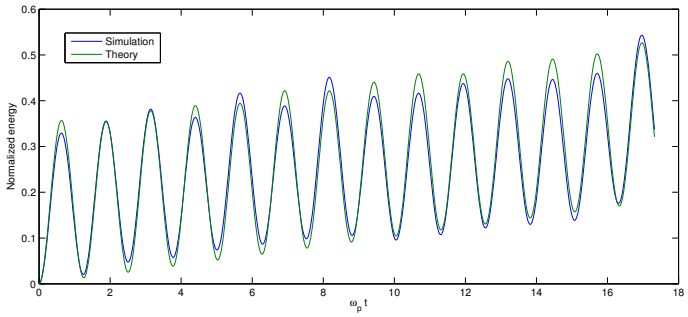


Figure 7: Normalized total energy over the course of the simulation and the theoretically expected result.

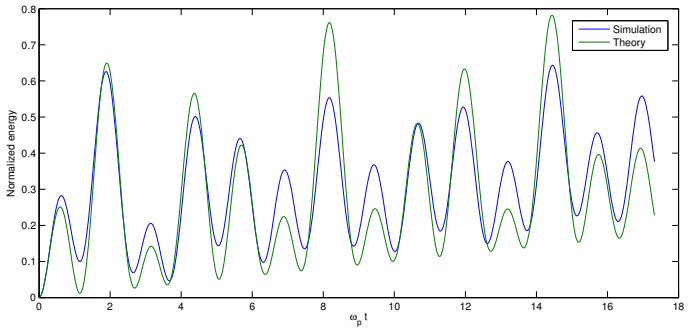


Figure 8: Results from a simulation with the same parameters as in Fig. 7, but with a different particle placement. The effect of the eigen-oscillations is clearly visible.

4.2.1 Variation of simulation parameters

In this section, the simulation from Fig. 7 is used as a reference case. The plasma parameters are varied one by one with respect to this case to study their influence on collisional heating.

In Fig. 9, the results of two more simulations with a different external field strength, and therefore a different ponderomotive energy are plotted. Plotted here is $\tilde{U} - \tilde{U}_{quiver}$ versus the scaled time. Because the main oscillations, as visible in Figs. 7 and 8, have been subtracted out, this gives a better view of the collisional heating, but it also emphasises the errors in the approximation for the expected quiver energy.

In Fig. 9, the numerical results are compared again with the energy increase predicted by Eq. (27). What can be seen in this figure is that the theory for collisional heating also works for lower electric field strengths than that of Fig. 7. This was to be expected since in this case the quiver velocity is even smaller than the thermal velocity, so the assumption that $v_{quiver} \ll v_{thermal}$ is even better.

Also for different starting temperatures than that of Fig. 7, Eq. (27) theory remains valid as can be seen in Fig. 10. In all cases, the plasma remained weakly coupled and $v_{quiver} \ll v_{thermal}$, so this is still the right parameter regime.

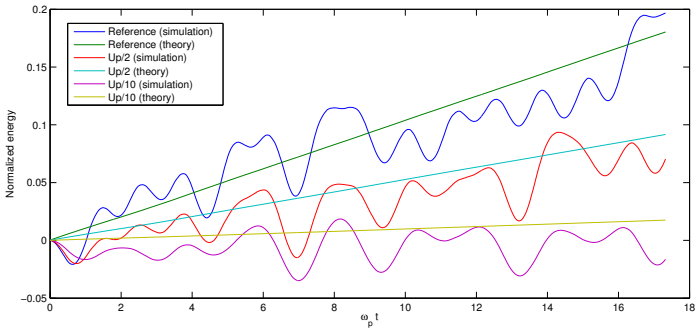


Figure 9: For several values of the ponderomotive energy, the theoretically expected collisional heating (straight lines) is plotted along with the heating obtained from the simulations.

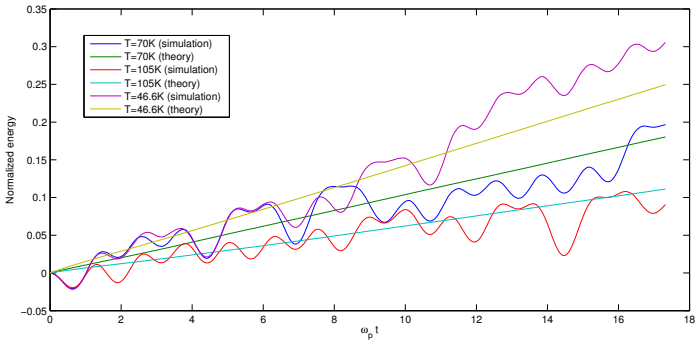


Figure 10: For several values of the initial electron temperature, the theoretically expected collisional heating (straight lines) is plotted along with the heating obtained from the simulations.

Lowering the density gives unexpected results however. As can be seen in Fig. 11, the plasma heats less than expected from Eq. (27) when the density is reduced by a factor 4 compared to that of Fig. 7.

A possible explanation for this is that the theory assumes that electrons interact with ions within a full Debye-sphere around them. By decreasing the density, the relative size of the Debye length λ_D compared to the radius of the plasma sphere r_b becomes larger. This means that a larger fraction of the electrons does not have its Debye-sphere filled with ions, which may cause the observed discrepancy. By taking two times as

much particles as in Fig. 11, the ratio λ_D/r_b becomes about as large as in Fig. 7 again. The result from this simulation can be seen in Fig. 12. In this simulation Eq. (27) and the simulation do agree.

In section 2.4.1 it was argued that the plasma parameters can be scaled to eliminate the density dependency. In Fig. 13, the same scaled parameters are used as in the simulation from Fig. 11, only for a density that is the same as in the reference simulation. Because the parameters are scaled, the same scaled behaviour is expected in this situation. However, as can be seen in Fig. 13, the simulation agrees with the theory, but it was expected that the behaviour of this scaled plasma is the same as the behaviour of the plasma from the previous paragraph. It is therefore not really clear why the theory does not give a good result for that simulation. Perhaps this simulation contained by accident many pathological initial conditions, such as many electrons that were placed very close to each other to create very high local densities. The theory might perform well using another initial configuration in which these pathological conditions were absent. More research is needed however before that can be concluded.

What can also be seen in Figs. 12 and 13 is that there is a clear beating at the Mie frequency. The model from section 2.4.3 might therefore improved by including damping. This will eliminate the beating that is still visible.

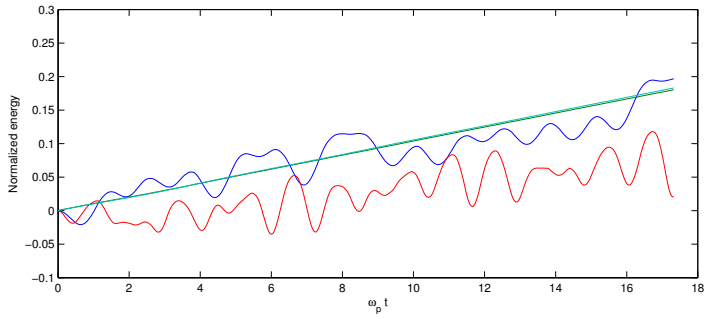


Figure 11: Collisional heating for a simulation with a lower density (red line) than the reference simulation (blue line). The two straight lines are the predictions of the collisional heating from Eq. (27)

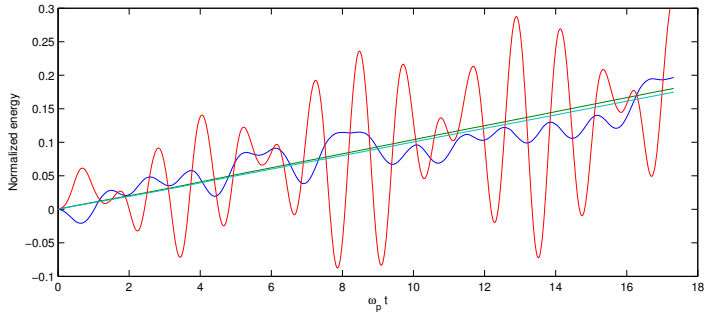


Figure 12: Collisional heating for the reference simulation (blue line) and a simulation with twice as much particles and a lower density (red line). The two straight lines are the predictions of the collisional heating from Eq. (27)

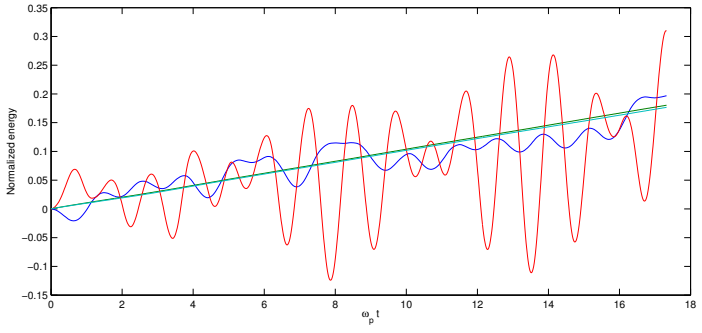


Figure 13: Collisional heating for the reference simulation (blue line) and a simulation with the same scaled parameters as for Fig. 11 (red line). The two straight lines are the predictions of the collisional heating from Eq. (27)

5 Conclusion and outlook

In this thesis, numerical simulations of ultra-cold plasmas were performed. Two situations were studied: one in which the plasma was allowed to develop undisturbed and one in which an external oscillating electric field was applied. For the case without an external field, simulations were done with zero initial temperature. The results are similar to the results found by Kuzmin and O'Neil [6], therefore these results can be expected to be reliable. The results from the simulations were independent of the density of the simulated plasma. In this report, it was shown that this is expected theoretically, because the equations of motion for the individual electrons can be written independent of the plasma density. The simulations clearly showed two phases in the development of the UCP. The first phase is disorder-induced heating. This heating gave the expected results: a rapid heating to $\Gamma \approx 1$. After the disorder-induced heating, another heating process is visible. This is contributed to three-body recombination. The three-body recombination theory from conventional plasmas gave quite good results. The theory is within a factor 3 the same as obtained from the simulations. For the case with an electric field, an UCP was simulated to test whether the conventional theory for collisional heating is also valid for the extremely low temperature of UCPs. The theory was tested in the regime that was expected to give the best results: this is for a weak coupling and a relatively weak electric field in which the quiver velocity is much less than the thermal velocity. For this regime, the results match extremely well with the theory, although for one case – where the density is lower – the theory and the simulation results do not match. It can therefore be concluded that this theory is valid for most of the tested regime.

It would be interesting to also research other regimes for the case with an electric field. If the plasma is strongly coupled or if a strong electric field is applied, the collisional heating theory used in this report is not valid for these regimes and more advanced theory is necessary. This theory also needs to be tested for UCPs. There are also heating mechanisms which are not included in this thesis. For example, absorption due to plasma resonance [13] or collisionless absorption due to the finite plasma size [14]. Also, stronger electric fields may influence three-body recombination [14].

References

[1] T. C. Killian, T. Pattard, T. Pohl, and J. Rost, Ultracold neutral plasmas, Phys. Rep. **449**, 77 (2007).

[2] D. Nicholson, *Introduction to Plasma Theory* (John Wiley & Sons, 1983).

[3] L. Spitzer, *Physics of Fully Ionized Gases* (Interscience, 1962).

[4] D. O. Gericke, M. S. Murillo, and M. Schlanges, Dense plasma temperature equilibration in the binary collision approximation, Phys. Rev. E **65**, 036418 (2002).

[5] V. P. Silin, Nonlinear High-Frequency Plasma Conductivity, Soviet Physics JETP **20** (1965).

[6] S. G. Kuzmin and T. M. O'Neil, Numerical Simulation of Ultracold Plasmas: How Rapid Intrinsic Heating Limits the Development of Correlation, Phys. Rev. Lett. **88**, 065003 (feb 2002).

[7] M. Mitchner and C. H. Kruger, *Partially ionized gases* (Wiley-Interscience, 1973).

[8] E. Hinnov and J. G. Hirschberg, Electron-Ion Recombination in Dense Plasmas, Phys. Rev.(1962).

[9] P. Mulser and D. Bauer, *High Power Laser-Matter Interaction* (Springer, 2010).

[10] "`http://www.pulsar.nl/gpt`," .

[11] J. Barnes and P. Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, Nature(1986).

[12] E. A. Cummings, J. E. Daily, D. S. Durfee, and S. D. Bergeson, Fluorescence Measurements Of Expanding Strongly Coupled Neutral Plasmas, Phys. Rev. Lett. **95**, 235001 (Nov 2005).

[13] K. A. Twedt and S. L. Rolston, Electronic Detection of Collective Modes of an Ultracold Plasma, Phys. Rev. Lett. **108**, 065003 (Feb 2012).

[14] P. W. Smorenburg, L. P. J. Kamp, and O. J. Luiten, Heating mechanisms in radio-frequency-driven ultracold plasmas, Phys. Rev. A **85**, 063413 (Jun 2012).

A Simulation parameters

In the table below, the simulation parameters for the simulations in this thesis are displayed. Except for the simulations in Fig. 1 and Fig. 2, in all simulations the accuracy parameter *acc* is 8.5 and the Barnes-Hut parameter *theta* is 1. In Fig. 1 *acc* is on the x-axis and the Barnes-Hut algorithm is not used. In Fig. 2 *acc* is 8.5 and the usage of the Barnes-Hut algorithm is stated in the caption.

Figure	nps N_i [-]	dens n_i [m^3]	eps ϵ [-]	wd ω_d [rad s^{-1}]	Up U_p [J]	T T_0 [K]
1, see caption for nps	-	10^{14}	10^{-6}	0	0	0
2	1000	10^{14}	10^{-3}	0	0	0
6	500	10^{14}	10^{-3}	0	0	0
3, see legend for *dens*	500	-	10^{-3}	0	0	0
4	1000	10^{14}	10^{-3}	0	0	0
5, see legend for *nps*	-	10^{14}	10^{-3}	0	0	0
7, 8	4000	10^{16}	10^{-3}	1.41×10^{10}	1.5×10^{-23}	70
9, *Up* for reference run	4000	10^{16}	10^{-3}	1.41×10^{10}	1.5×10^{-23}	70
10, see legend for *T*	4000	10^{16}	10^{-3}	1.41×10^{10}	1.5×10^{-23}	-
11	4000	2.5×10^{15}	10^{-3}	1.41×10^{10}	1.5×10^{-23}	70
12	8000	2.5×10^{15}	10^{-3}	1.41×10^{10}	1.5×10^{-23}	70
13	4000	10^{16}	10^{-3}	2.82×10^{10}	2.38×10^{-23}	111

B Simulation code

B.1 GPT input files

Listing 1: plasma.in

```
# -- Create the particles --
setparticles("electrons", nps,      me,  qe,  nps*qe);
setparticles("ions"     , nps, 85.5*mp, -qe, -nps*qe);

# Calculate the initial bunch radius
rb  = ((3*nps)/(4*pi*dens))^(1/3);
# Calculate the Wigner-Seitz radius
a = (3/(4*pi*2*dens))^(1/3);
# Calculate the plasma frequency
wp = sqrt((dens*qe*qe)/(me*eps0));
# Calculate the Mie frequency
wm = wp/sqrt(3);

# set random seed
if (rseed==-1) {
    # use current time to set the random seed
    randomize();
} else {
    # set the random seed
    randomize(rseed);
}

# Particle placement:
# 'setellipse' places the particles with a uniform
# random distribution inside an ellipsoid.
# In this case the ellipsoid is a sphere.
setellipse("electrons",rb,rb,rb);
setellipse("ions"     ,rb,rb,rb);

# initial thermal velocities
kb = 1.38e-23;
setGBxdist("electrons", "G", 0, sqrt(kb*T/me)/c, 3, 3);
setGBydist("electrons", "G", 0, sqrt(kb*T/me)/c, 3, 3);
setGBzdist("electrons", "G", 0, sqrt(kb*T/me)/c, 3, 3);

# set the Coulomb round-off parameter R
R = eps*a;
setrmacrodist("electrons","u",R,0);
setrmacrodist("ions"     ,"u",R,0);

# -- External fields --
# Calculate the amplitude of the field from the
# given ponderomotive energy:
E0 = sqrt(4*me*wd^2*Up/qe^2);
# tells GPT to include the oscillating field
Eoscillating("wcs","I",E0,wd,0);

# save values in the gdf output file
outputvalue("wd", wd);
outputvalue("E0", E0);
outputvalue("R", R);

# Use the Barnes-Hut approximation to calculate
# the electrostatic particle-particle interactions
spacecharge3Dtree(theta);

```

```
57 # Set the accuracy for the Runge-Kutta solver
58 accuracy(acc);
59
60 if( verbose ) {
61     # print some values to the terminal
62     pp("rb=", rb);
63     pp("nps=", nps);
64     pp("a=", a);
65     pp("rmacrodist:", R);
66     pp("wp=", wp);
67     pp("wd=", wd);
68     pp("E0=", E0);
69     pp("Up=", Up);
70 }
71
72 # Simulate up to t=10/wm and save positions/velocities/etc.
73 # every 0.01/wm
74 snapshot(0, 10/wm, 0.01/wm);
```

Listing 2: scan.mr

```
1  # set the number of electrons and ions
2  # the total number of particles is 2*nps
3  nps 4000
4
5  # set the electron density in [m^-3]
6  dens 1e16
7
8  # set random seed
9  # different random seeds cause different initial positions/speeds,
10 # setting rseed=-1 will cause the random seed to be the current time
11 rseed 1234
12
13 # Coulomb potential round-off parameter
14 eps 1e-3
15
16 # Set the angular frequency of the external electric field
17 wd 1.4e10
18 # Set the ponderomotive energy belonging to the external field
19 Up 1.5e-23
20 # Set the initial electron temperature
21 T 70
22
23 # Set the accuracy parameter for the Runge-Kutta solver
24 acc 8.5
25
26 # Set the accuracy parameter for the Barnes-Hut algorithm
27 theta 1
28
29 # show some debug output
30 verbose 1
```

Listing 3: plasma.bat

```
1 mr -j 1 -v -o plasma.gdf scan.mr "time gpt" plasma.in
2 gdfa -o analysed.gdf plasma.gdf time Kinside invG totalEnergy insideRb x v
3 gdf2a -w 16 -o analysed.txt analysed.gdf
```

B.2 GPT custom elements

Listing 4: setellipse.c

```
/* setellipse.c - Set homogeneous ellipse */

#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "elem.h"

extern double dblpulsarrand(void);

void setellipse_init(gptinit *init)
{
  double a,b,c;
  double x,y,z;
  gptparset *set;
  gptinitpar *par;
  char *name;
  int i, len;

  if( gptgetargnum(init)!=4 )
    gpterror("Syntax: %s(set,a,b,c)\n", gptgetname(init));

  name = gptgetargstring(init,1);
  a    = gptgetargdouble(init,2);
  b    = gptgetargdouble(init,3);
  c    = gptgetargdouble(init,4);

  /* Get particle set */
  if( gpttestparset(name)==NULL )
    gptwarning("The particle set \"%s\" does not exist\n", name);
  set = gptgetparset(name);
  par = gptgetparsetpars(set,&len);

  /* Set ellipse */
  for( i=0; i<len; i++ )
  {
    do
    {
      /* Uniform in box between -a and a, -b and b, -c and c */
      x = 2*dblpprand()-1;
      y = 2*dblpprand()-1;
      z = 2*dblpprand()-1;
    }
    while( x*x+y*y+z*z >= 1 );

    par[i].Wr[0] = a*x;
    par[i].Wr[1] = b*y;
    par[i].Wr[2] = c*z;
  }
}
```

Listing 5: Eoscillating.c

```
/* Eoscillating.c:  */

#include <stdio.h>
#include <math.h>
#include "elem.h"

/*Info structure containing all relevant parameters for this element*/
```

```
 8 struct Eoscillating_info
 9 {
10   double Eo ;
11   double w ;
12   double phi ;
13 } ;
14
15 /* Forward declaration of the routine calculating the electromagnetic
       fields */
16 static int Eoscillating_sim(gptpar *par,double t,struct
       Eoscillating_info *info) ;
17
18 /* Initialization routine */
19 void Eoscillating_init(gptinit *init)
20 {
21   struct Eoscillating_info *info ;
22
23   /* Read Element Coordinate System (ECS) from parameter list */
24   gptbuildECS( init ) ;
25
26   /* Print usage line when the number of parameters is incorrect */
27   if( gptgetargnum(init)!=3 )
28     gpterror( "Syntax: %s(ECS,Eo,w,phi)\n", gptgetname(init) ) ;
29
30   /* Allocate memory for info structure */
31   info = (struct Eoscillating_info *)gptmalloc( sizeof(struct
         Eoscillating_info) ) ;
32
33   /* Read all parameters as doubles and store them in info structure */
34   info->Eo        = gptgetargdouble(init,1) ;
35   info->w         = gptgetargdouble(init,2) ;
36   info->phi       = gptgetargdouble(init,3) ;
37
38   /* Register the routine calculating the electromagnetic fields to the
         GPT kernel */
39   gptaddEBelement( init, Eoscillating_sim, gptfree, GPTELEM_LOCAL, info
         ) ;
40 }
41
42
43 /* The following routine calculates the electromagnetic fields */
44 static int Eoscillating_sim(gptpar *par,double t,struct
       Eoscillating_info *info)
45 {
46   /* Copy of parameters in info structure for convenience */
47   double Eo, w, phi ;
48
49   /* Retrieve parameters from info structure */
50   Eo        = info->Eo ;
51   w         = info->w ;
52   phi       = info->phi ;
53
54   /* Calculate electromagnetic fields from the above parameters
55    * Particle coordinates: X,Y and Z   must be written UPPERCASE
56    * Simulation time     : t           must be written lowercase
57    * Electric field      : EX = ... ; EY = ...  ; EZ = ... ;
58    * Magnetic field      : BX = ... ; BY = .... ; BZ = ... ;
59    */
60   EX = Eo*cos(w*t+phi) ;
61
62   /* Return 1 to notify particle is INSIDE element */
63   return( 1 ) ;
64 }
```

B.3 GDFA custom programs

Listing 6: insideRb.c

```
/* insideRb.c returns the number of particles inside the
     original plasma sphere */

#include <math.h>
#include <stdlib.h>

#include "gdfa.h"

int insideRb_func( double *result )
{
  int i, num, tmpnum ;
  double *nmacro, *x, *y, *z  ;
  double rb, rb2 ;
  int amount = 0;

  if( gdfmgetarr( "nmacro", &nmacro, &num ) || num<2 ||
      gdfmgetarr( "x",  &x,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "y" , &y,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "z" , &z,  &tmpnum ) ||  tmpnum!=num ) return(1) ;

  if( gdfmgetval( "rb", &rb ) ) {
    double dens, nps;
    if (gdfmgetval("dens",&dens) || gdfmgetval("nps",&nps)) return (1);
    rb = pow((3*nps)/(4*gpt_pi*dens), 1/3.0);
  }
  rb2 = rb*rb ;

  for (i=0; i<num; i++)
    if (x[i]*x[i]+y[i]*y[i]+z[i]*z[i] < rb2)
      amount ++;

  *result = (double) amount;

  return(0) ;
}
```

Listing 7: invG.c

```c
/* invG.c: calculate the average inverse electron coupling parameter */

#include <math.h>
#include <stdlib.h>

#include "gdfa.h"

int invG_func( double *result )
{
  int num, tmpnum, i, num_el ;
  double *nmacro, *m, *G  ;
  double Ga ;

  if( gdfmgetarr( "nmacro", &nmacro, &num ) ||
      gdfmgetarr( "m",  &m,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "G" , &G,  &tmpnum ) ||  tmpnum!=num ) return(1) ;

  Ga = 0;
  num_el = 0;
  for (i=0; i<num; i++)
    if (m[i]==gpt_me) {
      Ga += G[i];
      num_el ++;
    }
  Ga /= num_el ;
  /* average electron kinetic energy: */
  *result = gpt_me*gpt_c*gpt_c*(Ga-1);

  double dens, Enorm;
  if ( gdfmgetval("dens", &dens) ) return (1);
  Enorm =  gpt_qe*gpt_qe/(4*gpt_pi*gpt_eps0)*cbrt(4*gpt_pi*2*dens/3);
  /* calculate the inverse coupling parameter */
  *result = *result/Enorm*2/3;

  return(0) ;
}
```

Listing 8: Kinside.c

```c
/* Kinside.c: calculate the average kinetic energy of the
     electrons inside the original plasma sphere */

#include <math.h>
#include <stdio.h>

#include "gdfa.h"

int Kinside_func( double *result )
{
  int num, tmpnum ;
  double *nmacro, *m, *G, *x, *y, *z  ;

  if( gdfmgetarr( "nmacro", &nmacro, &num ) ||
      gdfmgetarr( "m",  &m,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "G" , &G,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "x",  &x,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "y" , &y,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "z" , &z,  &tmpnum ) ||  tmpnum!=num ) return(1) ;

  double rb;
  if( gdfmgetval( "rb", &rb ) ) {
    double dens, nps;
    if (gdfmgetval("dens",&dens) || gdfmgetval("nps",&nps)) return (1);
    rb = pow((3*nps)/(4*gpt_pi*dens), 1/3.0);
  }
  double rb2 = rb*rb;

  double Ga = 0;
  int num_el = 0;
  for (int i=0; i<num; i++) {
    if (m[i]==gpt_me) {
      if ( (x[i]*x[i]+y[i]*y[i]+z[i]*z[i])<rb2) {
        Ga += G[i];
        num_el ++;
      }
    }
  }
  Ga /= num_el ;
  *result = gpt_me*gpt_c*gpt_c*(Ga-1) ;

  return 0;
}
```

Listing 9: totalEnergy.c

```
1
2  /* totalEnergy.c: calculate the sum of potential and
3      kinetic energy of the system */
4
5  #include <math.h>
6  #include <stdlib.h>
7
8  #include "gdfa.h"
9
10 int totalEnergy_func( double *result )
11 {
12   int num, tmpnum, i, j ;
13   double *nmacro, *x, *y, *z, *q, *m, *Bx, *By, *Bz ;
14   double R,R2;
15   double r_1 = 0;
16   double Ukin, Upot ;
17   double dx,dy,dz;
18
19   if( gdfmgetarr( "nmacro", &nmacro, &num ) ||
20       gdfmgetarr( "x", &x, &tmpnum ) || tmpnum!=num ||
21       gdfmgetarr( "y", &y, &tmpnum ) || tmpnum!=num ||
22       gdfmgetarr( "z", &z, &tmpnum ) || tmpnum!=num ||
23       gdfmgetarr( "Bx", &Bx, &tmpnum ) || tmpnum!=num ||
24       gdfmgetarr( "By", &By, &tmpnum ) || tmpnum!=num ||
25       gdfmgetarr( "Bz", &Bz, &tmpnum ) || tmpnum!=num ||
26       gdfmgetarr( "m", &m, &tmpnum ) || tmpnum!=num ||
27       gdfmgetarr( "q", &q, &tmpnum ) || tmpnum!=num ) return(1) ;
28
29   if ( gdfmgetval( "R", &R) ) return (1);
30   R2 = R*R;
31
32   for (i=0; i<num; i++) {
33     for (j=i+1; j<num; j++) {
34       dx = x[i] - x[j];
35       dy = y[i] - y[j];
36       dz = z[i] - z[j];
37       r_1 += q[i]*q[j]/sqrt(dx*dx + dy*dy + dz*dz + R2);
38     }
39   }
40
41   Upot = 1/(4*gpt_pi*gpt_eps0)*r_1;
42
43   Ukin = 0;
44   for (i=0; i<num; i++) {
45     Ukin += m[i]*(Bx[i]*Bx[i] + By[i]*By[i] + Bz[i]*Bz[i]);
46   }
47   Ukin = Ukin/2*gpt_c*gpt_c;
48
49   *result = Ukin+Upot;
50
51   return(0) ;
52 }
```

Listing 10: x.c

```c
/* x.c: calculate the horizontal displacement of the center of mass
     of the ions and the center of mass of the electrons */

#include <math.h>
#include <stdlib.h>

#include "gdfa.h"

int x_func( double *result )
{
  int num, tmpnum ;
  double *nmacro, *m, *x  ;

  if( gdfmgetarr( "nmacro", &nmacro, &num ) ||
      gdfmgetarr( "m",  &m,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "x" , &x,  &tmpnum ) ||  tmpnum!=num ) return(1) ;

  double x_el = 0;
  double x_ion = 0;
  int num_el = 0;
  int num_ion = 0;
  for (int i=0; i<num; i++)
    if (m[i]==gpt_me) {
      x_el += x[i];
      num_el ++;
    } else {
      x_ion += x[i];
      num_ion ++;
    }

  *result = x_el/num_el - x_ion/num_ion;

  return(0) ;
}
```

Listing 11: vx.c

```
/* vx.c, calculate the horizontal relative speed of the center of mass
    of the ions and the center of mass of the electrons */

#include <math.h>
#include <stdlib.h>

#include "gdfa.h"

int vx_func( double *result )
{
  int num, tmpnum ;
  double *nmacro, *m, *Bx  ;

  if( gdfmgetarr( "nmacro", &nmacro, &num ) ||
      gdfmgetarr( "m",  &m,  &tmpnum ) ||  tmpnum!=num ||
      gdfmgetarr( "Bx" , &Bx,  &tmpnum ) ||  tmpnum!=num ) return(1) ;

  double Bx_el = 0;
  double Bx_ion = 0;
  int num_el = 0;
  int num_ion = 0;
  for (int i=0; i<num; i++)
    if (m[i]==gpt_me) {
      Bx_el += Bx[i];
      num_el ++;
    } else {
      Bx_ion += Bx[i];
      num_ion ++;
    }

  *result = (Bx_el/num_el - Bx_ion/num_ion)*gpt_c;

  return(0) ;
}
```

B.4 Matlab scripts

Script to import the data from GPT into Matlab.

Listing 12: import.m

```
file = 'analysed.txt'   % file name of the output file of 'gdf2a'

% extra parameters necessary for the analysis
dens = 1e16             % density
nps = 4000              % number of electrons particles
wd = 1.41e10            % drive frequency
Up = 1.5e-23            % ponderomotive energy


% import the data
data = dlmread(file, '', 1, 0);
% time in seconds
t = data(:,1);
% average kinetic energy of the electrons inside the plasma
K = data(:,2);
% inverse coupling constant of ALL electrons
invG = data(:,3);
% total energy of the system
U = data(:,4);
% number of particles inside the plasma sphere
%  since the number in this sphere does not change,
%  the number of electrons in the sphere is (insideRb-nps)
insideRb = data(:,5);
x = data(:,6);
vx = data(:,7);

% perform analysis
[T, Us, Uts, Uqs] = analysis(t,K,U,insideRb,dens,nps,wd,Up,x(1),vx(1));
```

Analysis for the simulation with an external field. This script calculates the theoretically expected energy and scales both this energy and the energy coming out of the simulation.

Listing 13: analyse.m

```
function [ T, Us, Uts, Uqs ] = scaled_analysis(t, K, U, ...
          insideRb, dens, nps, wd, Up, x0, v0 )
    me = 9.109e-31;
    qe = 1.602e-19;
    eps0 = 8.85e-12;

    % initial total density
    n0 = 2*dens;
    % initial electron density
    ne0 = dens;
    % electron density at every timestep
    ne = (insideRb-nps)/nps*dens;
    % ion density (remains constant during this simulation)
    ni = dens;

    % calculate the plasma frequency
    wp = sqrt(ne0*qe*qe/me/eps0);
    % and the Mie frequency
    wm = wp/sqrt(3);
    % the Wigner-Seitz radius
    a = (3/(4*pi*n0))^(1/3);
    % and the electric field strength
    E0 = 2*sqrt(Up*me)*wd/qe;

    % scaled time
    T = wp*t;
    % scale energy
    Uscale = qe^2/(4*pi*eps0*a);

    % Coupling constant for the electrons inside the plasma
    G = Uscale./(2/3*K);

    % calculate nu/wp: (spitzer collision frequency)/(plasma frequency)
    Lambda = sqrt(G.^-3*n0/3./ne);
    nu_wp = sqrt(2/3/pi)*ni/sqrt(ne0*n0)*G.^(3/2).*log(Lambda);

    % scaled total energy
    Us = U/Uscale/nps;
    % scaled theoretical power:
    % d(Us)/d(wp*t) = 2*nu_wp*Up/Us*(#electrons inside sphere)
    Ws = 2*nu_wp*Up/Uscale.*ne/dens;
    % scaled theoretical energy gain: integral(Ws dT)
    Uts = cumsum(Ws)*(T(2)-T(1));

    % quiver energy
    Uq = 0.5*me*(-x0*wm*sin(wm*t) + v0*cos(wm*t)+ ...
       qe*E0/(me*wd^2)*1/(1-wm^2/wd^2)*(wm*sin(wm*t)-wd*sin(wd*t))).^2;
    % scaled quiver energy
    Uqs = Uq/Uscale;

    %tozero = @(vec) vec-vec(1);
    %figure; plot(T, tozero(Us/nps), T, Uts+Uqs);
    %figure; plot(T, tozero(Us/nps-Uqs), T, Uts);
end
```