Towards a Hybrid Fluid-Monte Carlo Model for Pulsed Hollow Cathode Discharges

Mark Louwrens Beks
December 2003

EPG 03-17

Technische Universiteit Eindhoven

Elementary Processes in Gas discharges (EPG)

Supervisors :
ir. W.J.M. Brok,
ir. B.H.P. Broks,
dr. J.J.A.M van der Mullen

Abstract

This masters thesis reports research and design activities related to the construction of a model of a pulsed hollow cathode discharge in xenon. Pulsed hollow cathode discharges are under consideration for the production of extreme ultraviolet (EUV) radiation for use in lithographic systems. Shorter wavelengths are required by industry to continue the present trend towards smaller structures in integrated circuits. In the pulsed hollow cathode discharges for the production of EUV light, several Joule of energy are pumped into a small device (approximately 10 cm^3) filled with xenon, to a low pressure of 20 Pascal. A violent discharge follows and EUV radiation is emitted. The discharge lasts only some 100 ns.

The highly transient nature of the discharge, together with the low pressure, results in a plasma which is far from equilibrium. Essential features of a hollow cathode discharge, such as the “pendulum” effect, are non-local. The plasma in a pulsed hollow cathode discharge is, therefore, very difficult to model.

After examining existing methods of modeling pulsed hollow cathode discharges, the choice was made to use PLASIMO, a plasma simulation model developed at Eindhoven University of Technology, as part of a hybrid fluid-Monte Carlo model. A number of extensions have been added to facilitate the modeling of this type of highly transient discharge. These extensions have been verified using a number of test cases.

More specifically: a diffusion model has been implemented that solves the transport equation for the electrons without assuming quasi-neutrality, an extension has been added that solves the Poisson equation given the space charge distribution, an interface has been built that allows the use of the Monte Carlo code from Micro-dis to calculate ionization rates in PLASIMO and an extension has been added that injects electrons into the particle kinetic model to simulate secondary electron emission from the surfaces of the hollow cathode.

A number of issues, however, remain to be solved before a working model can be constructed. From the examination of the results of one of the test cases it has become clear that a different approach is required to solving the momentum balance without using the drift-diffusion equation. Additionally, parts of the PLASIMO code need to be rewritten to allow for the use of more complex geometries than the present rectangular grids.

Contents

Chapter 1

Introduction

1.1 Preface

This masters thesis reports research and design activities related to the construction of a model of a pulsed hollow cathode discharge in xenon. The research was carried out at the group Elementary Processes in Gas discharges (EPG) at Eindhoven University of Technology. Pulsed hollow cathode discharges are under consideration for the production of extreme ultraviolet (EUV) radiation for use in lithographic systems. Shorter wavelengths are required by industry to continue the present trend towards smaller structures in integrated circuits. In particular, light with a wavelength between 11 and 14 nm is required. In the pulsed hollow cathode discharges for the production of EUV light, several Joule of energy are pumped into a small device (approximately 10 cm^3) filled with xenon, to a low pressure of 20 Pascal. A violent discharge follows, ionizing the xenon to Xe^{12+} or higher and emitting EUV radiation. The discharge is highly transient and lasts only some 100 ns. Hence, the devices are operated in pulsed mode.

1.2 Technology Assessment

The hollow cathode discharges (HCD's) studied in this report are intended for the generation of radiation in lithographic systems for the production of integrated circuits. Therefore, we will shortly asses the development of these lithographic systems and the integrated circuits they produce.

The integrated circuits used in computers and other electronic devices have increased in complexity over the years. There is a clear trend to fabricate increasingly smaller structures on chips to allow for more components on the same chip while subsequently allowing for the increase of the clock speed of these devices. The 8008 processor, introduced by Intel® in 1972, and used in one of the first computers for home use, the Mark-8[1], contained 3,500 transistors operating at a clock speed of 500-800 kHz. The processor was built using 10 micrometer technology. Now thirty years later, Pentium IV® processors from the same company are built to run at clock speeds of 2-3 GHz, containing 55 million processors and built using 0.13 micrometer technology[2].

The plan of manufacturers of chips and lithographic systems alike is to continue on the road to increasingly smaller structures. This implies that the wavelength of the radiation used has to decrease accordingly since the wavelength puts a fundamental lower limit on the size of the structures. The continuation of the current miniaturization trend will require radiation with a wavelength much smaller than that what is used today.

This desire for smaller wavelength radiation is the driving force behind the research in hollow cathode devices for the production of extreme ultraviolet (EUV) radiation.

The shortest wavelengths for use in optical lithography to date is 157 nm. This radiation is called deep ultraviolet or DUV radiation. The imaging systems use lenses made from pure calcium fluoride[1]. However, there are no materials available that are sufficiently transparent to radiation of wavelengths shorter than 157 nm. Therefore, further reduction is only possible by using reflective systems. By using mirrors consisting of thin multiple layers, called distributed Bragg reflectors, a reasonable reflectivity is possible in the region between 11 and 14 nm. This radiation lies in the region of the spectrum, from 1 to 40 nm, called extreme ultraviolet radiation (EUV) but also known as vacuum ultraviolet (VUV) or soft X-Ray radiation [2].

Several sources for this radiation exist. An overview of all possible sources and a comparison on their performance is given by Stuik[3]. This study focuses on a pulsed HCD filled with xenon.

1.3 Pulsed Hollow Cathode Discharges for EUV Production

A hollow cathode of the type for EUV production is shown in figure 1.1. The discharge in such devices is highly transient. When the discharge is triggered, electrical breakdown follows with a rate of current rise reaching

[1] http://www.ics.uci.edu/ givargis/courses/217/articles/intel-timeline.pdf
[2] http://www.intel.com/pressroom/kits/quickreffam.htm

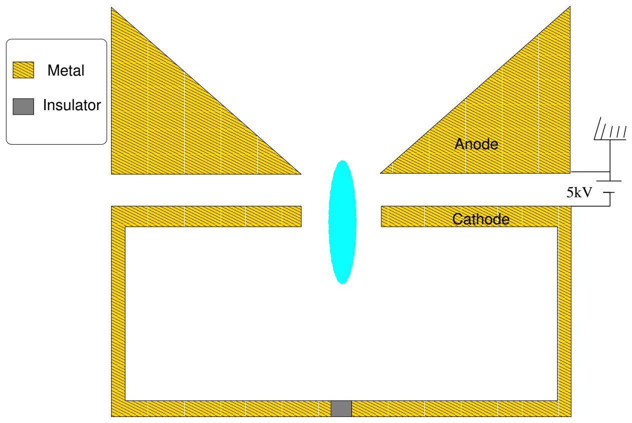


Figure 1.1: A schematic drawing of a hollow cathode of the type being considered for the production of EUV radiation. The diameter and height of the hollow cathode under study are both about 1.5 cm. The hollow cathode is filled with xenon to a pressure of approximately 20 Pascal. The potential difference between anode and cathode is 5 kV.

as high as 10^{12} A/s. The total duration of the discharge is some 100 ns during which the xenon rapidly ionizes. From time resolved spectroscopy performed with a pinhole camera, radiation from the 4d-4f transition in Xe^{12+} was observed, with evidence of the presence of Xe^{13+} [4]. Large currents (of the order of kA) are reached. This results in a pinched plasma emitting a EUV light. In one test, involving a 2 J pulse, 2.6 mJ of EUV radiation is emitted in a two percent band width around 13.4 nm [3], p. 62.

The highly transient nature of the discharge, together with the low pressure, results in a plasma which is far from equilibrium. The electron density increases from an initial value between 10^{13} to 10^{15} m^{-3} at the beginning of the pulse (depending on the repeat rate) to 10^{24} m^{-3} in the pinch. Essential features of a HCD, such as the "pendulum" effect, are non-local. The plasma in a pulsed HCD is, therefore, very difficult to model. After examining existing models of the hollow cathode the choice has been made to build a hybrid fluid-Monte Carlo model. At the group EPG two modeling platforms exist: PLASIMO and Micro-dis, also known as MD2D. The choice has been made to extend the former, because of its handling of the energy balance equations for all species. However, when this project was started PLASIMO could only handle quasi-neutrality and isotropic diffusion.

Therefore, extensions have been made to remove the quasi-neutrality assumption. To solve the electric field needed in this diffusion model a

Poisson solver has been added and tested.

Further extensions have been made to allow for anisotropic diffusion as a modification of the current fluid models. To allow for an even further departure from equilibrium a particle kinetic model, using Monte Carlo collisional (MCC) methods, has been added to PLASIMO . This addition allows the calculation of ionization rates from non-local effects in plasmas with a non-Maxwellian energy distribution function. To this end an interface has been build to use the Monte Carlo code from Micro-dis. The addition of such a particle kinetic model turns PLASIMO into a hybrid fluid-Monte Carlo model.

1.4 Overview

This report is organized as follows: After this short introduction we start out with a qualitative description of a complete cycle in the operation of a pulsed HCD. Directly following this qualitative description we analyze existing models and their limitations. The required extensions are discussed in chapter 6. These extensions are then tested. In particular we discuss test cases for the Poisson solver in chapter 7 and the particle kinetic model in chapters 8 and 9. We conclude with recommendations for future research to complete the project. Matters concerning the implementation in C++ of extensions (or proposed extensions) to the code are discussed in the appendices.

Chapter 2

Pseudospark Discharges

If a high enough voltage is applied across a gas between two parallel electrodes breakdown occurs. Free electrons in the gas, always present in very low numbers, are accelerated by the electric field and collide with the neutral atoms or molecules in the gas. If the electric field is high enough some of the electrons will succeed in ionizing a molecule before being lost on the positive electrode. If more electrons are released by ionization than lost at electrodes breakdown occurs; the gas is ionized and conducts electricity. The voltage at which electrical breakdown occurs is a function of the distance times the pressure. This function is called the Paschen curve (see figure 2.1)[5]. For more complex geometries the Paschen curve can be used as a first approximation [6]. Each gas has its own curve.

Different sections of the curve correspond with different types of discharges. At higher pressures discharge streamers are formed, slightly lower pressures yield glow discharges and at the left side branch of the curve pseudospark discharges occur. The term ”pseudospark discharge” was first coined by Christiansen and Schulteiss in 1979 [7]. For specific geometries, at low pressures and beyond a breakdown voltage, a diffuse discharge occurs which bears resemblance to spark discharges even though the regime in which these diffuse discharges occur differs from that of real spark discharges. Hence the name pseudospark discharge.

Using special geometries and/or insulating materials at the edges of the anode and cathode in this regime results in a diffuse discharge with quickly increasing currents. Currents rise rates can be as high as 10^{12} A/s[1]. If the anode contains a central hole an electron beam is emitted.

The pseudospark devices studied by Christiansen et al. were intended

[1] http://www.hcei.tsc.ru/ssi/techn/tech15_en.shtml

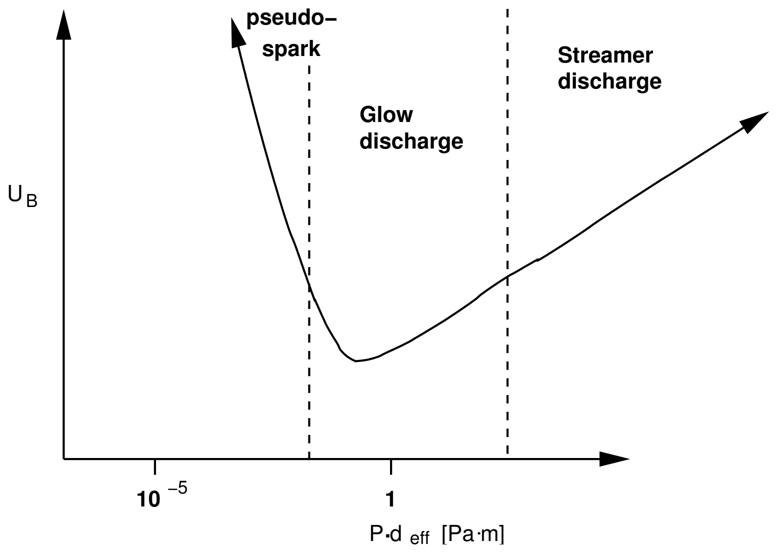
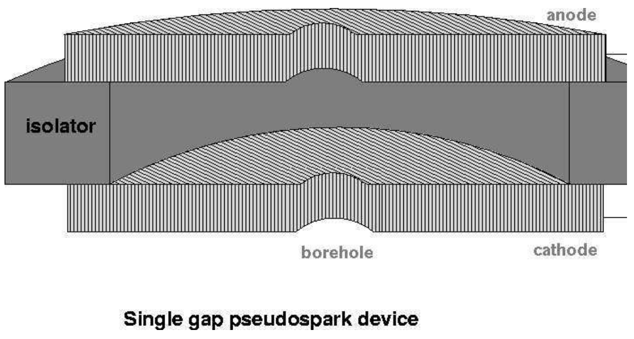


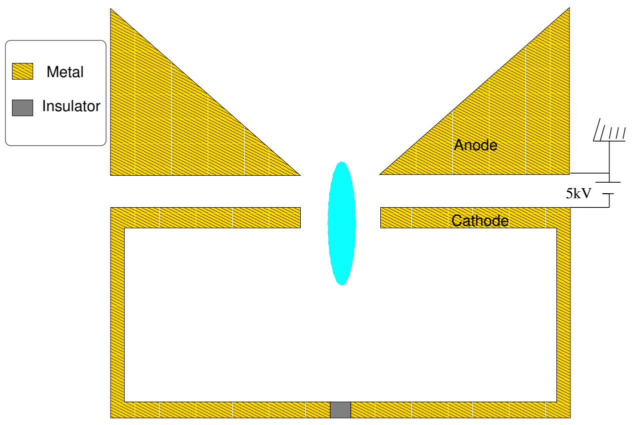
Figure 2.1: Typical Paschen curve.

for use as particle accelerators [7]. Stacks of them can be used to generate intense electron beams with densities exceeding 10^6 A/m^2 [7, 8]. A number of other applications exist. Because of their abrupt transition to a highly conductive state and relatively low jitter, they are frequently used in high-power switching applications [8, 9]. HCD's are also used for spectroscopic analysis [10]. More recently, they are being considered as a source of extreme ultraviolet (EUV) radiation. When filled with xenon, tin or lithium the gas rapidly becomes ionized to multiply ionized states. In xenon this can go to Xe^{12+}, emitting radiation in the band around 13.5 nm [4, 11]. This radiation can be used for lithography where the increasing demand for smaller structures requires shorter wavelengths[3, 11]. In figure 2.2 schematic drawings for some of these pseudospark devices are given.

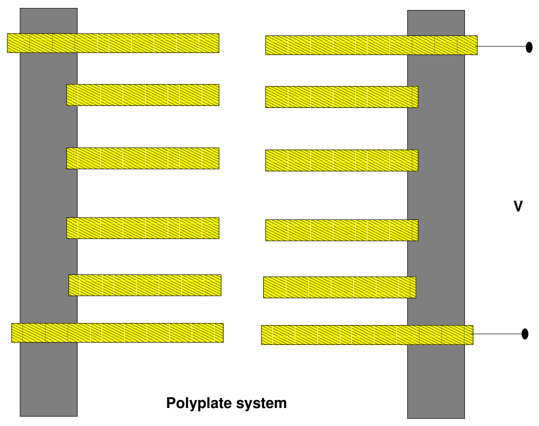
The main focus of this report will be to find modeling tools to describe the initial breakdown a of hollow cathode of the type being considered for the production of EUV radiation. Devices of this type are operated in a pulsed mode. Studying the operation directly before and during breakdown is of great interest in these devices. Operation in lithographic systems require a high reliability. Predictable and reliable startup times are needed to achieve a high repeat rate for the discharge. A high repeat rate is, in turn, necessary to achieve a sufficiently intense light production on the target.



1.



2.



3.

Figure 2.2: Several pseudospark devices. 1. A single gap device of the type investigated by Christiansen. 2. A Hollow cathode for EUV production. 3. A poly-plate system to produce an electron beam. The primary focus of the work presented in this document will be on type 2.

Chapter 3

Stages in the Operation Cycle of a Hollow Cathode

Pseudospark devices are operated in a pulsed mode. A capacitor bank is charged for every pulse, prior to discharge. Then the discharge is triggered, draining the capacitators, whereafter they are recharged. In general there are two methods of controlling the timing of the discharge. One is to apply a voltage just below the breakdown voltage and to use a trigger of some kind, either optical (by shooting a laser pulse through the discharge), or electrical to initiate the discharge. Another method to initiate the discharge is to apply a voltage greater than that required for spontaneous discharge and to use an electrical inhibitor. The inhibitor increases the voltage required for breakdown. The discharge is then initiated by releasing the inhibitor.

Each cycle can be broken down in a number of different phases. Bloess [8] describes two main phases in the breakdown based on experimental evidence, namely the pre-breakdown phase and the proper breakdown phase. This is expanded by Pitchford [9] to five. However, for the purpose of describing a full cycle, one needs to consider an additional last phase, the decay of the plasma. These six phases will be described next in order of their appearance, with the hollow cathode for EUV generation as the primary example of a pseudospark device.

1. *Townsend phase* At the start of the discharge the ionization degree is low, the ions and free electrons which are present result from the trigger, photoionization by background radiation, and/or ions left over from the last discharge. The electric field is determined by the geometry of the device as the total space charge from charged particles is negligible. As a result, the equipotential surfaces between anode and

cathode are nearly parallel to the anode. The field is large in the gap between the anode and cathode and is small inside the hollow cathode itself (see also figure 3a).

The strong electric field accelerates the electrons, whose mobility is much greater than that of the ions, towards the anode. These electrons are either propelled out of the device or absorbed by the anode. This results in a positive space charge. Secondary electrons emitted from the inside surface of the hollow cathode by ion impact contribute to the buildup of space charge by creating avalanches of ionization events. During this phase the mean free path of the electrons is long compared to the typical dimension of the device. This also means that the average number of ionization collisions a free electron causes before being absorbed by the anode is low. The high energy electrons travel only a short distance before being absorbed by the anode or accelerated out of the device. The electron multiplication factor

$$M = -\frac{\int_{anode} \vec{j}_e \cdot d\vec{s}}{\int_{cathode} \vec{j}_e \cdot d\vec{s}},$$

with $\vec{j_e}$ the electron current density, is low in this phase. In the case of triggered discharges: $M < 1 + \frac{1}{\gamma}$, where γ is the secondary electron-emission coefficient. In cases of self-breakdown or breakdown following the release of an inhibitor, M is somewhat larger than $1 + \frac{1}{\gamma}$.

2. *Plasma formation* The buildup of positive space charge distorts the electric field, drawing the equipotential lines into the hollow cathode (see figure 3b). The deformation of the electric field lengthens the path of the electrons from cathode to anode, increasing the electron multiplication factor. The field in the main gap is also decreased, thereby reducing the acceleration of the high energy electrons out of the cathode. A plasma is formed in the main gap.

3. *Pendulum effect* The term "pendulum effect" was first introduced by Günterschulze [12] to explain the greatly enhanced current in the HCD compared to a linear discharge of the same dimensions and voltage drop. The formation of a plasma in the hollow cathode creates a potential barrier for electrons (see figure 3c). This potential barrier accelerates electrons towards the center giving them an energy high enough to pass through the center of the hollow cathode and penetrate the sheath on the other side. There, the electron is again accelerated back towards the center of the hollow cathode. In effect the

potential barrier swings the electrons back and forth across the hollow cathode, hence the name "pendulum effect". The pendulum effect greatly increases the distance electrons travel before they collide with the wall. This confining effect increases the effective ionization rate as the chance of an ionization collision increases with the increase in the path of electrons through the vessel. The electron multiplication factor M increases by orders of magnitude. The resulting increase in the ionization degree causes the plasma sheath to contract.

4. *Plasma expansion* As the sheath contracts further, the electron multiplication factor drops somewhat. Although the sheath has relatively few electrons, the ionization in the sheath creates high energy electrons, since these are accelerated over part of the sheath potential. As the sheath contracts the total ionization in the sheath is reduced, thereby producing less high energy electrons. This reduces the ionization due to secondary electrons. However, the electron multiplication factor remains above the value required to sustain the plasma. The result is an ever increasing current. Since the hollow cathode keeps the ionization rate above the value to sustain the discharge, a steady state is not reached.

5. *Pinching* The magnetic field induced by the large currents in the plasma (a few kA) through the small borehole exert a Lorentz force on the electrons. The Lorentz force has the effect of contracting the current channel running through the plasma. The effect on the current density can be illustrated by the effect of two parallel wires with current running through them in the same direction. The increase of the current density due to the contraction from the Lorentz force further increases the Lorentz force, and so on. This effect is called pinching. In higher density plasmas the electron pressure finally stops the contraction, in low density plasmas the electrons simply miss each other in the center of the pinch (similar to the implosion of an incandescent bulb) and the pinch oscillates.

   In a pulsed HCD filled with xenon the energy density is high enough to yield ionization stages as high as Xe^{12+}[4]. The highly ionized xenon plasma emits EUV light, some (about 3%) of it in a band around 13.5 nm. The sheath inside the hollow cathode becomes much smaller than the dimensions of the device, creating large electric fields. For example, a voltage drop of 3kV across a gap of 100 μm yields an electric field of

3×10^7V/m. It has been suggested that thermionic emission[1] at the surface is responsible for a further rise in current.

6. *Decay* Eventually the runaway current drains the capacitor bank. Without an energy source the plasma decays due to recombination at the walls to which the charged particles diffuse. To a much lesser extent three particle recombination in the plasma also contributes to the plasma decay. The decay rate of the plasma also determines the electron densities at the start of the next cycle.

[1]When the electrons in the cathode wall have a high enough energy to pass over the surface potential barrier between the solid material of the cathode wall and the plasma, electrons are emitted from the surface. This is aided by the Schottky effect; where the electric field at the surface of the emitter reduces the surface potential barrier, thereby increasing the number of electrons able to escape from the surface.

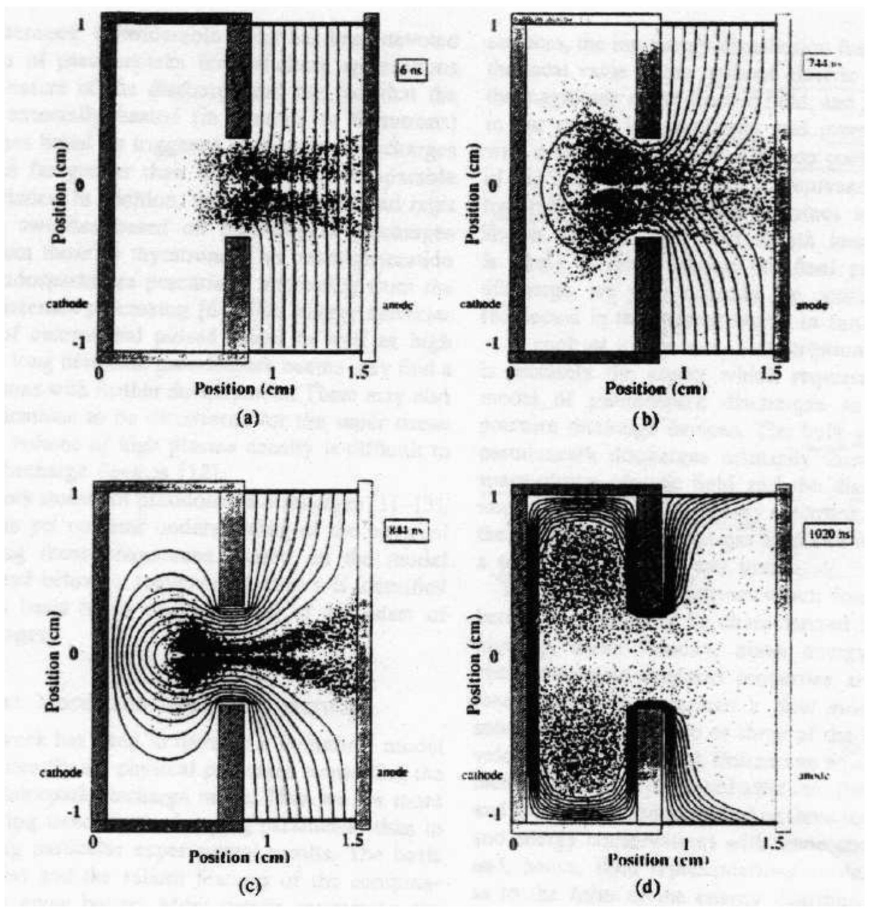


Figure 3.1: Four snapshots from a hybrid fluid-Monte Carlo model by Boeuf and Pitchford [9] roughly corresponding to phase 1 through 4 of the discharge. The anode potential in the model is 2kV. The gas is helium at a pressure of 0.5 torr (67 Pa). The lines show equipotential surfaces 200 Volts apart. The dots represent multiple ionization events. The model assumes an initial injection of electrons (the trigger). The snapshots shown are: (a) t=6 ns (Townsend phase), (b) t= 744 ns (Plasma formation phase), (c) t=844 ns (the pendulum effect) and (d) t=1020 ns (plasma expansion in the hollow cathode and sheath contraction). Pinching, thermionic emission and decay are not included in the model.

Chapter 4

Existing Models of Hollow Cathode Discharges

Because of the transient nature of pulsed HCD's and the large dynamic range of parameters, such as the electron density, one cannot hope to use the same method for an efficient description of all phases of the discharge. For different phases of the discharge different approximations are appropriate. An approximation which is valid at one stage of the discharge may well not be valid for another stage. Not using approximations valid for some stages for the sake of keeping the model general for all would make the model slow, inefficient and even incorrect.

A number of different approaches to model HCD's exist. In the following we will list the different models which have been applied to various phases of the discharge. An overview of different approaches and codes for modeling pinched plasmas and HCD's is given by Garloff [13].

To study the early phases of the discharge several different approaches exist. An analytical model has been developed by Kolobov and Tsendin [14]. This model assumes continuous constant energy losses for the fast electrons. Particle losses and electron scattering are not included in the model. The energy distribution function (EDF) for the fast electrons is calculated for an inhomogeneous field. The electric field in the model comes from a sheath approach. Using this model the pendulum effect and the electron multiplication factor are quantitatively described. Work on calculating the EDF for both fast and slow electrons has also been done by Arslenbekov et al. [15, 16].

Particle In Cell- Monte Carlo Collisional (PIC-MCC) methods have also been used to model the early phases of the discharge, one example being the work by Cai and Striffler [17]. Such methods require less restrictive assumptions and can therefore be used to describe more characteristics of the discharge than what is possible with analytical models. For high electron densities, however, the computation time makes these models impractical. Additionally PIC-MCC Models tend to have problems with statistical heating [18][1]. The latter problem can be avoided by not taking the space charge into account at all. An example of this approach is a Monte Carlo collisional model by Kushner et al [19] which describes the Townsend phase of the discharge and uses Laplace's equation to solve for the electric field.

The hollow cathode or pendulum effect (phase 3 in the previous chapter) cannot be modeled by a local field approximation. In the local field approximation the plasma is described by parameters which are a function of the local electric field only. In the local field approximation, for instance, the ionization in one part of the plasma depends only on the reduced electric field at that location and not on the reduced electric field elsewhere.

This assumption does not hold in the HCD because the pendulum effect causes the ionization in the center of the plasma to be influenced by the electric field at the walls (the pendulum effect is essentially non-local). Furthermore, the ionization rate is determined by the tail of the electron energy distribution. The low densities and large electric fields in the hollow cathode cause a significant deviation from a Maxwellian energy distribution, which is most pronounced in the tail. As stated before, in the previous chapter, the ionization rate does not reach equilibrium. Indeed the absence of an equilibrium for the ionization and its dependence on the non-local parameters are of prime importance in the description of the hollow cathode effect. A (multi-)fluid approach cannot, therefore, describe the hollow cathode effect. However, a fluid approach can be used to model certain other aspects of the discharge. The pinching of the plasma is a local effect. Densities in the pinch phase are high enough to warrant a description in terms of average properties such as mass, momentum and energy. Pinched plasmas have been described by MHD (Magneto-Hydro Dynamic) models. In these models the plasma is described by a set of transport equations coupled to Maxwell's equations for the magnetic field. MHD models have mostly been developed to study fusion plasmas, where the magnetic field is used to confine the plasma. MHD codes tend to use a Local Thermal Equilibrium (LTE)

[1]The charge density, given by the projection of the position of all charged particles on a grid or mesh has random statistical fluctuations. These cause an error in the electrical field. The energy of the particles is roughly proportional to the square of the electrical field. Consequently the random fluctuations in the electric field result in an overestimation of the mean energy of the particles.

approach[13]. This approach will not work for HCD's as the electron temperature in these devices is more than an order of magnitude greater than the temperature of the heavy particles and the process is far from LTE.

The decay phase can also be treated as a fluid. Work on modeling the decay phase has been done by Broks [20]. Care does need to be taken, however, with the treatment of the diffusion at low densities. The mean free path of the electrons is not small compared to the typical dimensions of the hollow cathode. The model used in [20] uses a correction for Knudsen type diffusion to estimate the decay time of the hollow cathode.

To best combine different approaches and models a hybrid model is needed. Such a hybrid model combines fluid or modified fluid approaches with MCC models. Because of computational and practical limits a PIC-MCC model cannot be expected to work far beyond the initial Townsend phase. Hybrid fluid-Monte Carlo models, such as those used by Pitchford et al [9], have been used to describe the HCD upto the plasma expansion phase. Broks modified fluid approach with a "Knudsen" correction can be used for the decay phase. The pinching fase can be described with a fluid approach together with a module solving Maxwell's equations for the electric and magnetic fields to arrive at the Lorentz force on charged particles. For all phases but the decay phase MCC models are needed to determine the ionization rate.

Chapter 5

Hybrid Fluid-Monte Carlo Models

This chapter explains in a more detailed manner the hybrid fluid-Monte Carlo models discussed in the previous chapter. Both fluid and particle kinetic models are based on the Boltzmann transport equation. The Boltzmann transport equation, in turn, relies on a continuum approach. The difference between the two lies in the assumptions made on the distribution of the particles in phase space. The fluid models rely on a priori assumptions on the distribution of the particles in velocity space; leaving the distribution dependent on one or more parameters. Particle kinetic models make no such assumptions. Instead they seek a solution of the distribution function by tracking ensembles of particles. Thus, it is not for lack of enough particles to form a continuum that a particle kinetic model is used in favor of a fluid model, but because not enough information is available on the shape of the distribution function. Hybrid models use a fluid approach to model some macroscopic properties and a particle kinetic approach for other macroscopic properties. In the hybrid models used in the modeling of HCD's, the ionization rate is determined from a particle kinetic model, while other properties are determined from a fluid model. This is because the former is particularly sensitive to the shape of the distribution function.

5.1 The Boltzmann Transport Equation

Fluids (in the general sense of the word, meaning both liquids and gasses) consist of large collections of particles. For example, at the low pressure of 50 Pascal, a cube of 1 mm^3 filled with gas at a temperature of 300 Kelvin contains approximately 1×10^{13} particles. It is impossible to follow the temporal behavior of each of these particles. Fortunately, the very fact that the number of particles is so large allows one to predict the behavior of the system of particles without detailed knowledge of the behavior of each individual particle (see for example the discussion in [21]).

To describe the behavior of a collection of particles in terms of the behavior of the individual particles, and vice versa, the probability distribution function is introduced[1]. The function

$$f_p\,(\vec{x},\vec{v},t)\,d^3x\,d^3v\;.$$

is defined as the number of molecules of a species p at a time t with positions within the volume element $d^3x\,d^3v$, centered around $\vec{x},\vec{v}$. The volume element in this six dimensional phase space is small, but not infinitesimal. Each volume element, therefore, contains a large number of particles. This also means the distribution function should be sufficiently smooth on dimensions of this scale. Molecules (or atoms) in a different quantum mechanical state are seen as molecules of a separate species. The subscript p will be omitted in the rest of this section, but bear in mind that each state is represented by a separate distribution function.

This function is normalized so that the distribution function integrated over the entire phase space yields the total number of particles N at the time t [2]:

$$N(t) \;=\; \iint f(\vec{x},\vec{v},t)\,d^3x\,d^3v\;,$$

If the molecules or particles form a gas in a external force field $\vec{F}$ it can be shown that the distribution function for each species satisfies the continuity equation [22], p 57:

$$\frac{\partial f(\vec{x},\vec{v})}{\partial t} + \nabla_\psi \cdot \left(\vec{v}f(\vec{x},\vec{v})\right) \;=\; \left(\frac{\partial f}{\partial t}\right)_{int}\;, \tag{5.1}$$

where the term $\left(\frac{\partial f}{\partial t}\right)_{int}$ is due to interactions with other species. It is defined as:

$$\left(\frac{\partial f}{\partial t}\right)_{int}\delta t \;=\; f\left(\vec{x}+\vec{v}\delta t, \vec{v}+\frac{\vec{F}}{m}\delta t, t+\delta t\right) - f(\vec{x},\vec{v},t)\;,$$

[1] This particle distribution function is completely classical. The statistical nature of the distribution function comes from the lack of knowledge about the system, not from the non-commensurability of momentum and position. In a quantum mechanical approach a distribution function for the density of states is used.

[2] Note that this assumes a continuum, both in replacing the sum over all volume elements with an integral and in the definition of the distribution function.

with δt an infinitesimal time-step. Equation (5.1) is valid per species, where a species should be seen as the ensemble of particles in the same state. The notation ∇_ψ is used for the gradient in phase space. The term $\nabla_\psi \cdot (\vec{v} f(\vec{x}, \vec{v}))$ can be expanded to

$$\vec{v} \cdot \nabla_x f(\vec{x}, \vec{v}) + \frac{\vec{F}}{m} \cdot \nabla_v f(\vec{x}, \vec{v}),$$

Provided the condition $\nabla_v \cdot \vec{F} = 0$ is met, with ∇_v the derivative $\frac{\partial}{\partial v_i}$. The Boltzmann Transport Equation (BTE) in 5.1 serves as a basis to describe macroscopic properties of the fluid or gas. By integrating over velocity space one derives equations such as the mass, momentum balance, and energy balance. The macroscopic properties such as the mass and current density are retrieved from the probability distribution function by integration over the velocity space for every species. Ohm's law results from the integration of the BTE and subsequent summation over all charged particles.

In order to solve the BTE one needs more information on the interaction term. The interaction term represents collisions with particles of the same species, particles of different species and radiation events. One simple approach involves ignoring the term and treating collisions through the $\vec{F}/m$ term. In the Vlasov system the only collisions treated are long range Coulomb interactions, through a field approach. The charged particles form an electric field which acts as a volumetric force. The right hand side of the BTE is zero in this approach.

The electro-magnetic force on a charged moving particle is given by:

$$\vec{F} = q\left(\vec{E} + \vec{v} \times \vec{B}\right),$$

where the electric field $\vec{E}$ and the magnetic field $\vec{B}$ are given by the Maxwell equations:

$$\begin{array}{rclcrcl} \nabla \times \vec{B} & = & \mu_0 \vec{j} + \frac{1}{c^2}\frac{\partial \vec{E}}{\partial t}\,; & & \nabla \cdot \vec{B} & = & 0 \\ \nabla \times \vec{E} & = & \frac{\partial \vec{B}}{\partial t}\,; & & \nabla \cdot \vec{E} & = & \frac{\rho}{\epsilon_0} \end{array} \tag{5.2}$$

with q the charge of the particles, ρ the charge density and $\vec{j}$ the current density. The latter two are the result of the position of the charged particles in phase space. A graphical representation of the Vlasov system is shown in figure 5.1.

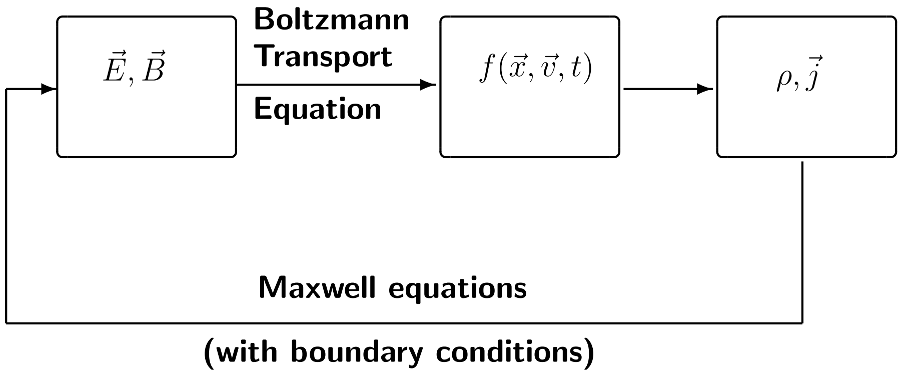


Figure 5.1: Graphical representation of the Vlasov system from [23], p 112. Collisions are absent in this system, all interactions take place through self consistent electric and magnetic fields.

5.2 Fluid Models

The method commonly used [22], p97, [23], p 112 to simplify the problem of solving the Boltzmann Transport Equation (BTE) is to consider quantities (usually mass, momentum and energy) which are rigorously conserved; that is, the sum of the loss and production terms over all species must be zero. By using the conservation theorem:

$$\frac{\partial}{\partial t}\langle n\chi\rangle - n\left\langle v_i\frac{\partial \chi}{\partial x_i}\right\rangle - \frac{n}{m}\left\langle F_i\frac{\partial \chi}{\partial v_i}\right\rangle - \frac{n}{m}\left\langle \frac{\partial F_i}{\partial v_i}\chi\right\rangle \;=\; S, \tag{5.3}$$

with χ a conserved property and S the source term. The average of a quantity A over velocity space is represented by

$$\langle A\rangle(\vec{x},t) = \frac{1}{n}\int d^3v A(\vec{x},\vec{v},t) f(\vec{x},\vec{v},t),$$

where

$$n(\vec{x},t) = \int d^3v f(\vec{x},\vec{v},t)$$

is the particle density.

With χ set to the mass, momentum and thermal energy one gets the familiar conservation equations:

$$\frac{\partial \rho_m}{\partial t} + \nabla\cdot(\rho_m\vec{u}) = \left(\frac{\partial mn}{\partial t}\right)_{int}, \tag{5.4}$$

$$\rho_m \left(\frac{\partial}{\partial t} + \vec{u} \cdot \nabla \right) \vec{u} + nm\vec{a} + \nabla \cdot \mathbf{P} =$$
$$\int m\vec{v} \left(\frac{\partial f}{\partial t} \right)_{int} - m\vec{u} \left(\frac{\partial n}{\partial t} \right)_{int} d^3r d^3v, \tag{5.5}$$

$$\frac{3}{2} \frac{\partial k_B T}{\partial t} + \frac{3}{2} \nabla \cdot (nk_B T\vec{u}) + (\nabla\vec{u}) : \mathbf{P} + \nabla \cdot \vec{q} =$$
$$\int \frac{1}{2} mv^2 \left(\frac{\partial f}{\partial t} \right)_{int} - \vec{u} \cdot \vec{v} n \left(\frac{\partial f}{\partial t} \right)_{int} d^3r d^3v, \tag{5.6}$$

with the derived quantities:

$\rho_m(\vec{x}, t)$	$=$	$m \int f(\vec{x}, \vec{v}, t) d^3v$	(mass density)	(5.7)
$\vec{u}(\vec{x}, t)$	$=$	$\langle \vec{v} \rangle$	(average velocity)	(5.8)
$T(\vec{x}, t)$	$=$	$\frac{1}{3} m \left\langle \|v - u\|^2 \right\rangle$	(temperature)	(5.9)
$\vec{q}(\vec{x}, t)$	$=$	$\frac{1}{2} m\rho_m \left\langle (\vec{v} - \vec{u})\|v - u\|^2 \right\rangle$	(heat flux vector)	(5.10)
$\mathbf{P}$	$=$	$\rho_m \langle (\vec{v} - \vec{u})(\vec{v} - \vec{u}) \rangle$	(pressure tensor)	(5.11)

[20], see also for a complete derivation from the BTE. These equations are themselves generally too complicated to be of practical use. They need to be simplified further. In section 6.3 a simplified version of the momentum equation (5.5) is discussed. More simplifications exists for the equation for the energy balance. This is not discussed in this report, but more on this subject is found in [24]. In addition, the set of equations in (5.4) through (5.6) is not closed. Each moment of the BTE contains a term with the next moment. Going to higher moments will not solve this problem as the pattern continues. One requires a simplification of one of the equations so that it no longer depends on the next moment of the BTE.

Additionally, one must solve the balances for each species. Typically though, the balances for short lived species can be greatly simplified, as can the energy balance for species of similar mass. The level of deviation from thermodynamic equilibrium is crucial to determining how many balance equations and macroscopic quantities are necessary to characterize a plasma. For example, a plasma in local thermal equilibrium can be fully characterized with just the pressure, elemental composition and temperature, while a plasma in partial local Saha requires two additional parameters- the electron temperature and the electron density (see also [23], p. 112 for a more complete discussion).

Note also that the derivation of the conservation equations from the moments of the BTE does not rely on a Maxwellian distribution function.

One could replace the temperature in (5.6) with the average energy. In writing down the equation though, the implicit assumption is made that the plasma can be described fully in terms of parameters averaged over velocity space. This requires an a priori assumption on the form of the distribution function.

5.3 Particle Kinetic Models

Particle kinetic models make far less restrictive assumptions on the nature the distribution function of the species. Instead they seek a solution to this distribution function. The Vlasov system described earlier is one example of such a kinetic model. However, the Vlasov system is collisionless so that the right hand side of the BTE is zero, greatly simplifying the equation. In plasmas which cannot be treated as collisionless the problem arises that phase space is "turbulent". An ensemble of particles occupying a small portion of phase space is smeared out over a large part of phase space after just a few collisions. To solve the problem Monte-Carlo methods are used. The term *Monte Carlo method* in physics refers generally to any method whereby a limited number of processes are selected randomly out of large number of possible processes. The processes selected are then held to be representative of all possible processes.

In particle kinetic models samples of the distribution function, called "super-particles" are taken. Super-particles represent large ensembles of particles occupying a finite but small volume of phase space. The probability distribution function of this ensemble of particles is approximated by a delta function centered around that part of the phase space. The super-particles are accelerated by the electric field and/or magnetic field. In Particle in Cell (PIC) models the positions of these particles are mapped on a grid to arrive at particle densities. These particle densities can then be used to calculate the electric field by using the Poisson equation.

PIC-MCC (Particle In Cell/Monte Carlo Collisional) models combine the use of super-particles to represent ensembles of particles with Monte Carlo methods to represent collisions. Short range interactions are simulated by randomly selecting one of a range of possible interactions at random times reflecting the frequency distribution of those interactions. By using large numbers[3] of super-particles the interactions the particles would undergo are simulated by the interactions of the super-particles. The number of super-particles typically determines the number of real particles per super-particle.

[3]typically of the order of 10^4 to 10^6

Long range interactions, such as Coulomb interactions, are calculated by projecting the positions of the charged particles on a grid. The charge density is then calculated as the weight of the super-particle (the number of particles it represents) times the charge of the species smeared out over one or more cells on the grid. Hence the name Particle In Cell. The electric field is calculated by solving the Poisson equation. A similar approach is used to calculate the magnetic field. PIC models are known to have several drawbacks. In order to solve the Poisson equation accurately a dense grid is needed. Getting an accurate estimation of the charge density on that grid requires a large number of super-particles per cell. PIC models are, therefore, computationally expensive.

Other methods exist, which also combine the use of super-particles to represent ensembles of particles and Monte Carlo methods to represent collisions or other short range interactions but do not make use of grids. These have advantages for open systems and systems where the space charge distribution has extreme gradients, or is distributed in a highly non-uniform manner but tend to be computationally expensive and difficult to implement. For the purpose of studying confined plasmas PIC-MCC methods are far more common than grid-less methods. In the hybrid model discussed next grids are also used, though the potential is not calculated from the charge distribution of the super-particles.

5.4 Hybrid Models

In some cases a mix between fluid and particle kinetic models is used. The pendulum effect in the hollow cathode cannot be described by a pure fluid model, but some aspects of the discharge can. In a hybrid model some aspects of the problem are described by a particle kinetic model while other aspects are described in terms of average quantities, in a fluid model. The models used by Pitchford and Baguaer describe the fast secondary electrons with a particle kinetic model, whereas the heavy particles and the bulk of the electrons are described with a fluid model. The ionization, which is extremely sensitive to the tail of the energy distribution function for most plasmas, is described by a particle kinetic model of those electrons with an energy above the ionization threshold. The particle kinetic model of these high energy electrons yields an ionization rate which is used in the fluid model for the heavy particles. The fluid model provides a "background gas" for the high energy electrons to collide with. The fluid model also provides the electric field. The net charge density is computed by the sum of the product of the density and charge for all particles in the fluid model. The potential then follows by solving Poisson's equation. This approach is valid as long as the number of electrons in the particle kinetic model is small compared to the charge density in the fluid model. If this is not the case a PIC-like technique is required to account for the charge of the particles in the particle kinetic model. The schematics of the interaction between the fluid and particle kinetic models in the hybrid model is given in figure 5.2.

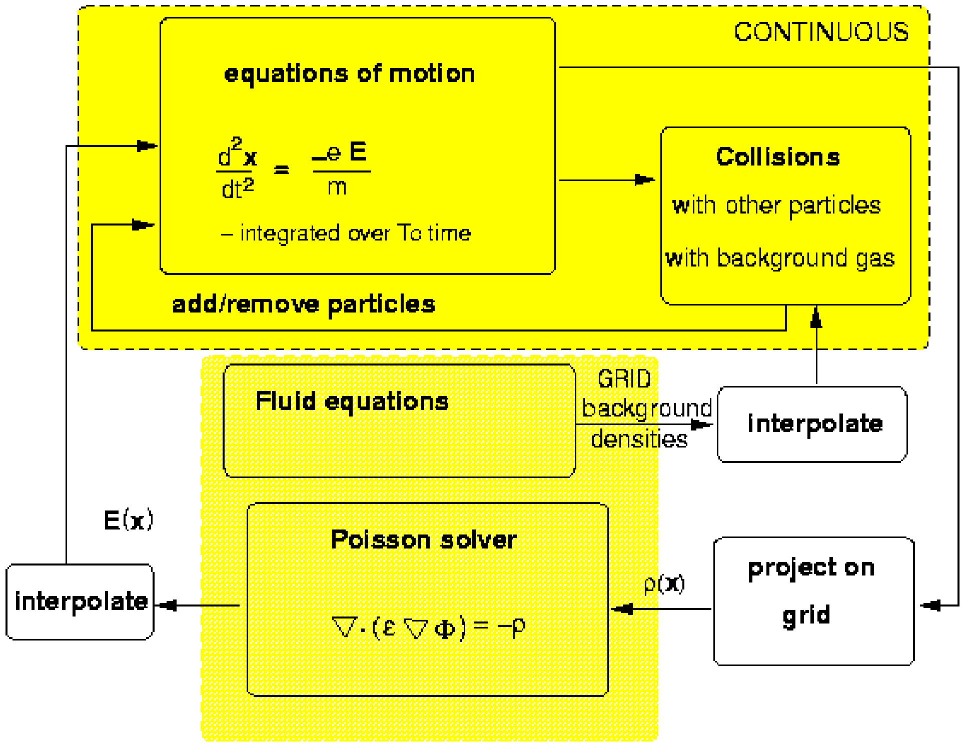


Figure 5.2: A schematic overview of a hybrid model stressing the fact that the particle sub-model is grid-less, while the fluid model is discretized on a grid.

Chapter 6

Extensions to PLASIMO

We wish to construct a hybrid fluid-Monte Carlo code to model as many phases of the discharge as possible. It is generally more efficient to make use of available and pre-existing code rather than write a separate dedicated application to model a problem. In the EPG group two modeling platforms are being developed: PLASIMO and MD2D. The Micro Discharge 2D (MD2D) code was written by Gerjan Hagelaar to model micro discharges for display technology. The code is now being further developed by the PLASIMO team, most notably by Jan van Dijk and Wouter Brok. MD2D is a time dependent model and uses a finite volume method on orthogonal grids. Though it is limited to 2D applications it has considerable flexibility in the geometry. The code solves transport equations for charged particles separately using the drift-diffusion equation and the continuity equation. Since the drift-diffusion equations for the electrons is solved as well, one is not confined by the assumption of quasi-neutrality. The code assumes a dominant background gas whose temperature is known. The temperature of all other species besides electrons are assumed to equal to the temperature of the background gas. The presence of a background gas whose composition and temperature is not effected by the other particles poses a problem for the modeling of hollow cathodes for EUV generation. It is for this reason that the author chose to adapt another platform in use at the EPG group: PLASIMO . This choice is made in spite of the advantages the Micro-dis code has in other areas, such as the absence of the assumption of quasi-neutrality and the capability of handling more complex geometries.

6.1 PLASIMO

PLASIMO (PLAsma SImulation MOdel) is a modeling toolkit for the simulation of low temperature plasmas[1]. The code was originally designed by Benoy [24] to model inductively coupled argon plasmas. Researchers working under the supervision of Van der Mullen contributed, and continue to contribute, to the code as they work on specific applications. The code underwent a major redesign in 1998, using object oriented programming techniques to make the code more orthogonal[2] and robust to extensions [25], p 14,15.

PLASIMO is modular in design. The sub-models for solving transport problems are based on finite volume methods on orthogonal or ortho-curvilinear grids. Problems are limited to two dimensions, where the third coordinate can be either Cartesian or azimuthal. See [26], chapter 16 for more information on the numerical grids used in PLASIMO .

PLASIMO assumes a Maxwellian distribution function, with the additional feature of describing the deviation from this Maxwellian distribution with different temperatures in the tail. The code also assumes quasi-neutrality. The electron density is calculated from the sum of the ion densities. More information on the features and limitations, are found in [25][3]. Using PLASIMO as the basis for a hybrid fluid-Monte Carlo code has the advantage of reusing tried and tested code developed for a large variety of electrically operated plasma light sources. A major advantage over MD2D is that it solves energy balance equations for all species without assuming a single dominant background species of fixed temperature. It has some disadvantages over MD2D in terms of geometry and the assumption of quasi neutrality. Additionally, the code has not been designed to use with Monte Carlo methods. However, the modular structure of the code is such that it facilitates the addition of extensions.

Constructing a hybrid model with the use of PLASIMO implies that the following additions to the code are required:

1. A diffusion model that solves the transport equation for the electrons rather than inferring the electron density from the ion densities,

2. A module to solve the Poisson equation given the net space charge,

3. A kinetic Monte Carlo model to provide the ionization rates,

[1]as opposed to fusion plasmas

[2]in the sense that components can be used independently of each other

[3]The code has continued to develop since this publication. The PLASIMO website http://plasimo.phys.tue.nl contains more current information.

4. A module simulating secondary electron emission from the ion flux to the walls as input from the fluid part of the model (which treats the ions) to the kinetic part of the hybrid model,

5. Methods to project the grid-less positions of the particles in the kinetic module on the grid in the fluid sub-model,

6. An expansion to the fluid model to allow for more complicated geometries. PLASIMO requires rectangular regions in coordinate space. More complicated geometries are handled by using ortho-curvilinear coordinates to map these problems onto a rectangular, orthogonal grid. Ortho-curvilinear grids, however, cannot be used together with a grid-less method as there is no general analytical mapping of all of the Cartesian space on the two dimensional ortho curvilinear grid. Projecting a point in 3D Cartesian space on the curvilinear grid requires that one solve a partial differential equation [26]. Thus, using ortho-curvilinear grids for the fluid part of a hybrid model would be prohibitively computationally expensive. Therefore, hybrid models with more complicated geometries require a different approach.

The main focus on the remainder of this chapter will be a self consistent approach to the transport and diffusion of charged particles in a plasma without the assumption of quasi-neutrality and the extension of the fluid model towards anisotropic diffusion. The method used to solve the Poisson equation is discussed in the next chapter. The problem of projecting points from the grid-less model on the cells of the grid is dealt with in appendix A on coding issues. Appendix B contains a proposal on how to deal with more complicated geometries.

6.2 Removing Quasi-neutrality

The central region of a discharge is usually assumed to be quasi-neutral. That is, the density of the electrons is determined by the sum of the density of all the ions multiplied with their charges

$$n_e = \sum_i n_i Z_i,$$

with n_e the density of the electrons, and n_i the density of the positive ion with charge number Z_i. The assumption of quasi-neutrality comes from the shielding effect of the plasma. A small difference between the density of the ions and electrons yields large electric fields, which can compensate externally applied electric fields. Deviations from quasi-neutrality are then assumed to occur only in a small region near the boundaries called the sheath. Large deviations from quasi-neutrality also occur on a scale smaller than the Debye length [23], p11. The quasi-neutral approach is valid when the ratio of the applied electric field to the electron density is small.

In a thought experiment in [23] an area of positive space charge of 1mm thickness with no electrons, in an otherwise uniform one dimensional plasma creates an electric field of 10^5 V/m at an ion density of 10^{16} m^{-3} (in the neutral region, see also figure 6.1) and an electric field of 10^9 V/m at an ion density of 10^{20} m^{-3}. It is clear that electric fields of such magnitude will quickly accelerate charged particles to reduce the space charge.

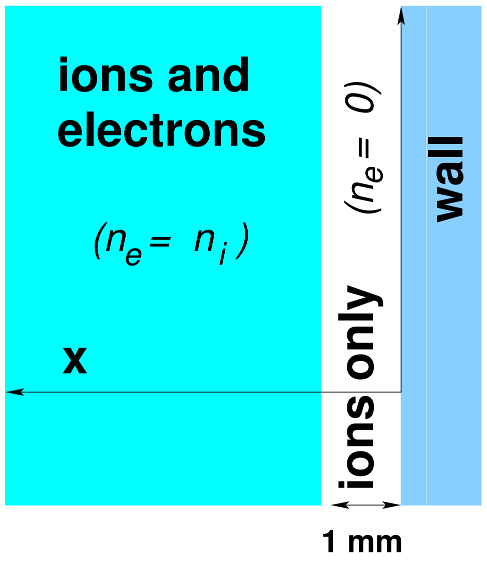


Figure 6.1: Schematic drawing of the one dimensional positive space charge region in the plasma mentioned in the thought experiment from [23]

If the applied electric field is large the space charge region created by the plasma to shield itself from the externally applied field can be significant, particularly at low electron densities. This effect needs to be taken into account in the initial phases of the discharge as the hollow cathode has several kilovolts of potential difference across a gap of a few millimeters. For a given constant field the thickness of the space charge region must decrease for increasing ion densities.

Although quasi-neutrality is a common assumption in literature for higher density plasmas, and has been built into PLASIMO from the start, it is not valid for low pressure cathode discharges. In the literature for HCD's quasi-neutrality is usually not assumed [9, 10].

Another reason to remove the assumption of quasi-neutrality from PLASIMO is the need for an accurate estimation of the local electric field. In a hybrid model the kinetic sub-model needs the local electric field as input. In the conditions in the startup fase of the HCD the sheath approach is not valid as the sheath is still in the process of formation. This means that one cannot estimate the local electric field from the ambipolar field and an externally applied electric field. The electron density is low (of the order of 10^{14} m^{-3}) and the applied electric field is of the order of several kV/cm.

The densities of the ions and electrons have to be calculated in conjuction with the electric field rather than assuming quasi-neutrality to calculate the electron density given the ion density. The electric field follows from the solution of the Poisson equation. This electric field, in turn, causes a net drift of charged species in the plasma.

Once a space charge region has been formed another approach is possible, where one keeps track of the position of the sheath edge and assumes quasi-neutrality within the central region of the discharge. An example of this approach is found in [27].

The simplest and most straightforward approach to solve for the space charge distribution is to use the drift-diffusion equation for each charged particle given the electric field and the proper diffusion and mobility coefficients. An example of this approach is to be found in [10], although one should take into account that the discharges studied are at much lower voltages (300V) and slightly higher pressure (0.65 torr). The validity of this approach, for the regime in which the hollow cathode for EUV production is operated, is examined next, starting with the derivation of the drift-diffusion equation.

6.3 The Drift-Diffusion Equation

The system of equations in (5.4), (5.5) and (5.6) is still equivalent to the BTE (5.1). However the difficulty in solving the equations has simply been moved to the determination of the derived quantities in (5.7) through (5.11). In the fluid approach one assumes that the system can be fully described in terms of properties averaged over velocity space. This requires an a priori assumption on the form of the distribution function. The prime example would be a Maxwellian distribution, but Druyvestein or two-temperature distributions can also be used as long as one has a function dependent on quantities averaged over the velocity space. For many practical purposes the equations in (5.4) through (5.6) are still too complex to handle. To calculate the drift of particles due to density gradients and (externally applied) electric fields the drift-diffusion equation is often used. This is a simplified form of (5.5).

Let us examine the speed $\vec{u_p}$ of a species p in the situation where the weighted sum of the velocity of all species (the bulk flow) $\sum_p \rho_p u_p / \sum \rho_p$ is zero[4]. Using conservation of momentum one may write:

$$\frac{\partial \rho_p \vec{u}_p}{\partial t} + \nabla \cdot (\rho_p \vec{u}_p \vec{u}_p) = -\nabla \cdot \mathbf{P}_p + \vec{u_p} S_p + \vec{F_p} + \sum_j \vec{F}_{pj}, \tag{6.1}$$

with $\mathbf{P}_p$ the pressure tensor, $\vec{F_p}$ the force per unit volume on species p and $\sum_j \vec{F}_{pj}$ the forces on species p as a result from the interaction with all other species [28]. The momentum balance can be written in a more familiar form by subtracting the mass balance (5.4 multiplied with the velocity from it.

$$\underbrace{\rho_p \frac{\partial \vec{u}_p}{\partial t}}_{1} + \underbrace{\rho_p \left(\vec{u}_p \cdot \nabla\right) \vec{u}_p}_{2} = \underbrace{-\nabla \cdot \mathbf{P}_p}_{3} + \underbrace{\vec{F_p}}_{4} + \underbrace{\sum_j \vec{F}_{pj}}_{5}, \tag{6.2}$$

where the first term and second term (6.1) have been expanded with the product rule. In particular we make use of the relation

$$\nabla \cdot (\rho_p \vec{u}_p \vec{u}_p) = \vec{u}_p \nabla \cdot (\rho_p \vec{u}) + \rho_p \left(\vec{u}_p \cdot \nabla\right) \vec{u}.$$

We will now examine each term in the equation to arrive at the simplified form of the momentum balance known as the *drift-diffusion equation*.

term 1 The drift-diffusion equation is quasi-static. The time dependent term is ignored. This is valid provided the time scales of interest are much longer than the relaxation times of the momentum exchange interactions. The validity of this approximation for the hollow cathode will be reexamined in chapter 9.

term 2 The term $\rho_p \left(\vec{u}_p \cdot \nabla\right) \vec{u}_p$ is assumed to be small compared to ∇p and is also neglected. The ratio ∇p can be approximated as $\rho u_{therm}^2 / L$,

[4] For the case in which the bulkflow is not zero, $\vec{u}_p$ should be replaced by the relative velocity to the bulkflow. The derivation then follows along the same lines. See [28], section 7.2.

with u_{therm} the random thermal velocity and L the typical dimension of the plasma. Neglecting the term $\rho_m\left(\vec{u}\cdot\nabla\right)\vec{u}$ is, therefore, valid as long as u_{therm} is much greater than the systematic velocity.

term 3 Without loss of generality, the pressure tensor $\mathbf{P}_p$ can be split into a scalar pressure times the identity matrix and a viscosity tensor $\mathbf{\Pi}_i$ ($\mathbf{P}_p = p_p\mathbf{I} - \mathbf{\Pi}_p$). In the absence of significant temperature gradients the term $\nabla\cdot\mathbf{P}$ simplifies to $kT_p\nabla n_p$. The viscosity tensor term is neglected as well. This is valid for sufficiently dilute plasmas.

term 4 The only volumetric force considered is that on charged particles due to the electric field, gravity and the Lorentz force are neglected.

term 5 The absence of temperature gradients also allows one to ignore the thermophoretic part of the last term in 6.1. The sole contribution to the force between particles then comes from the friction force:

$$\vec{F}_{pj} = m_{pj}n_pn_j\Omega_{pj}(\vec{u}_j - \vec{u}_p),$$

where Ω_{pj} is the rate coefficient for momentum transfer between the species and m_{pj} is the reduced mass of the system.

Under the previous assumptions the momentum equation reduces to :

$$kT_p\nabla n_p = n_pq_p\vec{E} + \sum_j m_{pj}n_pn_j\Omega_{pj}(\vec{u}_j - \vec{u}_p). \tag{6.3}$$

Solving for the velocity $\vec{u}_p$ yields:

$$\vec{u}_p = \frac{n_pq_p}{f_p^{eff}}\vec{E} + \frac{kT_p}{f_p^{eff}}\nabla n_p + \frac{\sum_j m_{pj}n_pn_j\Omega_{pj}\vec{u}_j}{f_p^{eff}}, \tag{6.4}$$

with $f_p^{eff} = \sum_j m_{pj}n_pn_j\Omega_{pj}$, the effective friction force for the species p. In the case where there is one dominant species with a density much greater than that of other species the last term can be ignored. The result is the familiar drift-diffusion equation

$$\vec{u_p} \;=\; \mu_p\vec{E} + D_p\nabla n_p/n_p, \tag{6.5}$$

with

$$\mu_p = \frac{n_pq_p}{f_p^{eff}}$$

the mobility and

$$D_p = \frac{n_p k T_p}{f_p^{eff}}$$

the diffusion coefficient. The Einstein relation ($\mu = qD/(k_B T)$) automatically follows from (6.5), without any further simplifications.

In deriving the drift-diffusion equation some restrictive assumptions have been made. A more general and self-consistent approach for diffusion is followed in [28], but at the low densities in the hollow cathode it is not so much the stated assumptions in the previous derivation that are cause for concern as the underlying fluid approach. Runaway electrons can give rise to an electron beam that has little in common with the concept of a drift velocity or indeed a fluid.

6.3.1 Small Anisotropy

The drift-diffusion equation can be derived more generally without assuming a Maxwellian distribution by the classical two term expansion of the BTE combined with the *local mean energy approximation* and the *small anisotropy condition*. The expansion will not be reproduced here as there are many examples of such an approach to be found in literature, see for example the derivation in [29]. The *local mean energy approximation* assumes that the plasma can be described in terms of different electron parameters as a function of the mean electron energy by solving the two term expansion of the space independent BTE. The small anisotropy condition is the requirement that the mean free path of the particles are small compared to the size of the vessel, that the energy gained between collisions is small and that the rate at which the plasma parameters change is slow compared to the collision frequency. In addition to the afore mentioned, the two term expansion usually only takes into account collisions of the charged species with a singal neutral species. Coulomb collisions between different species are treated through the $\vec{F}/m$ term, not in the interaction term of the BTE. This condition is equivalent to the condition of the presence of a single dominant species in the derivation of the drift-diffusion equation from the conservation of momentum by a fluid approach. The result of this derivation is an expression for the mobility and the diffusion coefficient in terms of the isotropic component of the space dependent distribution function $F(\vec{x}, u)$.

$$D = \frac{1}{3} \frac{1}{g(\vec{x})N} \sqrt{\frac{2e}{m}} \int_0^\infty \frac{u}{\sigma(u)} F(\vec{x}, u) du \qquad (6.6)$$

$$\mu = -\frac{1}{3}\frac{1}{g(\vec{x})N}\sqrt{\frac{2e}{m}}\int_0^\infty \frac{u}{\sigma(u)}\frac{\partial F(\vec{x},u)}{\partial u}du \tag{6.7}$$

$$g(\vec{x}) = \int_0^\infty F(\vec{x},u)\sqrt{(u)}du. \tag{6.8}$$

$$\tag{6.9}$$

The equations 6.6 and 6.7 form the basis for methods [9, 10] which use the drift-diffusion equation in conjunction with tables of the mobility and the diffusion coefficients as a function of the reduced electric field strength (E/n). In this approach a steady state solution for the EDF is sought for a uniform electric field in a given density of the neutral particles. Using equations (6.6) (and 6.7) the corresponding mobility and diffusion coefficients are found. See also the discussion in [30], section 6.1.

However, in the case of HCD's the mean free path of the electrons is of the same order of magnitude as the size of the vessel and the pendulum effect is essentially non-local. The small anisotropy condition is not met, nor is the local mean energy approximation valid. The problems this poses for the fluid part of a hybrid model are discussed later.

An attempt has been made to produce mobility and diffusion coefficients from a Monte Carlo drift experiment. This is discussed in chapter 9. To incorporate the results two simple plugins have been added to PLASIMO . One to accept a lookup table for the mobility as a function of the reduced electric field strength, and another using the Einstein relation to calculate the diffusion coefficients from the mobility.

After the initial phases the degree of ionization increases and the assumption of the presence of a single dominant species or background gas is no longer valid. The plasma will then have to be described by a different diffusion model.

6.3.2 Anisotropic Diffusion

Until recently the models used in PLASIMO assumed isotropic diffusion. This is a valid assumption for unmagnetized plasmas with low Townsend numbers. The degree of magnetization is determined by the Hall parameter, the ratio between the Hall component of the resistivity tensor and the resistivity in the absence of the magnetic field. A plasma is magnetized if the Hall parameter for the electrons is greater than one, in which case the electrons gyrate around the magnetic field lines. The result is that the diffusion of electrons becomes anisotropic. At even greater magnetic field strengths ion diffusion can also become anisotropic [23], p 55. However, apart from the pinch fase of the discharge the magnetic fields do not play a significant role in the HCD[5].

Anisotropic diffusion is not exclusively the result of high magnetic field strengths. Electric fields can also lead to anisotropic diffusion. More specifically, we have to consider the ratio of the electric field strength to the density of the neutral particles. This ratio, E/n, is known as the Townsend number of the discharge. The common unit for this ratio is the Townsend, abbreviated with Td. 1 Td is equal to $10^{-21}\mathrm{Vm}^2$.

Discharges with high Townsend numbers show anisotropic diffusion. For HCD's in the startup fase electric fields are of the order of $10^5\mathrm{V/m}$ and the xenon pressure is low (approx. 20 Pa). These high electric fields, together with the low density, (some $7 \times 10^{20}\mathrm{m}^{-3}$ assuming a temperature of 2000 K) give rise to large drift speeds. Depending on the isotropy of the scattering from elastic and other collisions the EDF may become anisotropic. That is, the root mean square (rms) velocity in the direction of the electric field will differ significantly from the rms velocity in other directions if anisotropic scattering occurs. The study of the transport properties of xenon [31] show that the diffusion of electrons in xenon is anisotropic at reduced electric fields strengths above 1 Td. With a pressure of 20 Pa and a temperature of 2000 K electric fields of the order of $10^5\mathrm{V/m}$ yield 10^5 Td.

Anisotropic temperature

The use of a temperature T_p in (6.5) assumes an EDF in which the anisotropy in the velocity space is small. One possible remedy that allows one to still use a generalized form of (6.5) for larger anisotropies is to introduce an anisotropic temperature. Examples of this approach are found in the work of Ellis, Vliehland, Mason and others [32] who use a semi-empirical approach to calculate the transport properties of ions in gasses. This work introduces a anisotropic temperature with a semi-empirical relation between the temperature in the direction along the field lines and the temperature perpendicular to the field lines to arrive at an approximation for the anisotropic diffusion coefficients through a generalized form of the Einstein relation. This approach is impractical in its reliance on large numbers of experimentally defined quantities. A very rough approximation is obtained by simply using

$$kT_{\parallel} = kT + \frac{1}{4}Mv_d^2, \tag{6.10}$$

[5] Even in the pinch fase a two dimensional approach can ignore anisotropic effects as the magnetic field is mostly tangential.

with $T_{\|}$ the temperature parallel to the electric field lines, v_d the drift speed of the species in question with mass M. The temperature perpendicular to the field lines is assumed unaffected by the electric field. This "temperature" is then substituted into the Einstein relation. From (6.10) it also becomes clear when one can expect large deviations from anisotropic diffusion due to the electric field. Deviations are to be expected when the term $\frac{1}{4}Mv_d^2$ is no longer small compared to kT. The drift experiments in chapter 9 and in particular the figure 9.4 show the relative importance of the latter term for a case studied.

6.4 Secondary Electron Emission

Secondary emission of electrons from ion impact on the surfaces of the vessel forms an important source of fast electrons, and therefore an important source term for the kinetic model. To model this process the flux of ions toward the surfaces is calculated from the temperature and density of the ions in the fluid model. This flux is multiplied with a constant factor representing the chance that electrons are released to calculate the total electron flux into the vessel (the secondary electron emission coefficient). Super-particles representing this electron flux are then injected into the vessel from the boundary with random velocities sampled from the tail of the Maxwell distribution function matching the local electron temperature at the wall.

In PLASIMO the ion flux Γ toward the wall is usually calculated with:

$$\Gamma = \frac{1}{4} n v_B, \tag{6.11}$$

where n is the ion density and v_B is the Bohm velocity $\left(\sqrt{\frac{8kT_e}{\pi m}}\right)$ [26]. The factor $\frac{1}{4}$ comes from the assumption of an isotropic velocity. As has been noted in [20], p. 83 the velocity distribution of ions near the wall in an environment where collisions are rare is not isotropic. The speed of ions coming back from the wall is smaller than those going towards the wall, if indeed any come back at all. This yields an extra factor of two in (6.11) yielding a flux:

$$\Gamma = \frac{1}{2} n v_B.$$

The primary difficulty, however lies in the estimation of the secondary electron emission coefficient. This coefficient depends on many factors, such as the work function of the material of which the surface is composed, the surface roughness, impurities on the surface and the local electric field. Many of these factors are difficult to determine and may vary as the surface properties change with use. In order to test the working of the model for secondary electron emission a factor 0.3 is used.

6.5 Ionization Rates

The primary (though not the only) purpose of the kinetic sub-model is to calculate the ionization rates for use in the fluid sub-model. There exist two approaches to this end. The most straightforward method is to simply count the number of ionization collisions in the kinetic model and keep track of where they occur. The ionization rate is then simply the number of ionization collisions per volume cell divided by that cell's volume and the time step over which the data is collected. Another method is to determine the energy distribution and then to calculate the convolution of the cross section with the energy distribution function. The author has opted to implement the straightforward method of counting the ionization events per grid cell for reasons of simplicity. This approach has some drawbacks: the primary problem is the requirement that a large number of ionization collisions [6] per grid cell and per timestep are necessary to limit statistical fluctuations. This can be remedied by spreading the ionization collisions out over multiple grid cells or time steps. The most transparent and flexible approach would be to implement filters. Such filters have not yet been implemented, but some suggestions are presented in the conclusions at the end of this report.

[6]Note that the other method requires a large number of super-particles per grid cell to determine the local EDF

Chapter 7

Solving the Poisson Equation

As stated in the previous chapter, in section 6.1 a method of solving the Poisson equation is needed to yield the potential, and thereby the electric field, given the space charge distribution. This electric field is required as input to the drift diffusion equation, the energy balance (Ohmic dissipation) and the particle kinetic model. This chapter discusses the implementation and testing of the Poisson solver.

With the Coulomb gauge the potential Φ on a domain $\vec{x} \in \mathbb{R}$ in free space is given by the Poisson equation :

$$\Delta\Phi = -\frac{\rho(\vec{x})}{\epsilon_0}, \tag{7.1}$$

with ϵ_0 the permittivity of free space, $\rho(\vec{x})$ the charge density and Δ the Laplacian. The electric field follows from

$$\vec{E} = -\nabla\Phi - \frac{\partial\vec{A}}{\partial t},$$

where $\vec{A}$ is the magnetic vector potential. The approximation $\vec{E} = -\nabla\Phi$ has been used to calculate the electric fields. This approximation is valid if the speed of light in the plasma is much greater than the velocity with which the electric field changes[1]. Other EM modules in PLASIMO are capable of calculating the magnetic field from the vector potential. These are used to model radio frequency driven plasmas. For more information see the work of Van Dijk [34].

[1]The potential can change instantaneously in the Coulomb gauge, changes in the electric field cannot propagate faster than the speed of light. If propagation of changes in the potential become of the same order or faster than the speed of light the term $\frac{\partial\vec{A}}{\partial t}$ is needed.

In order to reuse as much code as possible the Poisson equation is rewritten in the form used for conserved quantities, the so-called Φ-equation (not to be confused with 7.1 for the potential). In steady-state form the Φ equation is given by:

$$\underbrace{\nabla \cdot (\rho_\phi \vec{v}\phi)}_{\text{convection}} = \underbrace{\nabla \cdot (\Gamma \, \nabla\phi)}_{\text{diffusion}} + \underbrace{S_\phi}_{\text{source}} \quad .$$

PLASIMO already has methods to solve equations of this form with a finite volume method. More information on the discretization scheme is given in [24], section 7.2. Using existing code that has already been optimized and debugged is far preferable to writing new code as it not only saves a lot of work but also avoids increasing overall complexity and introducing new errors.

The Poisson equation can be written as:

$$0 \; = \; \nabla \cdot (\epsilon_0 \nabla\Phi) \; + \; \rho(\vec{x}) \; .$$

The charge density $\rho(\vec{x})$ acts as a source term and the permittivity as a diffusion coefficient[2]. There is no convection term.

7.1 Testing the Poisson Solver

Even though code reuse limits the need for debugging, some tests are still needed to determine the accuracy and convergence of the approach being used. To test the method used for solving the Poisson equation a fixed charge density is imposed and the resulting potential is calculated (and from it the corresponding electric field). This numerically calculated electric field is then compared to the analytical solution for the electric field derived from the use of Gauß' theorem for different grid densities. From this one gains information on the convergence to the analytical solution for increasing grid density.

A number of charge distributions on an infinitely long cylinder have been tested. The potential at R_{max} is fixed at zero. The homogeneous Neumann condition ($\vec{n} \cdot \nabla\Phi = 0$) is used at the other boundaries see also figure 7.1. The electric field is calculated as the gradient of the potential through a three point finite difference approximation in the center, and a two point

[2]Note that if one replaces ϵ_0 with ϵ_r (the relative permittivity) the ϕ form of the equation would be correct for a dielectric. This is not used at present, but may turn out to be a useful feature in the future.

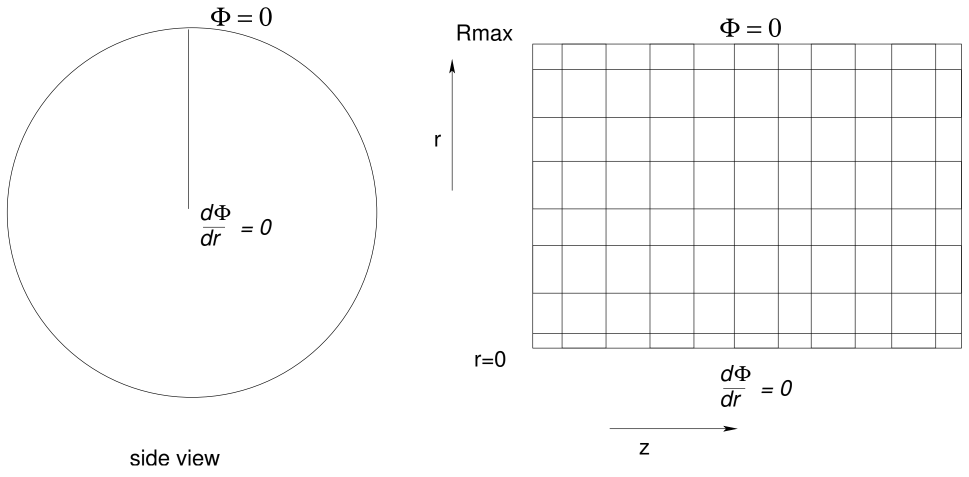


Figure 7.1: The grid on which the Poisson equation is solved.

approximation at the edges. In the analytical approach Gauß' theorem is used to arrive at the electric field directly.

Using Gauß' theorem it is easily shown that the electric field in the radial direction E_r is given by:

$$E_r \;=\; \frac{1}{\epsilon_0\, r}\int_0^r r\rho(r)\, dr \; . \tag{7.2}$$

Two test cases with different charge distributions have been constructed to compare with the results from PLASIMO .

1. *Constant charge density;* this case is the simplest. A constant charge density ρ is imposed on a volume with a radius R_{max}. Outside this volume the charge density is zero. The electric field in the radial direction is given by
$$E_r \;=\; \frac{\rho}{2\epsilon_0} r. \tag{7.3}$$

2. *"Sheath" type charge density;* in this case the space charge is concentrated along the outer ring of the cylinder. The charge is given by:

$$\rho(r) \;=\; \begin{cases} \rho_{max}\,(1-x)\left(x-\frac{\sqrt{2}}{2}\right)^2 & \frac{\sqrt{2}}{2} \le x \le 1 \\ 0 & \text{elsewhere.} \end{cases} \qquad \left(x = \frac{r}{R_{max}}\right) \tag{7.4}$$

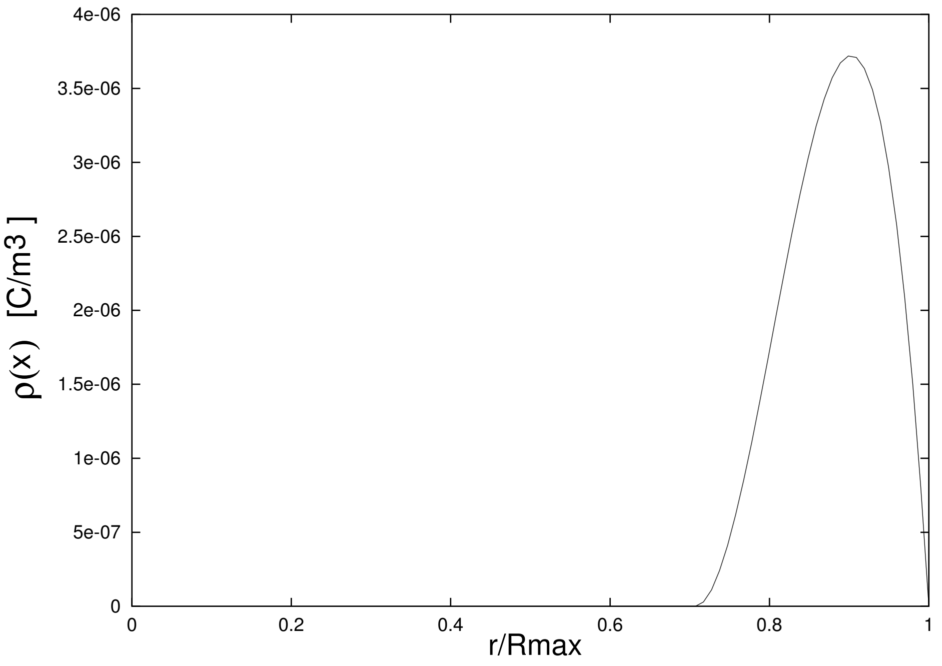


Figure 7.2: $\rho(r)$ with $\rho_{max} = 0.001$ Cm^{-3}

A graph of this space charge distribution is given in figure 7.2. Substitution of (7.4) into (7.2) and performing the integration yields

$$E_r(r) \ = \ \begin{cases} \frac{\rho_{max} R_{max}}{\epsilon_0} f(x) & \frac{\sqrt{2}}{2} \leq x \leq 1 \\ 0 & x \leq \frac{\sqrt{2}}{2} \end{cases} \text{, with} \tag{7.5}$$

$$f(x) \ = \ \left[-\frac{1}{5}x^4 + \frac{1}{4}\left(1+\sqrt{2}\right)x^3 - \frac{1}{3}\left(\frac{1}{2}+\sqrt{2}\right)x^2 + \frac{1}{4}x + \left(\frac{\sqrt{2}}{240} - \frac{1}{48}\right)\frac{1}{x}\right].$$

7.1.1 Test Results

A proper implementation of the Poisson solver requires that the difference between the numerical and the analytical solution convergences to zero as the gridspacing goes to zero. To examine the convergence behavior increasingly dense grids are tested. We compare solutions with the known analytical result for three equidistant grids with $R_{max} = 0.01$m doubling the number of gridpoints in each direction each time. The z coordinate range is $z \in [-0.01\text{m}, 0.01\text{m}]$ and ρ_{max} is equal to 0.001 Cm^{-3}. The grids had 20×20, 40×40 and 80×80 grid points respectively.

1. *Constant charge density;* for the case of constant charge density the results from Plasimo match very well on the interior points, but less so near the boundaries. The error is nearly 2% at the boundary for the sparsest grid (20×20). This is because the gradient is determined by a first order method near the boundary (by a two point method) whereas it is determined by a second order (three point) method for interior points. The discretization error is most evident for the grid with 20×20 points, as shown in figure 7.4. To test for convergence the differences between the results and equation (7.3) have been compared on different grids. The result is plotted in figure 7.4. From the graph of the residues it becomes clear that the convergence near the boundaries is linear; decreasing the gridspacing by a factor two reduces the error by the same factor. The error in the interior boundaries does not show clear convergence for decreasing grid spacing as the error remains constant. The error itself, though, is low (smaller than 10^{-6}). The lack of further convergence is most likely due to cancellation effects.

2. *"Sheath" type charge density*

   The case with the "sheath" type charge density requires more grid points for an accurate result. With 20×20 points the residue is more than 6% for $r = R_{max}$. In the region where the charge density has its peak the discretization error is also clearly present (see figures 7.5 and 7.6). The convergence seems to be better than in the case with a constant charge density. The absolute value of the error is larger, though. In figure 7.6, only the error between 0.0075 and 0.001 m has been plotted. Outside this region the analytical field is zero and the relative error is of the order of 10^{-16}.

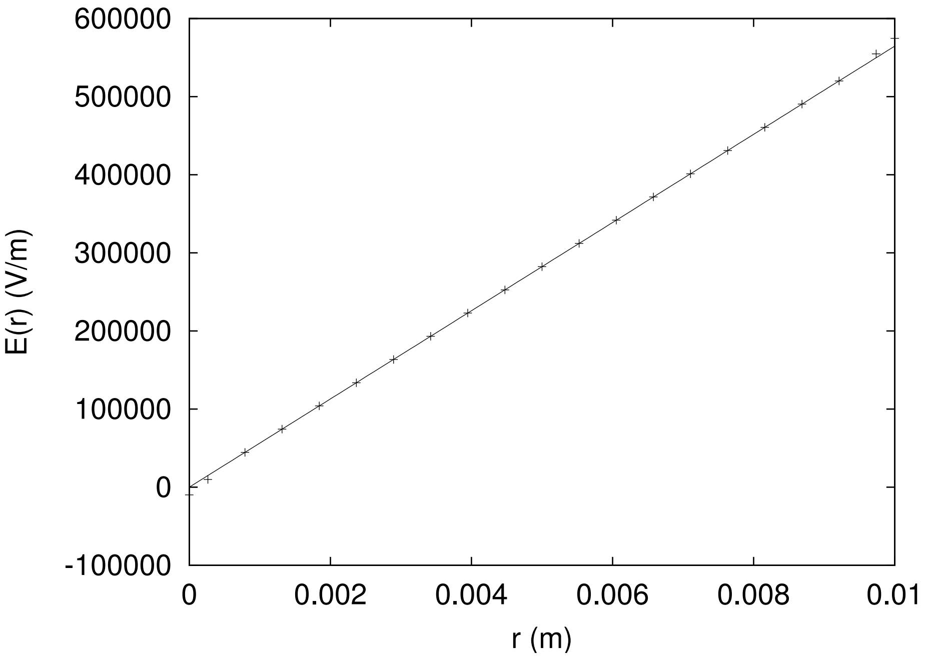


Figure 7.3: The electric field as calculated on a 20 by 20 grid ('+') and equation (7.3) (the straight line).

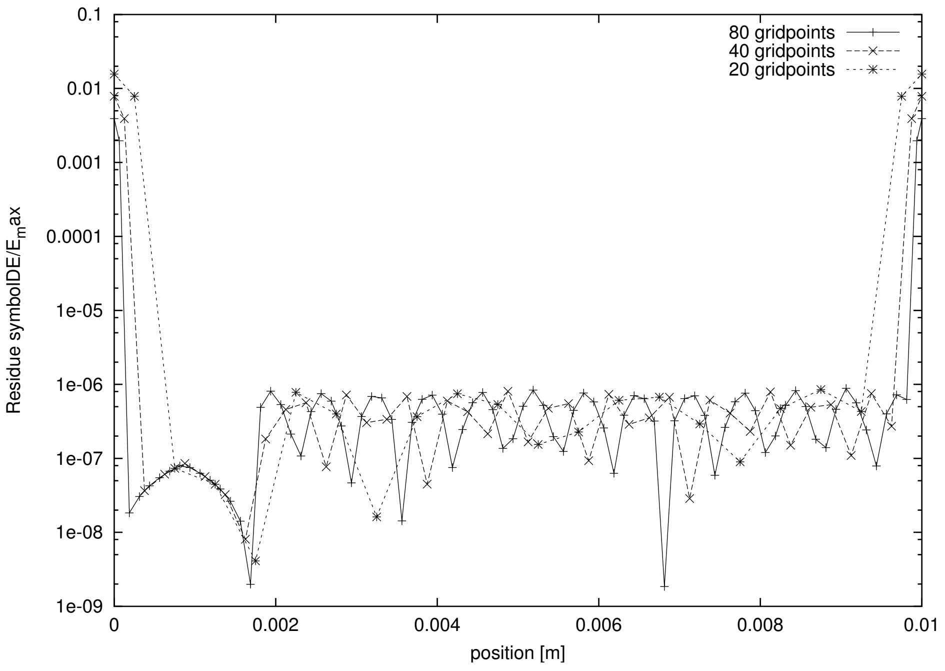


Figure 7.4: 'The residue for 20 × 20 (*), 40 × 40 (×) and 80 × 80 (+) grid points. The residue has been divided by the maximum field of 6×10^5 V/m to give a dimensionless result.

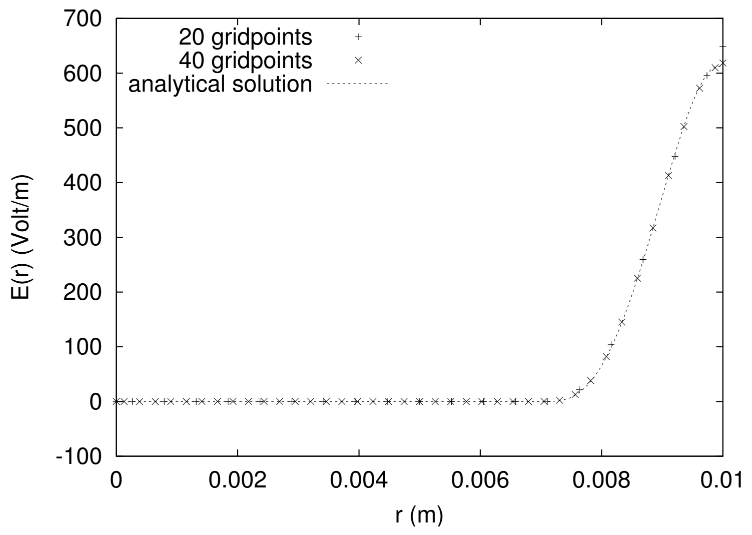


Figure 7.5: The electric field as calculated on a 20 by 20 grid ('+'), a 40 by 40 grid ('×') and equation (7.5).

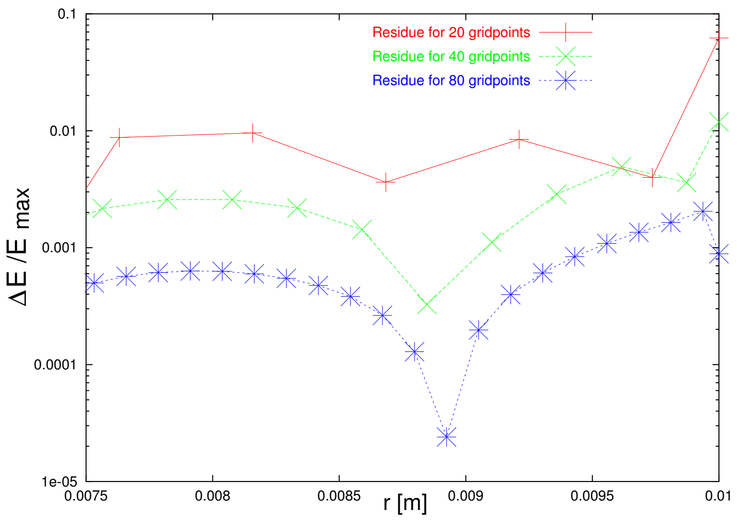


Figure 7.6: 'The residue for 20 × 20 (+), 40 × 40 (×) and 80 × 80 (*) grid points. The residue has been divided by the maximum field of 600 V/m to give a dimensionless result.

Chapter 8

The Particle Kinetic Sub-Model

As stated previously, PLASIMO needs to be expanded with a particle kinetic model. Rather than implementing a new Monte Carlo collisional model from scratch, the choice has been made to use a Monte Carlo code being developed in the Group Elementary Processes in Gas Discharges at the Eindhoven University of Technology for use with MD2D [33]. Using this code does, however, require that one implement methods to make data from PLASIMO available in a form that the Monte Carlo code understands (and vice versa). This is done by a construct known as a bridge class. Such a bridge class has been built. In principle all data from the fluid model is available to the Monte Carlo code. In practice the data actually used is limited to the electric field, the density of different species, the "temperature" of those species and the ion flux to the walls. In the other direction reactions rates such as the ionization rate are available for use in the fluid model. The resulting flowchart is shown in figure 8.1. Implementation details on the bridge class and related data structures are discussed in appendix A.

To test the working of the Monte Carlo code in isolation a simple test problem is required. Two test cases have been studied. One test case is a numerical drift experiment that compares the drift of electrons in a background of xenon with values from literature. This test case is discussed in the next chapter. The other test case simulates the diffusion of hydrogen in a background gas of xenon and is dealt with in the next section.

8.1 Diffusion

To test the handling of the collisions by the Monte Carlo code a numerical experiment has been devised consisting of a cylindrical vessel filled with xenon. In this vessel atomic hydrogen is released, instantaneously filling the entire vessel with a low density of hydrogen. The atomic hydrogen diffuses to the walls where it is absorbed. The resulting density as a function of time is compared with results from a "Knudsen" model for the diffusion. For the collisional cross section between the hydrogen and the xenon atoms a hard sphere model is used. The results from the kinetic Monte Carlo model are compared with a "Knudsen" diffusion model developed for PLASIMO by Broks in [20], chapter 7. This diffusion model uses a resistor model to account both for Knudsen like flows and standard Fick diffusion. The model was developed for regimes in which the mean free path is of the same order of magnitude as the size of the vessel so that the influence of the vessel size require only a minor correction to standard Fick diffusion. If the mean free path between collisions is much larger than the size of the vessel such a modified fluid approach is no longer possible.

The effective diffusion coefficient D_f is then given by:

$$D_f = \frac{D_s D_k}{D_s + D_k},$$

with D_s the standard Fick diffusion coefficient and D_k the "Knudsen" diffusion coefficient. The latter is given by

$$D_k = \frac{4}{3} K_0 v_{th},$$

with v_{th} the thermal velocity and K_0 the so called Knudsen factor. The Knudsen factor is, in turn, approximated by

$$K_0 = f_{Kn} \Lambda,$$

with f_{Kn} the Knudsen geometry factor and Λ the gradient length. See [20] for more information. The initial conditions used for both the kinetic model and the Knudsen model are shown in table 8.1. The results are presented in figure 8.2.

8.2 Results

As can be seen in figure 8.2 the decay of the hydrogen density in the Monte Carlo method diverges from that in the modified Knudsen corrected fluid

Gas Temperature	1000 K
Xenon Density	7.24 $\times 10^{20}$ m^{-3}
Atomic Hydrogen Density	1 $\times 10^{18}$ m^{-3}
Collisional cross section H-Xe elastic collision	5.56 $\times 10^{-20}$ m^{2}
Knudsen geometry factor	0.1875
Knudsen gradient length	0.008 m
Number of super-particles	80000
Vessel Dimensions:	
radius	0.016 m
height	0.016 m

Table 8.1: *Initial conditions and inputs for the Hydrogen Diffusion experiment, for both the kinetic and the Knudsen fluid approach.*

model. The modified fluid model assumes that the temperature of all the heavy particles is the same and that the energy distribution function of the hydrogen atoms remains Maxwellian. This is, however, not the case, as can be seen in figure 8.3. The fast particles reach the walls first, effectively cooling the hydrogen atoms down. The elastic collisions with the xenon atoms are insufficiently frequent to keep the temperature at the original 1000 Kelvin. In addition, the experiment starts showing statistical fluctuations due the depletion of super-particles from the experiment. The initial slope of the number of particles does match. This indicates that the "Knudsen" diffusion model can be used in a quasi steady state approach together with an additional energy balance for the hydrogen atoms. Such an approach would update the diffusion coefficients every time step (with each time step smaller than 10 microseconds).

The experiment has also been repeated with a xenon partial pressure of 200 Pascal, or a density of 1.45$\times 10^{22}$m^{-3} to decrease the initial mean free path for collisions between hydrogen and xenon from 0.025m to 0.00124m. Bringing the mean free path down to an order of magnitude smaller than the size of the vessel takes the problem to the regime for which the Knudsen part of resistor model amounts to a small correction of the Fick model. As is clear from 8.2 the Monte Carlo results are a much better match to those of the fluid model in this regime.

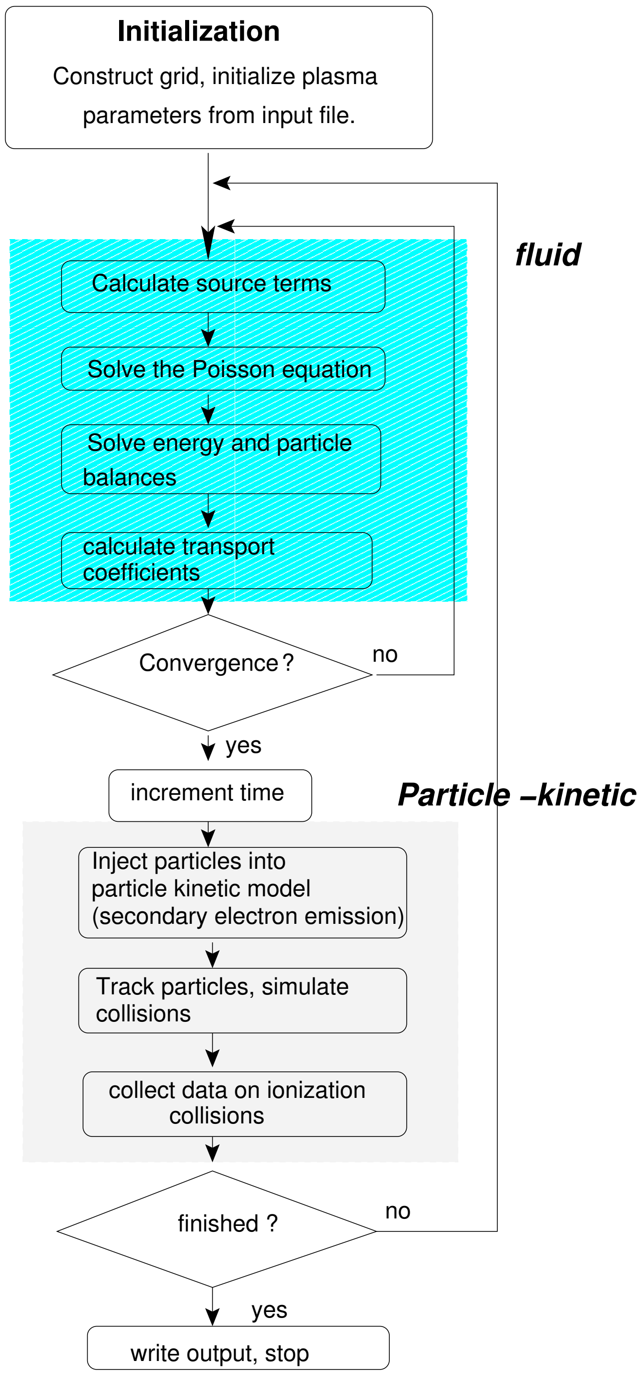

Initialization
Construct grid, initialize plasma parameters from input file.
fluid
Calculate source terms
Solve the Poisson equation
Solve energy and particle balances
calculate transport coefficients
Convergence ?
no
yes
increment time
Particle –kinetic
Inject particles into particle kinetic model (secondary electron emission)
Track particles, simulate collisions
collect data on ionization collisions
finished ?
no
yes
write output, stop

Figure 8.1: Flowchart of the hybrid model.

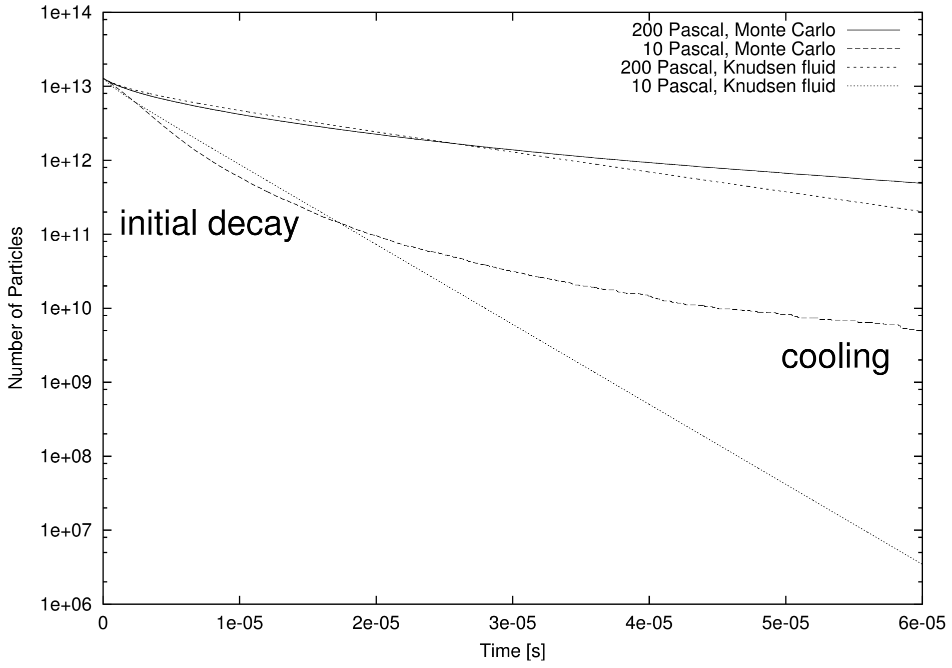


Figure 8.2: Hydrogen diffusion in a cylindrical vessel: Plot of the number of hydrogen atoms remaining in the vessel as a function of time according to the two methods used.

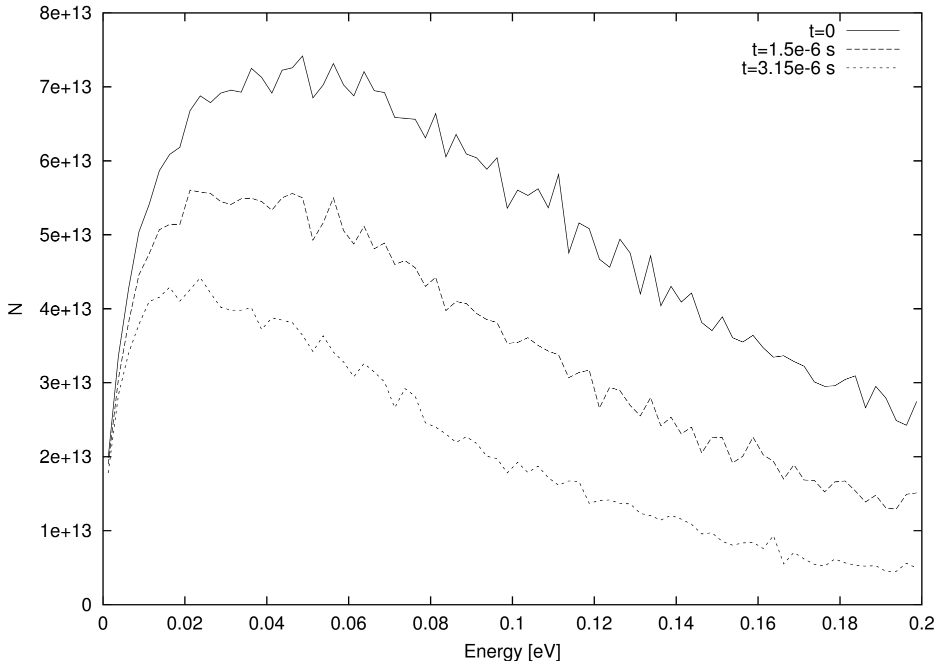


Figure 8.3: The energy distribution function of the hydrogen atoms. Note that the EDF has not been normalized.

Chapter 9

Transport Properties of Xenon at High Townsend Numbers

Fluid models require transport properties for the species in the model as input. In particular any fluid model which uses the drift diffusion equation requires as input either the mobility and diffusion coefficients, or a method of calculating them. This can be by calculating the convolution of the cross sections, whose values can be obtained from literature with the EDF, or by using simplifications such as the Langevin limit to arrive at an analytical expression for the collision integrals, and thereby the diffusion coefficients. In PLASIMO several different approaches are possible to arrive at the mobility and diffusion coefficients. More on these the different approaches is found in the work of Johnston [35] in chapters 7, 8 and 9. To the pre-existing possibilities a method using a simple lookup table has been added. This method requires a file containing the mobility for different ratios of E/n. The diffusion coefficient is then calculated using a (generalized)Einstein relation.

In deriving the drift diffusion equation assumptions are made that are questionable under the conditions present in a pulsed HCD. In particular we wish to examine what happens at high values for the reduced electric field strength (E/n). Simply assuming a modified version of a Maxwellian EDF and then integrating the cross sections for various collisions over this assumed EDF is unlikely to yield accurate transport properties. In effect, this is another example of a modified fluid approach where a deviation from a Maxwellian EDF is corrected for. To get a more accurate approach and to test the assumptions explained earlier in 13 we conduct a numerical experiment.

9.1 Mobility and Diffusion Coefficients from a Numerical Drift Experiment

Transport properties for a low temperature plasma can be calculated by solving the Boltzmann equation and integrating the resulting EDF over the appropriate cross sections for the dominant collisions. Cross sections for elastic and inelastic collisions as well as ionization reactions are available from literature, though values may be inaccurate near the threshold energy. We test this approach with the aim of providing a lookup table for the mobility coefficients. The BTE is calculated with a Monte Carlo method, using a program developed for the Micro-dis project [33], cross sections from McEachran and Stauffer (as published by [36]) for the elastic collisions, cross sections for xenon excitation by electron impact from [38], and ionization cross sections from [39]. This numerical experiment uses mostly the same code as the particle kinetic part of the hybrid model but with a given uniform and constant electric field and background gas. Using the code also used for the particle kinetic model also provides a further test case for the code.

The program solves the Boltzmann equation starting with a given initial distribution function of the electrons and integrates their path given the electric field. The program waits a given time for the EDF to obtain a steady solution until it starts collecting data. Using the data from the program the drift velocity is obtained. This has been used to calculate the effective mobility coefficient ($\mu = E/v_d$). An example of this approach is shown in 9.1.

Some work has also been done on obtaining diffusion coefficients but at the time the calculations where carried out it was not yet possible to obtain reliable diffusion coefficients. Diffusion coefficients can be obtained with the MAGBOLTZ program by S. Biagi. Since the MAGBOLTZ program also returns the mobility, it can also be used to verify the Micro-dis code.

The MAGBOLTZ [37] program, originally written by S. Biagi from CERN, and now maintained by R. Veenhof, does a Monte Carlo integration of the Boltzmann transport equation. The program output yields anisotropic diffusion and mobility coefficients as well as Townsend coefficients and the EDF. By running this program over a wide range of values for the electric field at a pressure of 0.5 Torr Xe and a temperature of 300K the results found in figures 9.2 and 9.3 are produced. The former also shows, for comparison, value for the mobility of electrons in xenon as found in literature. Note the

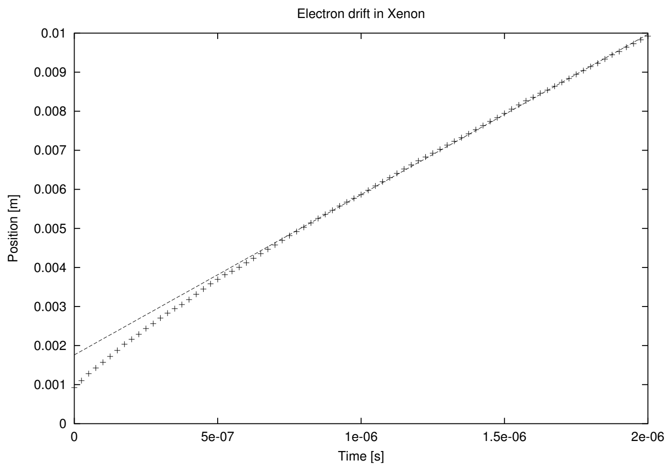


Figure 9.1: An example of how the drift speed, and thereby the mobility is calculated. The graph shows the average position of the swarm in time. This particular example uses an electric field of 150 V/m.

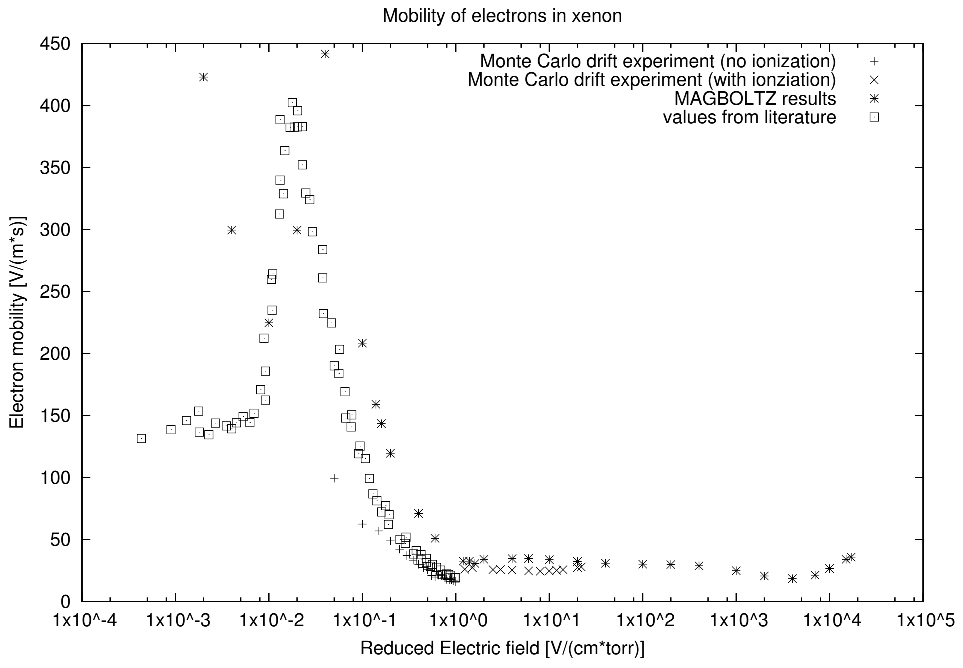


Figure 9.2:

peak in those values from literature values, due to the Ramsauer minimum. As the error in the Monte Carlo models is large at low electric field strengths the Ramsuaer minimum is less pronounced in the results from these models.

9.2 Verifying the Coefficients

The accuracy of the data predicted by a numerical experiment is best checked by comparing the results with known experimental results. Unfortunately, no experimental results are available in the regime under which the hollow cathode operates. The spontaneous breakdown of the gas would make it near impossible to get accurate experimental data.

Experimental results are available for lower field values. The results from the Monte Carlo drift experiment have been compared with values from [40]. The results for the mobility can also be compared with those from the numerical drift experiment done with the Micro-dis Monte Carlo code[33]. The results from the calculations are presented in figure 9.2. There are large discrepancies at low values for the electric field. This is because of statistical fluctuations in the drift velocity. In particular, the Ramsuaer minimum causes a large peak in the mobility coefficient. This is clear in the values from literature, but less so in the drift experiment. The relative error in these fluctuations is large when the field is small. The values at low electric fields are not important for the hollow cathode though. For higher values of the reduced electric field the result converges with the experimental values. The MAGBOLTZ calculations are also consistent with those from the Monte Carlo drift experiment that uses cross sections from a different source and a slightly different method.

Together with the "temperature" (using $kT = \frac{2}{3}\bar{\epsilon}$, with $\bar{\epsilon}$ the average energy) from the EDF one can check the validity of the Einstein relation as well as the generalized Einstein relation using anisotropic temperatures. As becomes clear by examining the results in figure 9.4 Mv_d^2 is always much smaller than the average swarm energy. The same figure also shows the central problem with the approach of calculating the transport properties of electrons by looking for a steady state solution for the drift velocity, while taking only the first ionization step into account. While higher electric fields would cause further ionization in practice, the model knows no such ionization steps, and instead the average energy of the electrons increases. Furthermore, the steady state solution sought in the drift experiment does not resemble the conditions in the hollow cathode during the initial phases of the discharge.

The result for the diffusion coefficients are presented in figure 9.3. As can be clearly seen in the graph the calculated diffusion coefficients are anisotropic, with the largest differences between 1 and 100 V/m. The difference between the generalized Einstein relation and the usual Einstein relation is small for these fields though, as the correction is small. For electric fields between 200 and 1×10^6 V/m the anisotropy is smaller. For these values the generalized form of the Einstein relation does not help to increase accuracy, though. The proposed anisotropic diffusion model had best be discarded when modeling hollow cathode discharges.

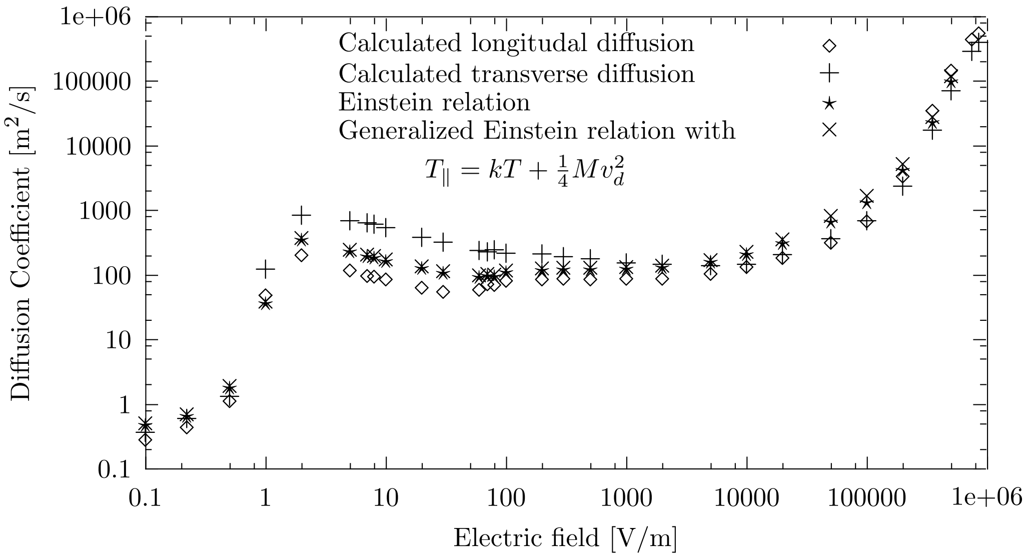


Figure 9.3: Diffusion according to the Einstein relation compared to results from MAGBOLTZ.

9.3 Conclusion

The results from the numerical experiments show that the assumptions required to allow the use of mobility and diffusion coefficients and the drift diffusion equation to study the breakdown of the plasma are not consistent with the conditions in the hollow cathode. To examine the transport of electrons in the early phases of the breakdown it is necessary to solve the momentum equation in the time dependent form. Let us reexamine equation

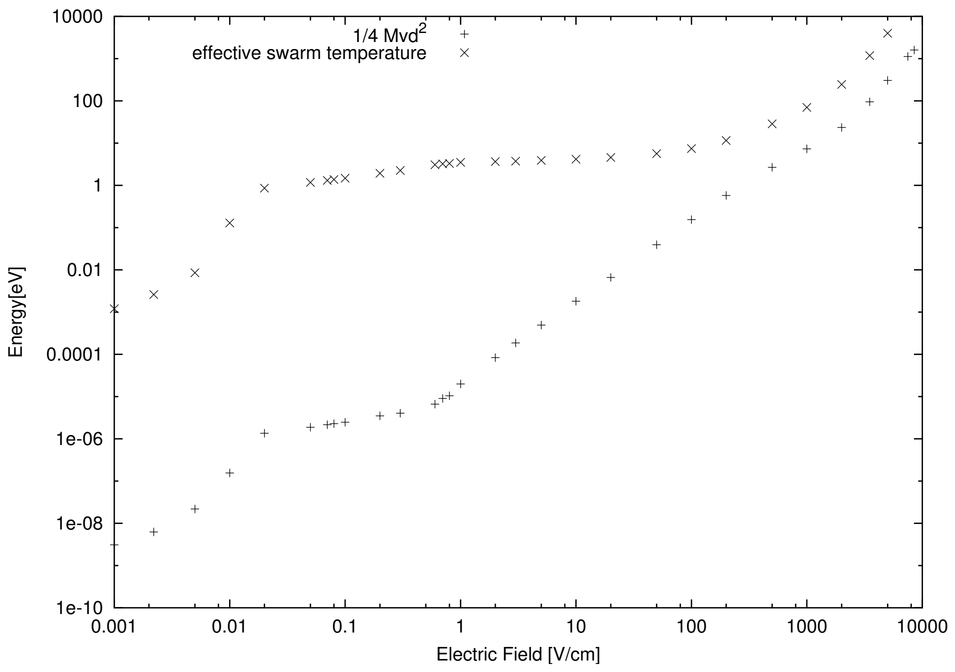


Figure 9.4: Temperature of the swarm compared to $\frac{1}{4}Mv_d^2$.

6.2:

$$\underbrace{\rho_p \frac{\partial \vec{u}_p}{\partial t}}_{1} + \underbrace{\rho_p \left(\vec{u}_p \cdot \nabla\right) \vec{u}_p}_{2} = \underbrace{-\nabla \cdot \mathbf{P}_p}_{3} + \underbrace{\vec{F}_p}_{4} + \underbrace{\sum_j \vec{F}_{pj}}_{5} .$$

Consider the condition were one applies an electric field of 10^5 V/m to a xenon gas with neutral xenon density of 10^{21}particles/m^3. The electron density is still low (10^{14} m^{-3}). The dimensions of the vessel are 1 cm.

term 1 For highly transient plasma the time dependency cannot be ignored. This is particularly the case for the ions, but to a lesser degree even for the electrons.

term 2 This term can be safely ignored for a gas this dilute.

term 3 This term accounts for Coulomb collisions between like species. Removing it would allow infinite gradients. One can approximate it with $kT_p \nabla n_p$. With a temperature of the order of 1 eV this results in an order of magnitude of 10^{-3} N/m^3.

term 4 This term is the driving force. At the electric field under consideration the force on a single charged particles is 1.6×10^{-14} Newton. This brings the order of magnitude of the term to 1 N/m^3.

term 5 In this term we need only consider collisions with neutral species. There are simply to few of the other species to collide with. Estimating the collisional rate for elastic electron xenon collisions to be of the order of 10^{-13} m^3/s, the order of magnitude of this term is estimated to be 10^{-3}kgm^{-4} times the speed. This term becomes effective as the average drift speed increases, working to put a limit on this increase.

Chapter 10

Conclusions

Hollow cathode discharges for the production of EUV light are highly transient and far from equilibrium. During a time scale of 100 ns several Joule of energy are pumped into a small device (approximately 10 cm^3) filled with xenon to a low pressure of 20 Pascal. This makes modeling of the discharge difficult, especially since the discharge goes through several phases. Approximations valid in one phase of the discharge will not be valid in another phase.

After examining existing methods of modeling pulsed hollow cathode discharges, the choice was made to use PLASIMO as part of a hybrid Monte Carlo-fluid model. A number of extensions have been added to facilitate the modeling of this type of highly transient discharge. These extensions have been verified using a number of test cases. A number of issues, however, remain to be solved before a working model can be constructed.

Many of the items named in section 6.1 have been implemented. To recuperate:

The diffusion model A diffusion model has been implemented that solves the transport equation for the electrons without assuming quasi-neutrality. Some work remains to be done on solving the momentum balance through an approach not involving the use of the drift-diffusion equation. This is discussed in the section on remaining issues.

The Poisson equation An EM module has been added that returns the electric field given the space charge distribution.

The particle kinetic model An interface has been built that allows the use of the Monte Carlo code from Micro-dis to calculate ionization rates in PLASIMO . Because of its general setup it can also be used for other reaction rates.

Secondary electron emission A module has been added that injects electrons into the particle kinetic model to simulate secondary electron emission from the surfaces of the hollow cathode. The module uses the ion flux to the surface from the fluid model to calculate how many particles it needs to inject. The module was constructed by relying on the code from PLASIMO 's module for boundary conditions.

Projection and interpolation Code has been implemented to interpolate quantities defined on a grid to positions required by the particle kinetic module. Vice versa, code has also been written to project positions in the particle kinetic module back onto the grid. The latter can be used to obtain ionization rates for use in the fluid model.

Though much work has been done on the required extensions some issues still remain before a working module can be constructed. The remaining issues are discussed section 10.2.

10.1 Test Cases

The Monte Carlo code from the Micro-dis project has been subjected to two separate test cases. One test case examined the behavior of a swarm of electrons in xenon accelerated by uniform electric field. The results from this test case indicate that the mobility and diffusion coefficients can be calculated to good accuracy with literature values in the range of 0.1 to 1 V/(cm torr)[1]. This is well below the operating regime of the pulsed hollow cathode discharge. Since no reliable data was found for the regime in which the hollow cathode is operated the results from the Micro-dis code were compared with those from MAGBOLTZ. These results also compare favorably.

The results, however, also show that an approach using a lookup table for the diffusion and mobility coefficients as a function of the reduced electric field is not well suited for the regime in which the hollow cathode for EUV production is operated. This is because the reduced field approximation seeks a static solution to the BTE while the plasma in the hollow cathode is highly transient. As a result the average kinetic energy of the electron swarm reached is, therefore, unrealistically high (greater than 100 eV, where

[1]1 V/(cm torr)=0.752 V/(m Pa)

the electron temperature for the bulk electrons is expected to be well below the first excitation energy of 8.3 eV).

The test case simulating the diffusion of hydrogen in a background of xenon also validates Broks' "Knudsen" correction model. This model is shown to work in the regime for which it is intended; a fluid in which the mean free path of the particles is of the same order of magnitude as the length of the vessel. Thus stage of recombination in the pulsed hollow cathode discharge can be reliably modeled with this approach.

10.2 Remaining Issues

In order to achieve a working model of the hollow cathode a number of remaining issues need to be addressed.

Geometry The most significant remaining issue in terms of the necessary effort on the part of the programmer is addressed in appendix B: the geometry of the problem. The geometry of the hollow cathode is an essential part of the hollow cathode effect. At the moment the code cannot handle more complicated geometries. An extension is needed to deal with more complex grids than the present rectangular coordinate spaces.

Transport properties A second problem remains in terms of the transport properties for the bulk of the (slow moving) electrons.

Assembly A third, and by no means trivial, remaining issue is the assembly of all the parts into a working model. This will involve further testing and debugging.

10.3 The Drift-Diffusion Equation

The use of the drift-diffusion equation to model pulsed hollow cathode discharges has been examined and found to be fraught with difficulties. Underlying assumptions, such as a small mean free path of the electrons compared to the size of the vessel as well as the local field approximation, are not generally valid in the regime in which hollow cathode discharges are operated. Modified fluid approaches, such as introducing anisotropic temperatures, or using lookup tables of the mobility coefficients as a function of the reduced electric field, do not suffice to solve these problems.

10.3.1 Another Approach for Particle Transport

At high electric fields, the current approach in using the drift-diffusion equation causes un-physically high drift velocities. One possible approach to solving this problem is to cut off bulk drift velocities above the thermal velocity belonging to the Maxwellian distribution function at the given electron temperature. Such an approach is consistent with the model in which the EDF of the electrons is assumed to deviate from Maxwellian in the tail only, with that tail fraction forming a small fraction insignificant to the transport of charge. A cut-off approach, however, ignores the underlying problems which cause the un-physically high drift velocities related to the long mean free path of the electrons.

A reexamination of the derivation of the drift-diffusion equation, however, shows that it is probably a much better idea to use a different approximation to the momentum balance. Instead of a quasi-static approach, one should follow an inertial approach. Such an approach would include the time dependent term, as well as the electric field and the elastic collisions with neutral atoms. The pressure gradient should be included to avoid un-physically high gradients.

Another possible approach would be to follow all electrons with the particle kinetic model. Care will have to be taken, however, with the numerical difficulties that may arise with this approach. Statistical fluctuations in the density of the electrons may be amplified by the Poisson solver. This problem can be solved with the use of adequate filters. To date, however, such filters are still under development for the Micro-dis Monte Carlo code.

10.3.2 Filters

The lack of filters to even out statistical fluctuations is also cause for concern in obtaining accurate ionization rates. The currently implemented approach projects the positions of the ionization events on the grid. At the end of each timestep the ionization events per grid cell are counted and the rate is determined by directly dividing the number of ionization events by the time step to arrive at the ionization rate for that gridcell. This approach causes large statistical fluctuations when the ionization rate is low. The use of a filter to smooth out the peaks in time and space could serve to limit such statistical fluctuations and improve the numerical stability.

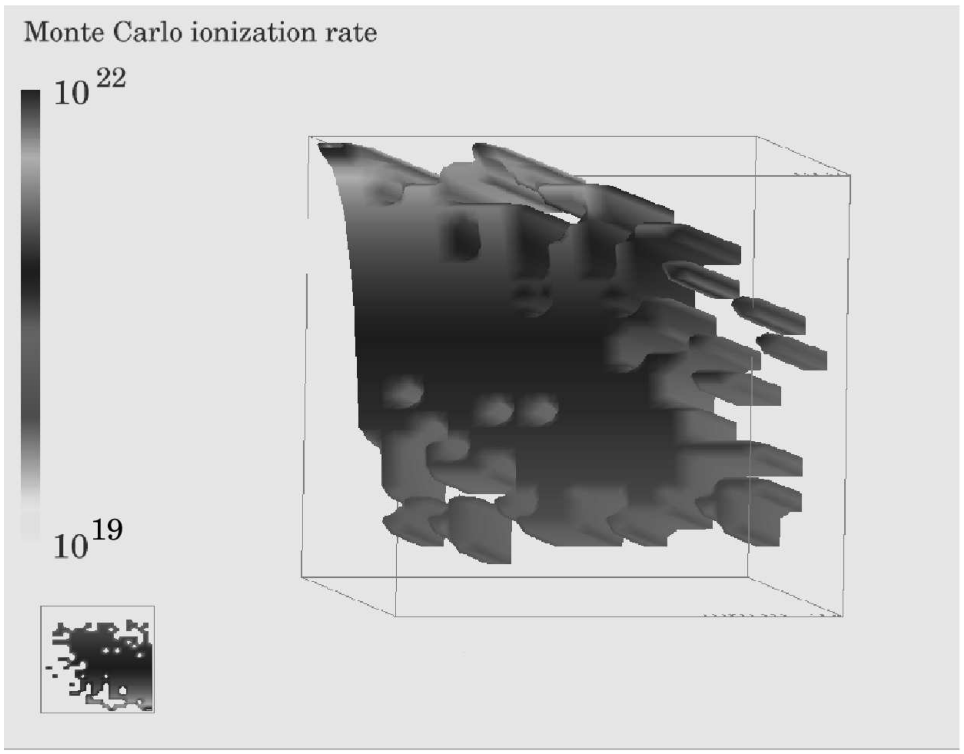


Figure 10.1: An example of why filtering is necessary. The figure shows the ionization rate (units $\mathrm{m}^{-3}\,\mathrm{s}^{-1}$) as determined by directly projecting the number of ionization events onto the grid during one time step.

10.3.3 The Energy Balance Equation

Another outstanding issue is the energy balance for the electrons. The focus of this work has been the formulation of the momentum transport, but one of the strengths of PLASIMO and an important reason for the choice of PLASIMO over the Micro-dis code is its handling of the energy balance equations. However, the simplified form of the energy transport equation used in the code relies on assumptions similar to those in the drift-diffusion equation. For example, a short mean free path of the electrons is assumed. A proper examination of the validity of the assumptions used is also needed.

10.4 A Final Word of Advice

Much of the remaining work requires good insight in software architecture and programming techniques. The author has spent a significant portion of the total time allocated for his graduation project learning C++ and gaining insight into PLASIMO 's design. Although the code has been written in such a way as to be modular in design and to allow extensions, such extensions are only easy to write if they can be implemented as simple plugins without requiring the rewriting of other parts of the code. This was not the case for the removal of the assumption of quasi-neutrality, and the interfacing of Monte Carlo collisional models with PLASIMO .

Further modification of the code currently used to represent the grids will require substantial reconstruction. It is not recommended that a graduate physics student attempt such an undertaking as part of the research towards a masters thesis as he or she may end up spending far more time on coding than on physics. As part of a PhD the time spent on obtaining the required level of programming skills and insight into PLASIMO 's design would be better spent, as time would remain to make good use of such skills.

List of Symbols

M	electron multiplication factor ( )	16
$\vec{j}_e$	electron current density ($\mathrm{A\,m^{-3}}$)	16
γ	secondary electron-emission coefficient ( )	16
$f(\vec{x}, \vec{v}, t)$	particle distribution function ($\mathrm{m^{-6}\,s^{3}}$)	26
N	total number of particles ( )	26
t	time (s)	26
$\vec{F}$	volumetic force per unit volume ($\mathrm{N\,m^{-3}}$)	26
∇_v	gradient in the velocity space ($\mathrm{m^{-1}\,s}$)	27
$\left(\frac{\partial f}{\partial t}\right)_{int}$	collision/interaction term of the BTE ($\mathrm{m^{-6}\,s^{2}}$)	26
∇_ψ	gradient in phase space ($\mathrm{m^{-1}}$)	27
$\vec{E}$	electric field (V/m)	27
$\vec{B}$	magnetic field (Tesla)	27
q	charge per particle (C)	27
ρ	charge density ($\mathrm{C\,m^{-3}}$)	27
$\vec{j}$	current density ($\mathrm{A\,m^{-3}}$)	27
n	particle density ($\mathrm{m^{-3}}$)	28
ρ_m	mass density ($\mathrm{kg\,m^{-3}}$)	29
T	temperature (K)	29
u	average velocity (m/s)	29
$\vec{q}$	heat flux ($\mathrm{J\,m^{2}/s}$)	29
$\mathbf{P}$	pressure tensor (Pascal)	29
n_e	electron density ($\mathrm{m^{-3}}$)	35
n_i	ion density ($\mathrm{m^{-3}}$)	35
Z	charge number ( )	35
p	pressure (Pa)	39
$\mathbf{\Pi}$	viscosity tensor (Pa)	39
u_{therm}	random thermal velocity (m/s)	39
Ω	rate coefficient for momentum transfer ($\mathrm{m^{3}/s}$)	39
f^{eff}	effective friction force per unit volume ($\mathrm{N\,m^{-3}}$)	39

μ	mobility (m^2 /(V s))	39
D	diffusion coefficient (m^2/s)	40
$F(\vec{x}, u)$	space dependent distribution function (from the two term expansion of the BTE) ($m^{-6} s^3$)	40
$T_{\parallel}$	temperature parallel to the electric field lines (K)	43
v_d	drift speed (m/s)	43
Γ	the ion flux toward the wall ($m^{-2} s$)	43
v_B	Bohm velocity (m/s)	43
ϵ_0	the permittivity of free space (Farad/m)	45
Δ	Laplacian (m^{-2})	45
$\vec{A}$	magnetic vector potential (V m/s)	45
D_f	effective diffusion coefficient (m^2/s)	54
D_s	Fick diffusion coefficient (m^2/s)	54
D_k	Knudsen corrected diffusion coefficient (m^2/s)	54
v_{th}	thermal velocity (m/s)	54
K_0	Knudsen factor (m)	54
f_{Kn}	Knudsen geometry factor ( )	54
Λ	gradient length (m)	54

A Note on the Use of Subscripts

In most of this work subscripts are used to differentiate between quantities per species. For this purpose the subscript p is used; except in those cases where an additional subscript is needed to differentiate different species, in which case a subscript j is used. For example, $\vec{F}_{pj}$ is used to represent the force per unit volume from the interaction between species p and species j. In the discussion on the Boltzmann transport equation they are largely omitted to avoid clutter. In a few cases subscripts are used for other purposes (such as indicating coordinates, indicated by the subscript i). These cases should be clear from the context. Note also that the mass density for a species p is notated as ρ_p not $\rho_{m,p}$.

Bibliography

[1] ASML press release, http://www.asml.com

[2] J.E. Bjorkholm, *EUV Lithography - The Successor to Optical Lithography?* available for download from http://www.intel.com/technology/itj/q31998/pdf/euv.pdf.

[3] Remko Stuik *Characterization of XUV Sources* Technische Universiteit Eindhoven, thesis. 2002

[4] E. R. Kieft, J. J. A. M. van der Mullen, G. M. W. Kroesen, and V. Banine, *Time-resolved pinhole camera imaging and extreme ultraviolet spectrometry on a hollow cathode discharge in xenon* Phys. Rev. E 68, 056403 (2003).

[5] F. Paschen, Wied. Ann. 37 (1889) 69

[6] Yu. P. Raizer Gas Discharge Physics, Springer 1991

[7] J. Christiansen and Ch. Schulteiss *Production of High Current Particle Beams by Low Pressure Spark Discharges* Zeitung für Physik A 290, 35 (1979

[8] D. Bloess et. al. *Triggered pseudo-spark chamber as a fast switch and as a high-intensity beam source* Nuclear Instruments and Methods 205 (1983) 173-184

[9] L.C Pitchford, N. Ouadoudi, J.P. Boeuf et. al. *Triggered breakdown in low-pressure hollow cathode (pseudospark) discharges.* J. Appl. Phys. 78 (1), 1 July 1995

[10] N. Baguer, A. Bogaerts, R. Gijbels *Hybrid Model for a cylindrical hollow cathode glow discharge and comparison with experiments.* Spectrochimica Acta B. 57 (2002)

[11] K. Bergmann, O. Rosier, W. Neff and R. Lebert *Pinch-plasma radiation source for extreme-ultraviolet lithography with a kilohertz repetition frequency.* Applied Optics, 29 (2000) 3833-3837

[12] Günterschulze Z. Physik 19, 313 (1923)

[13] K. Garloff, J. v.d. Mullen *Modeling pinced plasmas for EUV generation: Achievements and challenges.* Internal report for the PLASIMO group.

[14] V.I. Kolobov and L. Tsendin *Analytic model of the hollow cathode effect.* Plasma Sources Sci. Techn. 4 (1995) 551-560

[15] R.R. Arslenbekov, A.A. Kudryavtsev, and I.A. Movchan *Spatial and energy distributions of fast electrons in discharges with a cylindrical hollow cathode* Sov. Phys. Tech. Phys. 37 4 (1992) 395-398

[16] R.R. Arslenbekov, A.A. Kudryavtsev, and I.A. Movchan *Slow electron distribution function in a cylindrical hollow cathode* Sov. Phys. Tech. Phys. 37 6 (1992) 620-624

[17] Cai, S.Y.; Striffler, C.D.

[18] R. Werner, Private communications

[19] Hoyoung Pak and Mark J. Kushner *Breakdown characteristics in non-planar geometries and hollow cathode pseudospark switches* J. Appl. Phys. 71 1 (1992) 94-100

[20] B.H.P. Broks *Extending the capabilities of a plasma simulation model* Technische Universiteit Eindhoven, masters thesis

[21] David Chandler, David Wu, emphIntroduction to modern statistical mechanics, Oxford University Press, 1988

[22] Kerson Huang, Statistical Mechanics, John Wiley & Sons, Inc. 1963

[23] D.C. Schram, R. Engeln, Inleiding Plasmafysica, Eindhoven University of Technology lecture notes, 1997.

[24] D.A. Benoy *Modeling of thermal Argon Plasmas* Eindhoven University of Technology, Phd thesis, 1993

[25] J. van Dijk, *Modelling of Plasma Light Sources an object-oriented approach*, Thesis Eindhoven University of Technology 2001

[26] Jan van Dijk, Ger Janssen, Dany Benoy, Harm van der Heijden, Bart Hartgers, Colin Johnston, Kurt Garloff, Marnix Tas, Joost van der Mullen *Plasimo: A General-Purpose Toolkit for Plasma Modelling* Internal report (usually referred to as "plBook"). *One-dimensional simulation studies of breakdown and electron beam generation processes for a hollow cathode pseudospark discharge* IEEE Particle Accelerator Conference 17-20 May 1993, Proceedings Page(s): 3075 -3077 vol.4

[27] A.V. Vasenkov, B.D. Shizgal *Numerical study of a direct current plasma sheath based on kinetic theory*, Physics of Plasmas, vol 9 (2) 691-700 (2002)

[28] B. Hartgers *Modelling of a Fluorescent Lamp Plasma* PhD thesis, Eindhoven University of Technology, The Netherlands, 2003. available for download from http://plasimo.phys.tue.nl/publications.

[29] L.L. Alves, G. Gousset, S. Vallée, *Nonequilibrium Positive Column Revisited* Special Issue IEEE Transactions on Plasma Science, aug. 2003.

[30] G. Hagelaar *Modeling of Microdischarges for Display Technology*, PhD thesis, Eindhoven University of Technology, The Netherlands, 2003.

[31] V. Žigman *Influence of Anisotropic Elastic Scattering on the Electron Energy Distribution in Weakly Ionized Xenon Plasma* International Conference on Atomic and Molecular Data and Their Applications, Oxford, March 2000.

[32] H.W. Ellis, R.Y. Pai, E.W. McDaniel, E.A. Mason and L.A. Viehland *Transport Properties of Gaseous Ions over a wide Energy Range* Atomic Data and Nuclear Data Tables 17, 177-210 (1976)

[33] W.J.M. Brok, J. van Dijk, J. J. van der Mullen, *Theory, Design, and Implementation of the EPG Monte Carlo Code*, Technical Report, Group Elementary Processes in Gas Discharges, Eindhoven University of Technology. See also http://plasimo.phys.tue.nl/micro_dis/.

[34] Jan van Dijk *Modelling of a High Pressure Closed Inductively Coupled Plasma for Lighting Purposes*, Masters thesis Eindhoven University of Technology, 1996

[35] Colin Johnston, *Transport and equilibrium in molecular plasmas: the sulfur lamp*, Eindhoven University of Technology 2003, thesis.

[36] V. Žigman *The viscosity cross-section for elastic electron-xenon collisions including electron spin polarization* Eur. Phys. J. D. 7, 11-16 (1999) (actual data from McEachran and Stauffer obtained by V. Žigman through private communication.)

[37] MAGBOLTZ is available for download from http://consult.cern.ch/writeup/magboltz/. Notes on usage and the cross sections are available from the same location. The program is sometimes also refered to as MAGBOLTZ-MONTE to differentiate it from an earlier version which did not use Monte Carlo methods. See also : S.F. Biagi, *Accurate solution of the Boltzmann transport equation* NIM A273 (1988) 533.

[38] J.T.Fons and Chun C. Lin, Measurement of the cross sections for electron-impact excitation into the $5p^5 6p$ levels of xenon. Phys. Rev. A 58, 4603-4615 (1998)

[39] A. A. Sorokin et al. *Measurements of electron-impact ionization cross sections of argon, krypton, xenon by comparison with photoionization.* Phys. Review A. 61-2(2000), vol. 61 (2000), p 22723

[40] E. W. McDaniel Collision Phenomena in Ionized Gases, John Wiley & Sons, Inc., 1964

[41] S.V. Patankar : *Numerical Heat Transfer and Fluid Flow*, Hempisphere Publishing Corporation (1980)

[42] B. Stroustrup The C++ programming language, 1997 AT&T

Appendix A

Code Issues and Implementation

A.1 Object Oriented Programming

Both PLASIMO and the Monte Carlo code are object oriented in design. This means that data and functions (or *methods* as they are called in C++) are grouped together in data structures called *classes* in C++. Instances of such a data structure are called objects, hence the name *object oriented programming*. The access to the data and functions between classes goes through an *interface*. In effect, most of the data a class uses is shielded from other classes to avoid this data being accidentally overwritten. In C++ the interface between classes is formed by the *public* members of that class. The *private* data, on the other hand, is shielded off from other data by the compiler.

Object oriented programming languages also know the concept of *inheritance*. A class can be *derived* from a *base* class. The derived class then *inherits* all the data and member functions from that *base* class. This process is known as *inheritance* and is fundamental to object oriented design as it allows one to express the degree of communality between data structures.

There is neither time nor space in this report to delve into great depth on the topic of object oriented programming. Readers interested in C++ and object oriented design should check out Stroustrup's book on the matter [42]. The design of PLASIMO is discussed in Van Dijk [25]. The latter also discusses the use of run time extensions in PLASIMO .

A.2 The Plasimo/Monte Carlo Bridge Class

In order to exchange information between the Monte Carlo code and PLASIMO a bridge class is needed. This bridge class is derived from the *mcenvironment* class [33] to pass on information required for the calculation of the path of the particles, such as the electric and magnetic field strength in a certain point.

The public and protected members of the class are as follows:

```
class mcEnvironment_pl : public mcEnvironment
{
  public:
        /** Constructor */
        mcEnvironment_pl(plSpeciesModelRegion * region);
        /** Destructor */
        virtual ~mcEnvironment_pl() {}
        //returns md2d style material properties at \a pos.
        virtual unsigned VesselMaterial(const mcGeomVector<3>& pos) const
        {       return (!InsideVessel(pos));
                /**Plasimo doesn't know about materials, so
                  *return 1 if outside of vessel
                */
        }
        /** Returns the normal on the tangent plane through pos,
          * Given a reference to a position pos the function
          * finds the nearest wall point and changes pos accordingly,
          * A reference to the norm (pointing inwards) is also returned.
          * This function should be useful for anything to do with
          * wall-reactions. The work is done by the SymmetryTransform
        class */

        virtual const mcGeomVector<3> & VesselWall(mcGeomVector<3>& pos) const
        {
                return m_sth->VesselWall(pos, m_region->Grid()->cd1(),
                                    m_region->Grid()->cd2());
        }
        /** Whether \a pos is inside the vessel: */
        bool InsideVessel(const mcGeomVector<3>& pos) const
        {
                // Our present InsideFDGrid func only works for non-OCL grids
                assert (m_region->Grid()->Type()!=GridDefs::OCL);
                return InsideFDGrid( (*m_region->Grid())(plGrid::LocNP),
                                    m_sth->Project(pos) );
        }
        mcGeomVector<3> GenRandLoc() const;
        //const accessors for help with 2D<=>3D conversions
        const Locate& LocateHelper() const { return m_loc;}
        const SymmetryTransformBase* STH() const { return m_sth;}
  ///accessor for the ion. rates
        plPtrVector<MCRateDensity>& Rates() { return m_rate_densities;}
        /// initialize above rates:
        void InitRates(const mcProcessContainer& procs);
protected:
        /* returns a reference to the electric field at \a pos. */
        virtual mcGeomVector<3>& E(const mcGeomVector<3>& pos) const;
        /* returns a reference to the magnetic field at \a pos. */
        virtual mcGeomVector<3>& B(const mcGeomVector<3>& pos) const;
         /** Function which should be called at the beginning of a simulation
          * to get the density swarms into the swarmlist \a swl. */
        virtual void AddDensitySwarms(const mcSpeciesList& spl,
                                      mcSwarmList& swl);
         /** Particle Injector: this function is to be called once, at the
          * start of the simulation, to inject the particles to be started off
          * with. */
        virtual void InjectParticles(mcSwarmList& swl, double t=0.0);
        /** Particle Injector: given time \a and timestep \dt this function
         * can inject particles into members of \a swl. */
        virtual void InjectParticles(double t, double dt, mcSwarmList& swl);
        // initialize the rates container:

                                    ⋮
```

Member functions such as *VesselMaterial(...)*, and *E(...)* are implementations of member functions of the *mcEnvironment* class that this class is derived from. The constructor takes a pointer to an object of the *plSpeciesModelRegion* type. This object is used in PLASIMO to store information (or references to that information) that is relevant to a region used in a model of the *plSpeciesModel* type. One of the member functions, for example, returns the grid for the region in question. In a slightly more indirect manner information such as the electric and magnetic field can also be obtained. The *mcenvironment_pl* class uses information from PLASIMO and returns them in a format required by the Monte Carlo code.

A.2.1 Interpolation and Projection

Since PLASIMO uses two dimensional grids of various types and the Monte Carlo code uses 3D grid-less positions and velocities one needs to implement methods to project these 3D positions on 2D grids and interpolate the quantities on the grids back to values in 3D space. To this end a number of classes are introduced. A class *Locate* is used to find which cell a particular point is in. This uses a simple recursive search algorithm to find the nearest discrete gridpoints; starting with the gridpoint it would be found on in an equidistant grid. An pure virtual class *SymmetryTransformBase* and three derived classes (*SymmetryTransformCylindrical* , *SymmetryTransformCartesian* and *SymmetryTransformSpherical*) are used for transformations from 2D to 3D and vice versa.

The full listing of *SymmetryTransformBase* is as follows:

```
class  SymmetryTransformBase
{
 public:
        ///c'tor
        SymmetryTransformBase() {}
        virtual ~SymmetryTransformBase() {}
        typedef mcGeomVector<3> mcvec3D;
        const RPos& Project(const mcvec3D& pos) const
        {       return DoProject(pos);
        }
        const mcvec3D& TransformTo3D(const double& vec1, const double& vec2,
                                const mcvec3D & pos) const
        {       return DoTransform(vec1, vec2, pos);
        }
        /**Get the minimum and maximum x,y and z coordinates based
         * on the properties of the 2D coordinates.*/
        void GetMinMax3D(mcvec3D& minxyz, mcvec3D& maxxyz,
                const plCoordDef& c1, const plCoordDef& c2) const
        {       MinMax3D(minxyz, maxxyz, c1, c2);
        }
         /** Returns the normal on the tangent plane through pos,
          * Given a reference to a position pos the function
          * finds the nearest wall point and changes pos accordingly,
          * A reference to the norm (pointing inwards) is also returned.
          * This function should be useful for anything to do with
          * wall-reactions.
        */
        const mcvec3D& VesselWall(mcvec3D& pos, const plCoordDef& c1,
                        const plCoordDef& c2) const
        {
                return GetVesselWall(pos,c1,c2);
        }
        /**expand a 2d vector, given an extra coordinate zeta between zero
        *and 1 to a 3d carte Sian vector in such a way that all of space can be
        *addressed.
        */
        const mcvec3D& Expand(const RPos& pos2d, double zeta) const
        {
                return DoExpand(pos2d, zeta);
        }

protected:
        ///Project a 3d vector onto a 2D surface
        virtual const RPos& DoProject(const mcvec3D& pos) const=0;
        ///given the two components of a vector and the position,
        ///transform this to a 3D vector.
        virtual const mcvec3D& DoTransform(const double& vec1,
                                          const double& vec2,
                                    const mcvec3D& pos) const=0;
        ///Get a box surrounding the area
        virtual void MinMax3D(mcvec3D& minxyz, mcvec3D& maxxyz,
                const plCoordDef& c1, const plCoordDef& c2) const =0;
        virtual const mcvec3D& GetVesselWall(mcvec3D& pos,
                        const plCoordDef& c1,
                        const plCoordDef& c2) const = 0;
        virtual const mcvec3D& DoExpand(const RPos& pos2d,
                              double zeta) const =0;
        //optimization:
        mutable RPos m_pos2d;
        mutable mcvec3D m_vec3d, m_norm, m_expanded;
};
```

Notice that the protected member functions are all virtual. These are geometry dependent and are all implemented in the derived classes. For example, the *DoProject* member of the *SymmetryTransformCylindrical* class is implemented as follows

```
const RPos& SymmetryTransformCylindrical::DoProject(const mcGeomVector<3>& pos) const
{
        m_pos2d.Y()=sqrt(pos[0]*pos[0]+pos[1]*pos[1] );
        m_pos2d.X()=pos[2];
        return m_pos2d;
}
```

while for the *SymmetryTransformCartesian* class the transformation is the trivial:

```
const RPos& SymmetryTransformCartesian::DoProject(const mcvec3D& pos) const
{       m_pos2d.X() = pos[0];
        m_pos2d.Y() = pos[1];
        return m_pos2d;
}
```

The *mcEnvironment_pl* class' constructor then examines what type of grid is being used and then constructs one of the transform classes. In the *mcEnvironment_pl* constructor one finds :

```
        switch(region->Grid()->Type())
                {
                        case GridDefs::Cylindrical :
                                m_sth = new SymmetryTransformCylindrical;
                                break;
                        case GridDefs::Cartesian :
                                m_sth = new SymmetryTransformCartesian;
                                break;
                        case GridDefs::Spherical :
                                m_sth = new SymmetryTransformSpherical;
                                break;
                        default:
                                throw plException("Non-standard (OCL) grid");
                }
```

A.3 Interfacing Monte Carlo Data with PLASIMO

Apart from obtaining information such as the magnitude and direction of the electric field at a given point for use in the Monte Carlo Code one also needs to get information from the Monte Carlo code back to the fluid model. This work is divided over a number of classes. The *mcStatistics_pl* class collects information on collisional processes such as ionization events. Through the input file one can specify a list of processes for which statistics are to be gathered. Every time such an event takes place the member records the position in 3D space at which this event has taken place, the number of real particles involved, the mass of these particles and the time at which this collision took place. This information is stored in a buffer. At the beginning of every time step in the fluid model the information from the last time step is processed for all processes in the list, after which the buffer is cleared of old events. The processed information is stored in a format that PLASIMO understands. The public members from the header file are as follows:

```
class mcStatistics_pl : public mcStatistics
{
  public:
```

```
mcStatistics_pl (const plNode& node)
          : mcStatistics(node) {
          m_data_buffer.resize(0);
          m_indxs.resize(0);
}
~mcStatistics_pl() {}
virtual void AtCollision(unsigned procindex,
          const mcParticle& p1, const mcParticle& p2)
{         //just pop it into the buffer and deal with it later:

          if (std::find(m_indxs.begin(),m_indxs.end(),procindex)
                    ==m_indxs.end())
          {
                    Log(1)<<procindex
                              << " not in list. Not added to buffer.\n";
                    return;
          }
          m_data_buffer.push_back(
                    new CollisionData(procindex,p1,p2));
                    Log (1) <<"added collision data to buffer for procedure"                              <<procinc
          return;
}
///calculate the density of collisions
/** Calculate the collision density rate per volume cell matching
 * procindex between the times t1 and t2. Needs a reference to a Locate
 * object and a SymmetryTransformBase pointer
 * to project the 3D positions onto the 2D grid and find the appropriate
 * cell*/
void RateDensity(plGridVar<double>& dens,
                    unsigned procindex,
                    double t1,
                    double t2,
                    const Locate & lh,
                    const SymmetryTransformBase* sth) const;
//add the index of a proc. index to keep track off:
void AddProcIndex(unsigned i) { m_indxs.push_back(i);}
virtual void WriteFinalResults() {}
///get rid of data older than time t.
void ClearBuffer(double t);

                                        ⋮
```

Note that the *RateDensity* member uses a reference to an object of the *plGridVar* type. The information is actually stored inside an object of the *mcEnvironment_pl* type, where the object is also initialized. The reason for this is that the statistics class has no knowledge of PLASIMO 's grids. This information is privy to the *mcEnviroment_pl* class. The *mcEnviroment_pl* class also contains a reference to the modelregion, which in turn, can be used to access an object of the *plFieldValueMap* type. This object can be used to access a wide variety of data for different grid locations and particles. See [28], section 4.2 on a discussion on how this is implemented. The rates per recorded process are stored per gridcell on the two dimensional grid. References to objects containing this data are stored in the *plFieldValueMap* object. Plugins, used to calculate rates in PLASIMO can access this object, and through it get the information required from the Monte Carlo part of the code.

A.4 Model Plugins

PLASIMO can be used to calculate many different types of problems. These all require different models. The models themselves are implemented as plugins. The hybrid kinetic-fluid model is itself a plugin. Since the fluid part of the code is essentially the same as the pre-existing non-LTE model the hybrid model is derived from the nLTE model. The class definition from the header file is shown below.

```
//Derive from plPlasmaNLTEModel since Hybrid MC−fluid models are
//going to be far  from LTE.
class NLTEHybridModel : public plPlasmaNLTEModel
{
  public:
        NLTEHybridModel(const plNode &node) : plPlasmaNLTEModel(node)

        {       m_fc. resize (NRegions());
                m_env.resize(NRegions());
                m_stat. resize (NRegions());
                //just  to  be on the  safe  side :
                for  (unsigned i=0; i<NRegions(); ++i)
                {         m_env[i]=0; m_env[i] =0; m_stat[i]=0;
                }
                Log(2) << "Creating_hybrid_model" << std::endl;
        }
        const mcFlightControl& FlContr(unsigned i) const { return (*m_fc[i]); }
        mcFlightControl& FlContr(unsigned i) { return (*m_fc[i]); }
        mcStatistics&  Statistics (unsigned i) { return (*m_stat[i]); }
        virtual void Prepare();
        virtual ~NLTEHybridModel(){ }
        bool Finished() const;
        virtual void Update();

  private:
        //What's in a name? Helper function used by Update();
        void DoMCStuff();
        void Cleanup();
        //these are pointervectors, to allow for multiple regions.
        //However, no interaction between regions is taken into account.
        plPtrVector<mcFlightControl> m_fc;
        plPtrVector<mcEnvironment_pl> m_env;
        plPtrVector<mcStatistics_pl> m_stat;
};
```

Notice the private members `plPtrVector<mcFlightControl> m_fc` , `plPtrVector<mcEnvironment_pl> m_env` and `plPtrVector<mcStatistics_pl> m_stat` . These are linked lists containing pointers to objects of the types indicated in the template argument. The class *mcFlightControl* is responsible for the kinetic model. It tracks the progress of the superparticles and sets the collision times. It also calls functions that handle the collisions. The *Update* member is repeatedly called by PLASIMO until the boolean member *Finished* yields a positive answer.

The implementation of this member is as follows:

```
void NLTEHybridModel::Update()
{
        if (TimeStepping()&&plPlasimoModel::Finished()&&CheckTimeStep())
        {
                DoMCStuff();
                //tell the statistics classes
                //to gather rates.
                const double t2 =Time();
                const double t1 = Time() − TimeStep();
                for (unsigned i=0;i<m_env.size(); ++i )
                { //iterate over everything we want a rate for :
                        const plPtrVector<MCRateDensity> & rates =
                                m_env[i]−>Rates();
                        //get rid of old data:
                        m_stat[i]−>ClearBuffer(t1);
                        Log(1) <<"Getting the rates for "
                                <<rates.size() << "processes\n";                    for (unsi
                        {
                                Log(1) <<"Obtaining rate nr" << ri <<"\n";
                                m_stat[i]−>RateDensity(rates[ri]−>GVar(),
                                        rates[ri]->ProcIndex(),
                                       t1,t2,m_env[i]->LocateHelper(),
                                       m_env[i]->STH());
                        }

                }
        }
        plModel::Update();

}
```

Notice the first statement. This line checks if the fluid part of the model has converged for that timestep. If this is the case it moves on with the kinetic part of the model. The member function *DoMCStuff* simply iterates over all flightcontrollers telling each of them to proceed. The next part calls on the *environment* to get a reference to the object which stores the rates, and then passes this on to the *mcStatistics_pl* type object to fill with the actual rates. Finally, the fluid model is told to update its values.

A.5 The *plMCRelationGroup* class.

Processes such as ionization but also inelastic collisions in general are grouped together in objects of the *plRelationGroup* type. These objects store references to objects representing the different relations. The relation type objects carry a reference to an object responsible for calculating the rate coefficient. The Monte Carlo code calculates the actual rates rather than the rate coefficient. To be able to use the rates rather than rate-coefficients several new classes were introduced: a class *plMCRelationGroup* derived from *plRelationGroup* to store all the relations depending on the Monte Carlo calculations, a class *plDirectParticleRelation* to allow for relations which depend on the calculated rate rather than the rate coefficient, and a class *plMCRate* derived from the *plRateCoefficient* class that retrieves the rate from the Monte Carlo calculation and passes it on to an object of the *plDirectParticleRelation* type. See figure A.1 for a schematic view, as generated by Doxygen[1].

[1]Doxygen, developed by Dimitri van Heesch, is used to generate developer's documentation of source code. It is used for both MD2D and PLASIMO . It is available from http://www.doxygen.org.

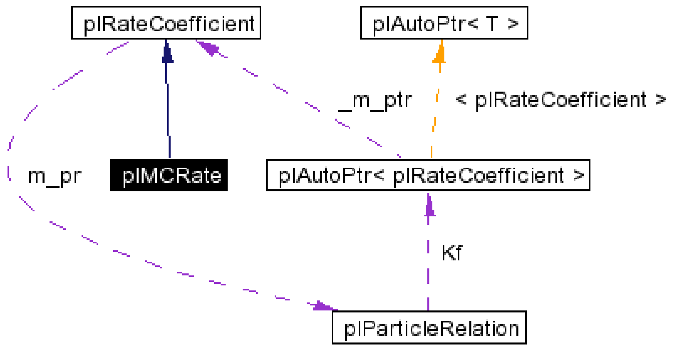


Figure A.1: Relation graph as generated by Doxygen for the class *plMCRate*.

A.6 The Poisson Solver

The implementation of the method of solving the Poisson equation by writing it as a Φ equation is spread over a number of classes. the class *plPoisson-Variable*, derived from *plPhiVariable*, initializes the "diffusion" term, with ϵ_0 as the diffusion coefficient on the entire grid. the potential and the charge density are defined in the *plEMPoissondata* class[2], which also updates the potential and calculates the electric fields from the gradient of that potential. the *plEMPoisson* class calls the field calculations for each region. For testing purposes a new model has been defined, which is used to solve the poisson equation for a given, fixed, charge density. the class for this model, called *plEMBoundedModel* is derived from *plGridModel*.

[2]actually the potential is a derived member, *plEMPoissondata* just contains the charge density.

plPoissonVariable	
derived from	*plPhiVariable*
methods:	*plPoissonVariable* (constructor) the constructor takes a model region, a name and a node as arguments and initializes the diffusion term of the ϕ – equation with ϵ_0 as the "diffusion coefficient".
members:	*constterm* class. this class is derived from *plDoublePhitermContribution* and it is used in the constructor above to produce a constant field for the "diffusion coefficient".
defined in	`plgrid/poisson.h` and `plgrid/src/poisson.cpp`

plEMPoissonData	
derived from	*plEMStaticPotentialdata*
methods	*plEMPoissonData* (constructor) the constructor takes a model region and a node as arguments and registers the net charge (read from the node) as the source term of the ϕ – equation. *netcharge()* this returns the net charge, as the name suggests. *nonzerorho()* this returns one. *nonzerorho* returns one if charge density is taken into account (which is the case in this class), and zero if it is not (in the *plEMStaticPotentialData* class).
members:	*m_rho* the charge density. *m_rho_cons* a private member.
defined in	`plem/plugins/em_poisson.cpp`.

plEMPoisson	
derived from	the *plBaseEMProxy* template using *plEMPoissonData*.
methods:	*plEMPoisson* (constructor) simply calls the *plBaseEMProxy* constructor.
	doiteration(real) this function calls *calculatefields(real)* for each region.
	createplasma(plModelRegion, node)* this method calls the *plEMPoissonData* constructor and returns a pointer to the region.
defined in	`plem/plugins/em_poisson.cpp`.

plEMStaticPotentialData	
derived from	*plBaseEMData*
methods:	*plEMStaticPotentialData(plModelRegion*, plnode)* (constructor) the constructor initializes the potential and the boundary conditions.
	calculatefields(real) this function updates the potential and calculates the electric fields and the dissipation density.
	accuracy() this returns the accuracy with which the ϕ–equation has been solved.
	nonzerorho() this returns zero (see the description for *plEMPoissonData*).
members:	`m_potential` the potential, of the type *plPoissonVariable*.
defined in	`plem/plugins/em_pot_static.h` and `plem/plugins/em_pot_static.cpp` .

A.7 Diffusion Without Quasi Neutrality

Plasimo can be used with several different diffusion models. Prior to the addition of the non quasi-neutral diffusion these were: Fick diffusion, "Knudsen" corrected diffusion, ambi-polar diffusion and full self-consistent diffusion. Several combinations (combining the Knudsen correction with ambipolar diffusion, for example) were also possible, but all implied quasi-neutrality. To implement the non quasi-neutral diffusion a new diffusion manager class was added. This diffusion manager class adds objects representing the density for each species. For each of these objects, including the one representing the electron density a Φ equation is solved. For the charged particles the drift velocity is calculated using the mobility and the local electric field. The calculation of the drift velocity thus assumes the validity of the drift diffusion equation. The mobility of the charged species comes from a plugin. Two have been made thus far: one using a lookup table, and one using the Einstein relation and the diffusion coefficient.

A.8 Anisotropic Diffusion

The implementation of anisotropic diffusion is spread over several classes. To calculate anisotropic diffusion coefficients a plugin is needed. A base class for this plugin is provided called *plAniDiffusion*. The implementation of anisotropic diffusion is spread over several classes. To calculate anisotropic diffusion coefficients a plugin is needed. A base class for this plugin is provided called *plAniDiffusion*.

```
class plAnisotropicDiffusion : public plDiffusion
{
  public:
        /** constructor. Calls the base class constructor,
          * m_flag is initialized to zero.
         */
        plAnisotropicDiffusion( const plNode& node,
                                const plParticleMap& pmap,
                                const plBaseRelationMap& rmap)
        : plDiffusion(node,pmap, rmap) , m_flag(0)
        { }
        ///Set the value of m_flag (see DoCalculate comments).
        virtual void SetFlag(int flag) { m_flag=flag;}
        ///Returns value of m_flag:
        virtual int GetFlag() { return m_flag;}
        /** Depending on the value of m_flag DoCalculate returns either the
        *diffusion coefficient in the direction of the first (m_flag=0)
        *or the second coordinate (m_flag != 0). This is the only thing
        * you need to implement in the derived class.
        */
        virtual void DoCalculate( const plParticleValue<REAL> &res,
                                  const plCrossSectionMatrix & cs,
                                  const plConstValueRef<REAL> & point )=0;
 private :
        /**store the direction we want the diffusion for:
```

```
        * m_flag = 0 :  first  dimension c1 in grid.
        * m_flag = 1 : second direction: c2 in grid.
        * Used in DoCalculate, set with SetFlag(int flag ), access with
        * GetFlag(). Setting is done in the class plAniDiffTerm.
        */
        int m_flag;
};
```

Plugins for anisotropic diffusion should have a member *DoCalculate* that returns a diffusion coefficient in the direction specified. For historical reasons, the direction is not an argument of *DoCalculate* but has to be set seperately. Depending on the origin of the anisotropy the one can use different plugins to obtain different results. A plugin using anisotropic temperatures and a generalized Einstein equation to arrive at anisotropic diffusion coefficients has already been written, but someone wanting to study magnetized plasmas would only have to write a new plugin.

Using anisotropic diffusion also has its effect on the Φ equation. For isotropic diffusion the diffusion coefficients are calculated using the values at the center of the grid cell. These are then interpolated to the boundaries of each grid cell, where they determine the flux across the grid cell's boundary. The anisotropic diffusion coefficients also typically depend on quantities known for the center of the grid cell. To solve the Φ equation for density in the case of anisotropic diffusion coefficients two sets of coefficients are kept, and interpolated to the appropriate boundaries.

Anisotropic diffusion can be turned on or off at will in the input file.

Appendix B

The Introduction of Sub-regions, a Proposal

B.1 Introduction

The finite volume methods on structured grids used in PLASIMO are ill suited for more complex geometries. There are many geometries one would like to investigate that cannot currently be handled (the hollow cathode geometry) or can only be handled with difficulty and highly specific plugins (the cascading arc).

The use of ortho-curvilinear grids is appropriate for smooth geometries such as the QL-lamp but it does not work well with sharper boundaries such as jumps in diameter in a cylindrical geometry. One method of dealing with irregular geometries in combination with finite volume methods on a orthogonal grid is to block off regions of the grid[41], as in figure (B.1). Given different regions on the grid one can then proceed to "maltreat" the different terms of the Phi equation accordingly. To fix the value of a variable described by the phi equation, for example, one can set large source terms (both linear, and constant) so that the effect of the other variables vanishes. There are many other ways one can handle or abuse the differential equations once different regions can be specified. In flow calculations, for example one can set the viscosity at large values for the blocked-off regions so that effectively no flow occurs in those regions. To avoid confusion with the "regions" concept in PLASIMO I will refer to these blocked-off regions as *"subregions"*.

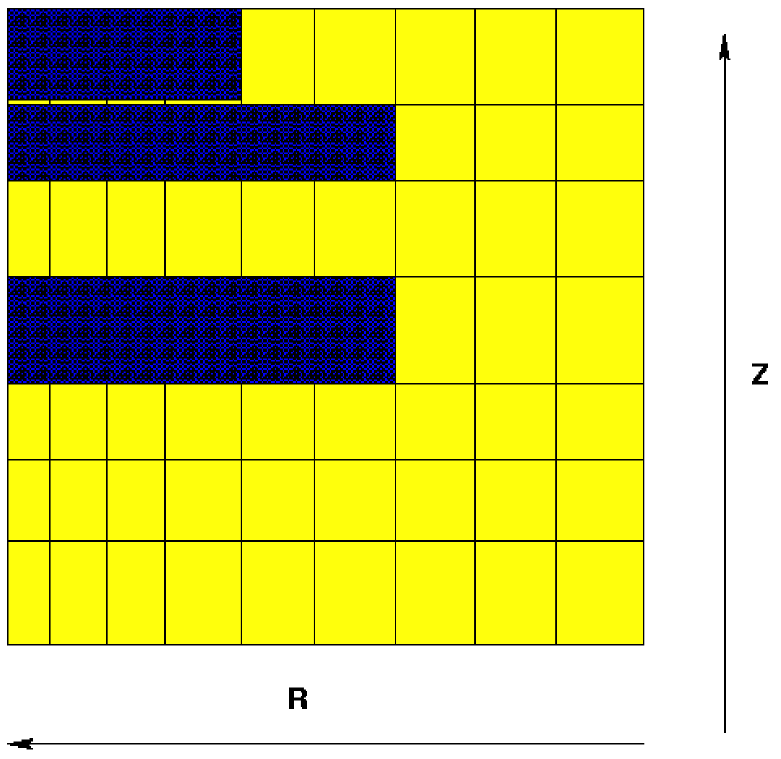


Figure B.1: Blocking off regions on a coarse grid.

B.2 Specifying and Parsing Subregions

If one is to allow subregions there has to be a mechanism in place to allow specification and parsing of these regions by PLASIMO through the input file. The method of doing so has to be flexible to allow for many different geometries while not frustrating users who do not need the subregions with demands for irrelevant information. The changes made should not "break" old input files - these should still function as before. One way of doing this would be to introduce polygons, specified by the positions of their nodes on the carte Sian projection of the grid. The input file than specifies a series of regions, and each regions is a collection of points defining a polygon (see figure B.2). For each gridvariable (and by extension derived classes) it will then be possible to specify different properties for different sub-regions. This will be done along the lines of Bart Broks "time schedule blocks". That is, failing more information in the input file the same properties are assumed to hold on the entire region.

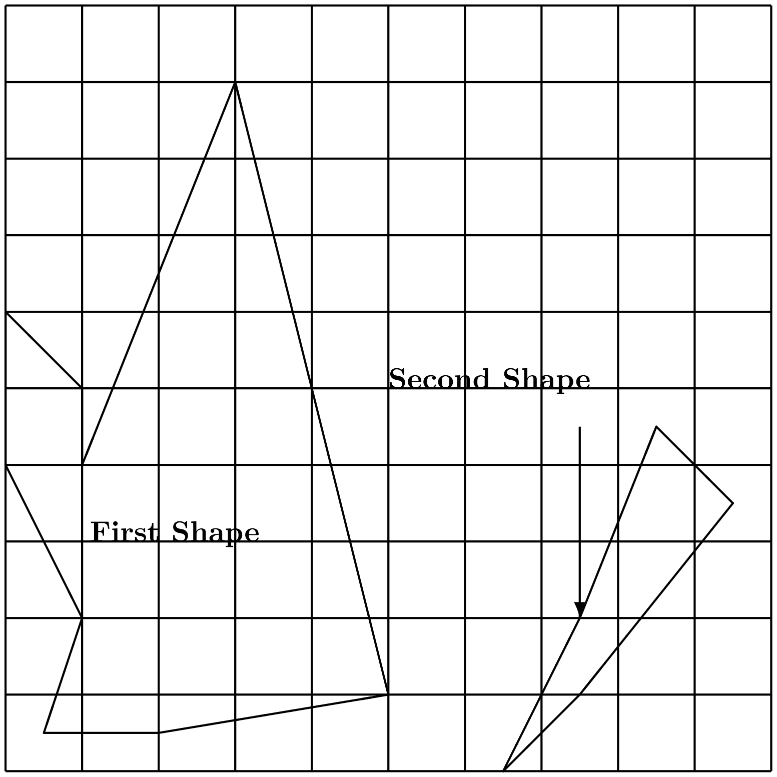


Figure B.2: Some polygons on a grid.

B.3 The Implementation

To implement the subregions a number of classes are defined. One class handles the definition of the polygons (a collection of points and lines) using the node to construct them. This class also has a member function which returns a boolean indicating if a specified point is on its inside or outside. Additionally, the polygon shape class will hold a name tag. Another class holds a collection of these polygonal shapes and has a member function indicating in which polygon a specified point is in. One could follow the MD2D convention and return 0 for a point that is not inside any sub-region and an integer numbering the sub-region otherwise. For purposes of avoiding repeated use of the point location algorithm it should also contain four matrices (one for grid location) with an unsigned integer indicating which sub region each point on each grid is in. Thus, basically, an MD2D type geometry will be generated on construction of this class, and on each subsequent grid resize this geometry will be updated. This approach will allow greater flexibility than the MD2D approach since the user will not be required to redefine a new geometry file every time he or she decides to change the number of gridpoints. Care will have to be taken, however, to update the matrices every time the grid is changed.

The plPolygons class would look something like this (for the sake of brevity the implementation of inline functions is omitted):

```
class plPolygons{
  public:
        //The c'tor, from node.
        plPolygons(const plNode& node);
        virtual ~plPolygons() { }
        //return the Label of the Polygon with index index.
        std::string ShapeLabel(unsigned index) const ;
        //find the Shapenumber the testpoint is in, return
        // 0 if not in any of the defined shapes.
        unsigned ShapeNumber(const RPos& testpoint) const;
        //return the Shapenumber for points on a grid
        unsigned ShapeNumber(const Pos& p, plGrid::Location loc) const;
        unsigned size() const;
        //Update the matrices used for the function Shapenumber(p,loc)
        //Call this after changing the grid.
        void UpdateMatrices(plModelRegion* reg);

  private:
        /*The class defining a polygon is called plShape
        This is not to make things confusing but to avoid typo's as
        having one class called plPolygon and one called plPolygons is
         likely to cause trouble. We store a pointervector of these shapes.
        Alternatively one could derive this class from plPtrVector<plShape>.
        */
        plPtrVector<plShape> m_shapes;
       /*The index of the last plShape found to contain a point.
        Since the member unsigned ShapeNumber(const RPos& testpoint) is
         likely to be called consecutively for points close to each other it
         is efficient to start with the last ShapeNumber.
        */
        int m_it;
        /*The shapenumbers on the points of the grid for each grid location.
         (use enum location) */
        plGridVar<unsigned>[4] m_geom;
};
```

Another class then uses the above to manage functions for a specific property (e.g. a diffusion coefficient). It would be somewhat similar to Broks' helper class [26] for the time blocks.

```
template <class T>
class plInteriorRegionHelper
{
  public:
        plInteriorRegionHelper( plPolygons* InteriorRegions,
                                const std::string Param,
                                 const std::string ParamUnit);
        ~plInteriorRegionHelper() {}
        //sanity checker for an interion region,
        void TestRegion(const std::string function,
                                    const plNode& node);
        //sanity checker for all int. regions
        void ParseRegions(const plNode& node);
        // Get the function belonging to one of the regions, given a
        std::string GetFunction(const plNode& IRnode,
                                 const RPos& testpoint) const;
        // Same thing but on a grid point
        std::string GetFunction(const plNode& IRnode,
                                   const Pos& testpoint,
                                   plGrid::Location loc) const;
        //Get the value on a point
        T SetParam(const plNode& IRnode, const RPos& point) const;
        //And again overloaded for gridpoints:
        T SetParam(const plNode& IRnode, const Pos& point,
                   plGrid::Location loc) const;
  private:
        std::string m_param, m_unit;
        plPolygons* m_int_regions;

};
```

Using the helper class above gridvars can be initialized or calculators modified/added to allow for different properties on different sub-regions.

B.4 Boundary Conditions

Using interior or sub-regions in the manner described in this proposal leaves open the problem of boundary conditions. In a mathematical sense there are no interior boundaries when solving the resulting differential equations. In practice this poses a problem for the modeling of reactions on the wall. In the hollow cathode, for example, the secondary emission of electrons is an important source of high energy electrons. On a side note, one is not really interested in the solution in many of the sub-regions. One possible solution is, therefore, to have secondary emissions of electrons in much of the sub-region representing a part of the wall. This results in an electron density and temperature which is unreasonably high in this section, but this is not important if one is not interested in the solution in that section. Similar problems arise with the reflection of heavy particles.

Dankwoord

Hierbij wil ik mijn grote dank uitspreken voor mijn begeleiders en collega's zonder wie dit werk niet tot stand had kunnen komen. In het bijzonder wil ik mijn begeleider Bart Broks bedanken die mij wegwijs heeft gemaakt in PLASIMO en met wie ik veel vruchtbare discussies gehad heb. Verder wil ik Wouter Brok bedanken voor zijn hulp bij het gebruik van de Monte Carlo code van Micro-dis, het verschaffen van inzicht in hoe Monte Carlo methoden werken en voor zijn hulp bij veel praktische problemen bij het computer gebruik.

Verder bedank ik Joost van der Mullen voor het verschaffen van beter inzicht in de plasmaphysica, Erik Kieft voor het met mij delen van de nodige experimentele gegevens, Bart Hartgers en Jan van Dijk voor het leren programeren van C++, Jeroen Jonkers en met hem Philips Extreme UV GmbH in Aachen voor hun ondersteuning en Rob Veenhof van CERN voor zijn ondersteuning bij het gebruik van MAGBOLTZ.