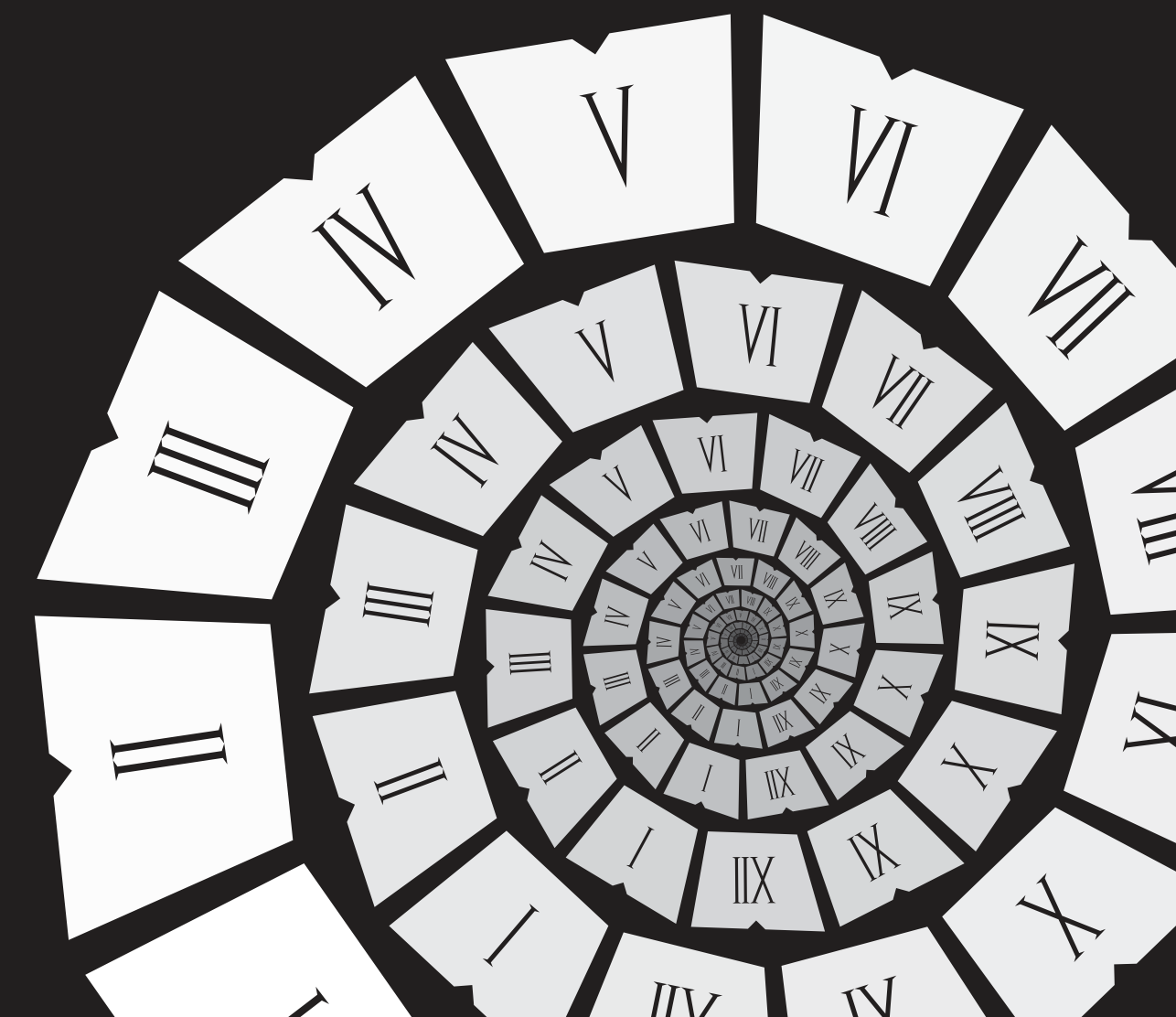
Zero-dimensional models for plasma chemistry

Zero-dimensional models for plasma chemistry

Wouter Graef



Zero-dimensional models for plasma chemistry

Wouter Graef

This research was financially supported by the CATRENE SEEL project (CA502)

CIP-DATA TECHNISCHE UNIVERSITEIT EINDHOVEN

Graef, Wouter Antonius Anna David

Zero-dimensional models for plasma chemistry / by Wouter Graef. -
Eindhoven : Technische Universiteit Eindhoven 2012. - Proefontwerp.

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-386-3168-4
NUR 928
Subject headings : plasma physics / plasma modeling / plasma chemistry / computer simulations / EUV lithography / sputtering / photoionization / non-equilibrium plasmas / software design

Printed by: Ipskamp Drukkers B.V.

Typeset in LaTeX 2$_\varepsilon$ using the Vim editor.
Figures made with PyX, Inkscape, and GIMP.
Cover made with Scribus.

Zero-dimensional models for plasma chemistry

PROEFONTWERP

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op dinsdag 26 juni 2012 om 16.00 uur

door

Wouter Antonius Anna David Graef

geboren te Horn

De documentatie van het proefontwerp is goedgekeurd door de promotoren:

prof.dr. J.J.A.M. van der Mullen
en
prof.dr. W.J. Goedheer

Copromotor:
dr.ir. J. van Dijk

Summary

Zero-dimensional models for plasma chemistry

In this thesis two different models for examining the *chemistry* in a plasma are studied. Both are zero dimensional, meaning that the *configuration* and *transport* aspects of plasmas are combined and reduced to frequencies. The first type of models are Collisional Radiative Models (CRM), in which the atomic state distribution function can be seen as being composed by contributions of a limited number of atomic states, typically the atom and ion ground state. Transitions between levels are facilitated by electrons and photons. The second type of models are Global Plasma Models (GPM), whose goal it is to predict mean values of internal plasma parameters as a function of external control parameters.

Collisional Radiative Models

The tasks of a CRM are threefold: calculating the Atomic State Distribution Function (ASDF), the effective conversion frequencies, and the source terms for the energy equation. In a general exploration, the derivation is described to structure the various collisional and radiative processes into simple matrix equations. Initially, a Quasi Steady State (QSS) solution is pursued, but the results are extended to a time dependent solution. The properties of the agents facilitating the transitions (electrons and photons), are allowed to be of a transient nature.

This description of a CRM is implemented in the PLASIMO framework; a plasma modeling platform. The addition of a CRM in the form of a model plug-in to PLASIMO is described, as well as the details for constructing and solving the CRM, both for the case of QSS and transient plasmas.

The PLASIMO CRM is used in two applications: a CRM of an Extreme UltraViolet (EUV) driven plasma in Ar, and a Laser Induced Fluorescence experiment in Ar plasmas. Both models are time dependent; in the first application due to the electrons created by an EUV pulse, in the second application due to a laser pulse.

The CRM of the EUV driven plasma gives insight into the time dependent spectrum of the plasma. It is shown that optical emission spectroscopy can be used to monitor the electron energy in the plasma, and thereby the sputtering action of ions.

The CRM of an Ar LIF experiment helps in understanding the mechanisms that follow a perturbation imposed on the Ar system by a laser pulse. The results of the model are compared to the results of experiments on a Surfatron Induced Plasma in which the electron density and temperature are well known.

Global Plasma Models

A GPM is also implemented as a model plug-in for plasimo. As in the CRM plug-in, the species densities are modeled, but this is extended with the modeling of the electron density and electron energy balance. The model is a collection of chemical reactions, and the external control parameter is the input power density.

This model plug-in is used to model a High Power Impulse Magnetron Sputtering plasma. Although the model is zero dimensional, sputtering is included in the form of a frequency of particle density entering the plasma; the same as dealing with other transport aspects. The model plug-in is a valuable tool in studying the relevant processes in the creation of species.

Contents

Chapter 1

General introduction

Plasma is a gas of which a substantial amount of the particles is ionized, allowing charges to flow freely, and giving it distinct properties, mainly in relation to electromagnetic fields. It is often called the fourth state of matter, though taking into account its abundance in the universe, 99% of all *visible* matter is in the plasma state [1], one could rank it as the first state of matter. Plasmas can be found in many shapes, sizes, and forms. The most common plasmas found in nature are the stars and galaxies. On earth, a well known manifestation of the plasma state is lightning. But plasmas are also artificially created for a plethora of applications. These applications range from use in every day life, such as lighting [2, 3, 4], to industrial processes, such as welding and cutting, and the processes of etching, deposition, and sputtering involved in the fabrication of semiconductors [5].

In contrast to solids and liquids, plasmas often display a high degree of non-equilibrium. This means that *transport* aspects play an important role. A plasma is affected by its environment, be it the shape and size of the vessel it is created in, or the way power coupling into the plasma is achieved. These characteristics are the *configuration* aspects. The nature of plasmas enables the effective creation and destruction of species; i.e., *chemistry*.

Plasma aspects can thus be divided into three categories: transport, configuration, and chemistry. These three categories are, however, not distinctly separated, but are strongly interwoven. For instance, chemistry causes gradients in species density to occur, giving rise to transport.

Chemistry in plasmas is very rich in nature, involving many species between which many reactions can take place. In this thesis we use chemistry in its broadest sense; i.e., including excitation and de-excitation, and ionization and recombi-

nation. Consequently, the term *species* is also used in its broadest sense: a group of particles with unique chemical properties. Any excited level of an atom or molecule can therefore also be regarded as a separate species.

Chemistry is one of the most important aspects of plasmas, and certainly the wide variety offered by plasma chemistry is the reason why it is used in so many applications. Thus it is desirable to study chemistry aspects themselves, independently of transport and configuration aspects.

An important feature of plasmas is the relation with electromagnetic waves; more specifically light. Some plasmas are created with the intention to produce light (fluorescent lamp), some are created by light (laser produced plasmas), but all plasmas interact in some way with light. This has been vital to the field of astrophysics, were the light produced by plasma (stars and nebulae) is studied to gain insight in astrophysical phenomena. These studies started out with the investigation of atomic H and He plasmas; the main constituents of the universe. The light reveals the Atomic State Distribution Function (ASDF), which is formed by Electron Excitation Kinetics (EEK). In these investigations Collisional Radiative Models (CRM) have widely been applied.

Nowadays, plasmas are studied with a far more rich composition of species, in which also molecules play an important role, for instance in the fields of biochemistry (plasma wound healing [6]), and environmental sciences (gas cleaning [7]). Apart from the electrons facilitating processes between species as in simple atomic plasmas, in these plasmas Heavy particles (HEK) (and Mixed, MEK) also take on the role of agents in excitation kinetics. To focus on chemistry, zero-dimensional models have been developed in which transport and configuration aspects are "collapsed"; i.e. reduced to frequencies. These are so-called Global Plasma Models (GPM).

1.1 Thesis outline

The goal of this thesis is to document the design of zero-dimensional models focusing on plasma chemistry, and to demonstrate the use of these models. The thesis is therefore divided into two parts: theory (chapters 2 through 6) and application (chapters 7 through 9). In chapter 2 a distinction between three different forms of zero-dimensional models for plasmas is made, describing the main characteristics of each. These models are CRMs, GPMs, and Reaction Exploration Models (REM).

CRMs are the subject of chapters 3, 4, and 5. In chapter 3 the tasks of a CRM are described, and a general mathematical description for these tasks is derived. This description will be given in the form of simple matrix-vector notation. How the elements of these matrices and vectors are formed from a general description of an atomic system is the subject of chapter 4. In chapter 5 a description is given of the implementation of the general CRM in the plasimo framework, a flexible, multi-

purpose plasma modeling platform written in the object oriented programming language C++.

The plasimo framework is also used for the implementation of a general GPM code, the design of which is the subject of chapter 6.

With these two zero-dimensional model codes three types of plasmas are modeled. In chapter 7 the CRM code is used to model the radiation from Ar plasmas that are created by Extreme UltraViolet (EUV) radiation. The goal is to verify whether time resolved spectral lines can give information about the Electron Energy Distribution Function (EEDF) and its evolution in time.

Another application of the CRM code is given in chapter 8, where it is used to model Laser Induced Fluorescence (LIF) experiments in Ar plasmas. The CRM allows to study the response of excited level densities, as a result of a disturbance of the ASDF imposed by a laser pulse.

The third application is a case study of High Power Impulse Magnetron Sputtering (HiPIMS) Ar plasmas in chapter 9. In this chapter the GPM code is used to investigate the ionization degree of metal species in plasmas, that are introduced by sputtering. Furthermore, the model enables the analysis of reaction paths, so that the dominant mechanism for the creation of the metal ions is revealed.

We will end in chapter 10 with some general conclusions.

Part I

Theoretical framework and implementation

Chapter 2

Particle balances: Species and rate domain

2.1 Introduction

In this chapter we will explore different types of models for plasma chemistry. These models describe the densities of particles within a plasma as a result of processes between these particles. Plasmas have many distinct features, which can be grouped into three main blocks:

- *Configuration* describes the interaction of the plasma with the environment. This includes aspects such as *geometry*, *boundary conditions*, and *energy coupling*.
- *Transport* describes the transport of *species*, *momentum*, and *energy* within the plasma and with its direct surroundings, resulting from sources and sinks.
- *Chemistry* describes the creation and destruction of species in the plasma.

These three blocks, however, are not strictly distinct but strongly interwoven. The main subject of this thesis is chemistry. It is evident that the mechanisms of species creation and destruction described by chemistry form the sources and sinks for the transport aspects of the plasma. Furthermore, as the composition determines the electrical conductivity, which is an important factor in the energy coupling, the chemistry aspects are also closely linked with the configuration.

As the processes in plasmas result in countless numbers of distinct species, it is desirable to restrict the number of species to a more practical size. However, the restricted number of species must still properly describe the chemistry

and the plasma features determined by the chemistry. It is seductive to simply neglect the species with the lowest densities, and to only take under consideration the Principal Density Reservoirs (PDR). However, species with low densities might provide fast transition routes between species with high densities. When low density species are removed, it is important to account for these intermediate processes in the form of Effective Conversion Coefficients (ECC) between PDRs.

In plasma chemistry the following three issues are important:

1. Determine the minimum set of relevant species that has to be taken into account, in order to properly describe the distribution of mass over the species.

2. Removal of species with small densities must not affect other plasma aspects. This might require introduction of effective conversion coefficients between PDRs.

3. Construct flexible zero-dimensional models that describe the essence of the chemistry in plasmas, and that are detached from configuration and transport aspects. This means that wall processes have to be accounted for in the form of rates of species appearing in or disappearing from the plasma.

In literature various types of models have been described that deal with these issues. They can be classified in three groups:

- Collisional Radiative Models (CRM).
- Reaction Exploration Models (REM).
- Global Plasma Models (GPM).

Collisional Radiative Models

These models originate from astrophysics and plasma spectroscopy and were, among others, initiated by the classical paper of Bates et al. [8]. Usually in these models the atomic state distribution function (ASDF) can be seen as being composed by two contributions: one from the atom ground state, the other from the ion ground state. These two ground states are the PDRs, and they are linked together by electron collisions and radiative transitions, most of which involve the intermediate, excited levels. The electrons and photons are treated as external agents that facilitate the transitions between the levels, making the system linear*. The number of PDRs in an atomic system can be increased. An example of this is demonstrated by Van Dijk et al. [9] in order to treat the generation and transport of resonance radiation in low pressure Hg lamps.

* The CRMs in this work (chapters 7 and 8) will not include non-linear processes, though a description is given of how their inclusion can be realized.

CRMs form a major part of this work; in the next chapter an in-depth treatment will be presented, followed by the description of a general implementation using the PLASIMO plasma modeling framework in chapters 4 and 5. Furthermore, two specific CRMs will be presented in chapters 7 and 8.

Reaction Exploration Models

The purpose of these models, mainly developed in the field of plasma chemistry, is to study reaction kinetics in especially molecular plasmas. Available models in this category are ZDplaskin [10] and the commercial package Quantemol-P [11].

In contrast to CRMs, the kinetics are not restricted to a external agents, but all species can initiate reactions. An example of this is the energy exchange between two vibrationally excited N molecules, taken from Guerra et al. [12]:

$$N_2(X, v) + N_2(X, w) \rightarrow N_2(X, v-1) + N_2(X, w+1). \tag{2.1}$$

Because the densities of particles that also facilitate transitions need to be determined, the system is non-linear.

Chapter 6 presents *ZDM*, a zero-dimensional model plug-in for the PLASIMO framework. This model plug-in is the continuation of a model by Jiménez [13] called *RateLab*, which in turn is based on the model *PyRate* by Van den Donker [14]. The aim of *PyRate* was to give insight into the oxygen chemistry for plasmas with a given elemental oxygen concentration, electron density, and electron temperature. A set of differential equations, one for each species, is solved in time, giving the species densities as a function of time. Analysis of the results reveals the PDRs and the effective conversion routes between them. The *RateLab* model extended the functionality of *PyRate* by including the electron density and energy balance in the computation. Furthermore, the flexibility was increased through the use of input files, and the possibility for a time dependent power input into the plasma. The ZDM module has the same basic functionality as *RateLab*, but the usability is greatly increased by the graphical user interface of PLASIMO. Models can easily be manipulated, and results can be analyzed while the computation is running.

Global Plasma Models

These models are often used in the field of plasma engineering. Their goal is to predict mean values of internal plasma parameters, such as electron density and temperature, as a function of external control settings. These control settings, such as pressure, geometry, and input power, can be seen like a set of control knobs, much like controlling an industrial setup.

In literature many different GPMs can be found, see for instance [15, 16, 17]. Also the PLASIMO team has experience in this field, for instance the model by Broks et al. presented in [18]. This is a model for Ar plasmas based on the electron

density and electron energy balance combined with the heavy particle energy balance. Only essential species, the atom and ion ground state, are taken into account, making the model lean and flexible. Since intermediate levels are not considered, effective conversion rates are required, which are available from a CRM of Ar. This demonstrates the fact that a GPM can not stand on its own. The results of other models are required, in this case a CRM.

The distinction between CRM, REM, and GPM is thus as follows:

- CRM and REM are mainly methods to describe the chemical interactions in a system of species, whereas a GPM models plasma parameters controlled by a limited set of external parameters, most importantly the (time dependent) power input. A GPM can use either the CRM or REM method to achieve a solution.

- The distinction between CRM and REM lies in the difference in agents that drive the reactions. While in a CRM they are external resulting in a linear system, in an REM the agents can be any species included in the model, so the system is non-linear.

This chapter serves as an introduction to chapters 3, 4, and 5 that treat CRMs and chapter 6, where a GPM with CRM and REM characteristics is presented. The goal is to narrow down the aspects of chemistry and the interaction with configuration and transport under consideration. Furthermore, two approaches to tackling the system of coupled balances are described, one used for CRMs and one for GPMs. The models mentioned so far are all zero dimensional, so the geometrical variances are somehow averaged. The interaction with the surroundings is in general heavily dependent on the specific geometry which means that extra care has to be taken to incorporate these processes in a model. The last section therefore deals with wall reactions.

2.2 Analysis

The density of a species in a plasma will behave according to the balance, describing the fact that the net production gives lead to accumulation and efflux, or simply put:

$$\text{Net Production} = \text{Accumulation} + \text{Efflux}. \tag{2.2}$$

The "efflux" in this equation results from the configuration and transport aspects whereas "Net production" is the result of chemistry. This balance can be written in a more formal way using the zeroth moment of the Boltzmann transport equation:

$$\mathcal{S}_s = \frac{\partial n_s}{\partial t} + \nabla \cdot \mathbf{\Gamma}_s, \tag{2.3}$$

where n_s, $\mathbf{\Gamma}_s$, and $\mathcal{S}_s$ are the density, flux density, and source of species s, respectively. The chemical source term itself can be split into production and destruction:

$$\mathcal{S}_s = \mathcal{P}_s - \mathcal{D}_s n_s, \tag{2.4}$$

where $\mathcal{D}_s$, the destruction, is a frequency. As we will see, this frequency can be used to make a classification of species.

To simplify the analysis we can write the efflux term using a transport frequency:

$$\mathcal{F}_s = \frac{1}{n_s} \nabla \cdot (n_s \mathbf{v}_s). \tag{2.5}$$

If the transport mechanism is diffusive, it can also be written as:

$$\mathcal{F}_s = \frac{D_s}{\Lambda^2}, \tag{2.6}$$

demonstrating the combination of the transport and configuration aspects. The transport aspects are represented by the diffusion coefficient D_s. The configuration aspects are represented by the characteristic diffusion length Λ, which is mainly determined by the shortest plasma size.

The diffusion coefficient for neutral (excited) species is [19]:

$$D_s = \frac{1}{3} \lambda v_{\text{th}}. \tag{2.7}$$

With the thermal velocity $v_{\text{th}} = \sqrt{8 k_B T_h / (\pi M_s)}$ and the mean free path $\lambda = 1/(\sigma n \sqrt{2})$ this becomes:

$$D_s = \frac{2}{3 n \sigma} \sqrt{\frac{k_B T_h}{\pi M_s}}, \tag{2.8}$$

where n is the buffer gas density, σ the cross section for elastic collisions, and M_s the species mass.

For charged species the electric field comes into play, which adds a drift component. When quasi neutrality is assumed, the extra drift leads to ambipolar diffusion, so the diffusion is enhanced:

$$D_a = D_s \left(1 + \frac{T_e}{T_h} \right). \tag{2.9}$$

Whereas the frequencies for transport of species typically range up to 10^4 Hz, destruction frequencies can be much higher, up to 10^8 Hz. In most cases the destruction frequencies of the excited species are much higher than their transport frequencies, while for the atom and ion ground states the destruction and transport frequencies are in a comparable range. This enables us to categorize

the species, which we will show in more detail using the general balance equation (2.3):

$$\frac{\partial n_s}{\partial t} + \mathcal{F}_s n_s = \mathcal{P}_s - \mathcal{D}_s n_s, \tag{2.10}$$

with the substitutions mentioned in (2.4) and (2.5). If we assume $\mathcal{F}_s$, $\mathcal{P}_s$, and $\mathcal{D}_s$ to be constant in time, the density of the species is described by:

$$n_s(t) = n_s(0)\, e^{-(\mathcal{F}_s+\mathcal{D}_s)t} + \frac{\mathcal{P}_s}{\mathcal{F}_s + \mathcal{D}_s}\left(1 - e^{-(\mathcal{F}_s+\mathcal{D}_s)t}\right), \tag{2.11}$$

with $n_s(0)$ some initial density. This solution immediately shows that on a timescale $(\mathcal{F}_s + \mathcal{D}_s)^{-1}$ the density of species s converges to $\mathcal{P}_s/(\mathcal{F}_s + \mathcal{D}_s)$. Furthermore, when the destruction frequency is much higher than the transport frequency, or $\mathcal{D}_s \gg \mathcal{F}_s$, the density converges to:

$$n_s = \frac{\mathcal{P}_s}{\mathcal{D}_s}, \tag{2.12}$$

on a timescale $\mathcal{D}_s^{-1}$.

The derivation shows that species can be categorized depending on their destruction frequency compared to their transport frequency: species with high destruction frequency are named *Local Chemistry* (LC) species, the other species are *Transport Sensitive* (TS) species. In general, LC species will not have a high density because of their high $\mathcal{D}_s$ values. Their density is mainly determined by TS species that form the principal density reservoirs. This distinction between LC and TS levels is comparable to the QSSS procedure as described by Bates et al. [8].

The assumption that leads to the simple solution for the species density (2.11) is, while being instructive, not very realistic. The different frequencies can be time dependent and can depend on the densities of other species. This means that the balance equations of all species form a set of coupled equations that have to be solved simultaneously. If the destruction frequency of a species also depends on the density of the species itself, the problem is also non-linear.

2.3 Balance domains

We can combine the balance equations for all species under consideration in a vector equation:

$$\frac{\partial \mathbf{n}}{\partial t} + \boldsymbol{T}\mathbf{n}(t) = \mathbf{S}(\mathbf{n}(t)), \tag{2.13}$$

where $\mathbf{n}(t)$ is a vector of species densities, $\boldsymbol{T}$ a matrix of transport frequencies (on the diagonal), and $\mathbf{S}(\mathbf{n}(t))$ the source vector. The source vector depends on the densities of other species and can be constructed in two domains:

- *Species Density Domain* (SDD), where the source vector is formed by multiplying a frequency matrix with the density vector, or
- *Reaction Rate Domain* (RRD), where the source vector is formed by multiplying a stoichiometry matrix with the reaction rate vector.

2.3.1 Species density domain

In the species density domain the source vector is formed by:

$$\mathbf{S} = \boldsymbol{F}\mathbf{n}, \tag{2.14}$$

where the matrix $\boldsymbol{F}$ is a combination of the production and destruction frequencies (2.4), with destruction frequencies populating the diagonal of the matrix. This form is commonly used in CRMs, and very convenient in case the balance equations are linear, i.e. when the destruction frequency of a species is independent of its own density.

In chapter 3 this method of forming the source vector will be treated in great detail. The frequency matrix $\boldsymbol{F}$ will describe the electron collisional processes and radiative processes between species. The electrons {e} and photons {f} are considered the *agents* facilitating transitions between species. The frequency matrix is therefore dependent on the electrons through the electron density and electron energy distribution function (EEDF). The photons influence the frequency matrix via the radiation field. This is used in chapter 8, where we present a CRM for Laser Induced Fluorescence (LIF) experiments. The electron density, EEDF, and radiation field can also be time dependent.

The frequency matrix describes the relations between levels, provided by the agents electrons and photons. In chapter 3 the three main tasks of a CRM will be treated, that can be derived from this matrix. They are to obtain:

1. the Atomic State Distribution Function (ASDF);
2. the effective conversion rates between TS levels;
3. the energy transport between the agents and the chemistry.

For these three properties simple expressions will be derived in chapter 3. In chapter 5 a code based on the plasimo framework will be presented to construct these kind of models.

2.3.2 Reaction rate domain

The calculation of the source term in the reaction rate domain is best explained by a simple example. For the following reaction with rate coefficient k_{AB} dependent

on the temperature T (note species B on both sides):

$$\alpha A + \beta B \overset{k_{AB}(T)}{\rightarrow} \gamma C + \delta B, \tag{2.15}$$

the rate R_{AB} (in $\mathrm{m^{-3}\,s^{-1}}$) is given by:

$$R_{AB} = n_A^{\alpha} n_B^{\beta} k_{AB}(T), \tag{2.16}$$

where n_A and n_B are the densities of species A and B. For the three involved species A, B, and C the source terms created by this particular reaction are $-\alpha R$, $(\delta - \beta)R$, and γR respectively, since the net result is required.

In general we can write a reaction r as:

$$\sum_s \nu_{s,r}^d X_s^d \rightarrow \sum_s \nu_{s,r}^p X_s^p, \tag{2.17}$$

where the sum runs over all species X_s, and $\nu_{s,r}^d$ and $\nu_{s,r}^p$ are the stoichiometric coefficients for destruction and production, respectively. When a species is not involved the stoichiometric coefficient is simply zero. For this general reaction the rate is:

$$R_r(\mathbf{n}, T) = k(T) \prod_s n_s^{\nu_{s,r}^d}. \tag{2.18}$$

If a species is not involved in the reaction, it can still be included in the product sequence, because $n_s^{\nu_{s,r}^d} = 1$ when $\nu_{s,r}^d = 0$. This is especially convenient from an implementation point of view.

Although the *rate* is always expressed in $\mathrm{m^{-3}\,s^{-1}}$ the unit of the *rate coefficient* is dependent on the number of involved species and their stoichiometric coefficients; i.e. expressed in units $\mathrm{m}^{3N_r-1}/\mathrm{s}$, with $N_r = \sum_s \nu_{s,r}^d$.

To determine the actual source term of a species for a reaction $S_{s,r}$, the net stoichiometric coefficient $W_{s,r}$ must be multiplied by the rate, as already shown for the example in equation (2.15). This means that the left-hand side stoichiometric coefficient is subtracted from the right-hand side coefficient, or $W_{s,r} = \nu_{s,r}^p - \nu_{s,r}^d$. The total source for a species s involved in reactions r is therefore:

$$S_s = \sum_r S_{s,r} = \sum_r W_{s,r} R_r. \tag{2.19}$$

The source vector in equation (2.13) can now be written using a convenient matrix-vector multiplication:

$$\mathbf{S} = \boldsymbol{W}\mathbf{R}, \tag{2.20}$$

where $\mathbf{R}$ is a vector of rates (one per reaction), and $\boldsymbol{W}$ the stoichiometry matrix. The rows of $\boldsymbol{W}$ refer to the species and the columns to the reaction rates, so each matrix element represents the "weight" a reaction has on a species. In other words $\boldsymbol{W}$ projects the rate space onto the species density space.

This representation of the reactions does not inherently include backward processes. To include those the backward processes must be defined explicitly.

In chapter 6 an implementation of a global model code using the plasimo framework is presented. The code is designed to create global models in which the *particle* balances as well as the *energy* balance for electrons are included. The system, including the balance equations for all other involved species, is solved simultaneously as a function of time. In chapter 9 a specific model implemented in this code for High Power Impulse Magnetron Sputtering (HiPIMS) plasmas is presented.

2.4 Wall reactions

So far we have made an analysis of how the chemistry in a plasma affects the particle balances and how this can be formalized. Transport was simply included in the balance equation in the form of a transport matrix consisting of transport frequencies on the diagonal, see equation (2.13). This is because the models treated in this thesis are zero dimensional; i.e., the configuration and transport aspects are combined and "translated" into a frequency.

Representing the complex transport processes of a species in a plasma with a certain configuration by only a single number is by all means not trivial. Whereas in the bulk of the plasma these aspects can be reasonably described using earlier mentioned (ambipolar) diffusion coefficients, interactions with the wall are quite different. Due to the presence of a (pre)sheath the dynamics can be quite different from the rest of the plasma. Because the wall can play an important role in many reactions, great care must be taken to incorporate its effects in the transport frequencies.

Near a wall, a plasma will form a layer that has an excess positive charge, called the sheath. It can be shown (see e.g. [20]) that in order for the layer to be stable, the so-called *Bohm criterion* must be fulfilled, stating that the velocity of ions entering the sheath, v_0, must be at least the Bohm velocity:

$$v_0 \geq v_{\text{Bohm}} = \sqrt{k_B T_e / M_i}, \tag{2.21}$$

where e is the elementary charge, and M_i the ion mass. The directed velocity v_0 dictates the presence of a finite electric field in the plasma over some region called the *presheath*.

Analytic descriptions of the species densities and potential in the sheath and presheath are only available for certain geometries and under certain assumptions. In general numerical techniques are required.

As an example we give the solution for the plasma density at the radial sheath

edge of a cylindrically shaped plasma, attributed to Godyak et al. [21]:

$$\frac{n_{\text{sheath}}}{n_{\text{bulk}}} \approx 0.80 \left(4.0 + \frac{R}{\lambda_i}\right)^{-1/2}, \tag{2.22}$$

where R is the radius of the plasma and λ_i the ion-neutral mean free path. The sheath density determined by this equation multiplied with the Bohm velocity gives the flux of particles leaving the plasma. In chapter 9 this will be used in a GPM of a HiPIMS plasma.

Equation (2.22) demonstrates how configuration aspects affect the transport aspects of a plasma. The combination of these two aspects are then "summarized" into a single frequency acting upon the chemistry.

2.5 Conclusion

Zero-dimensional models in which transport and configuration aspects are present in simplified form, can be employed to study the chemistry of plasmas. We have distinguished three main types of these models, each developed in its own discipline and aimed at different objectives. Chapters 3, 4, and 5 will focus on atomic CRMs, while in chapter 6 the implementation of a GPM is presented that possesses REM characteristics.

Chapter 3

The tasks of collisional radiative models

3.1 Introduction

In the previous chapter a short introduction was given for Collisional Radiative Models (CRMs). These models aim to describe *chemistry* aspects of plasmas in contrast to *transport* and *configuration*. We will restrict ourselves to atomic systems where the chemistry is formed by reactions involving the ground and excited states. These reactions are electron collisional (de-)excitation and radiative transitions in which the electrons and photons are the *agents* effectuating these processes. We will make a distinction between levels based on the time scales at which chemical processes act on the levels. Depending on the plasma conditions, for most of the excited species the chemical processes will be much faster than transport, so that local production will equate local destruction. These levels are called *Local Chemistry* (LC) levels. The LC levels generally have a much lower density than the atom and ion ground state, that are the Principal Density Reservoirs (PDRs). Since the PDRs are not only ruled by chemistry, but also by transport*, they are called *Transport Sensitive* (TS).

By this classification of levels, the system of excited levels, known as the *excitation space* has at least two *entries*: one at the low energy side (atomic ground level) and one at the high energy side (ion ground level). In between, the excitation space is filled by the LC levels, or in this context the *internal* levels. The

* Transport is meant in a general sense, i.e., any process that is not part of the system itself and that acts as a source term on a level.

occupation of the internal (LC) levels is determined by the entry (TS) levels and the excitation kinetics, which for many atomic plasmas is ruled by the electron gas. Thus we see that through Electron Excitation Kinetics (EEK) the densities of the LC levels are directly linked to those of the TS levels. Simultaneously, the LC levels facilitate traffic between TS levels through the excitation space. Apart from direct processes between TS levels, stepwise processes are possible involving the LC level. Between the atom and ion ground state, stepwise ionization and recombination enhance the relation between the entry points, depending on the agent (the electron gas). This method of distinguishing between TS and LC levels is familiar to the method introduced in the past as the Quasi Steady State Solution (QSSS) by Bates et al. [8].

The system of TS and LC levels and their relations is what the CRM describes. Whereas in the past CRMs were mostly used as a tool for the interpretation of spectroscopically determined atomic state distribution functions (ASDF), they are nowadays also of great importance in the description of the competition between chemistry and transport. As such they are a vital component for the formation of Grand Plasma Models. The tasks of a CRM are thus threefold: to provide *ingredients* for the calculation of

1. the ASDF,

2. the rate coefficients for effective conversion between TS levels,

3. and the source terms for the energy equation.

In order to construct a CRM one must first decide how many species or atomic levels have to be taken into account. A reduction of this number can be achieved using a *cut-off* procedure, see for instance Van der Mullen [22], which is based on the known analytical dependency of the ASDF of highly excited levels as a function of ionization potential. The second step is to determine which levels are classified as TS and LC. Subsequently the QSS solution can be employed, which enables the expression of the three tasks solely in terms of the densities of the TS levels.

This chapter will give an outline of the general structure of CRMs for atomic plasmas. The emphasis will lie on atomic plasmas ruled by EEK. The description of LC levels in terms of TS levels will be given in matrix form. Following this description, the three tasks will also be cast in the form of simple matrix expressions, replacing the tedious (double) summations often found in literature.

The general description of the structure of a CRM for atomic plasmas ruled by EEK, enables a proper description of the influence of radiation transport and Heavy particle Excitation Kinetics (HEK). The effect of formation and destruction of molecular ions is that extra sources and sinks are generated in the atomic excitation space. The level at that location is promoted from an internal level (LC) to

an entry point (TS). Radiation transport is dealt with in a similar manner, i.e. by promoting a radiating level from LC to TS.

This chapter will continue with a general exploration of a few simple systems. In this exploration we will introduce the notation that is used and work towards simple expressions for the dynamics in these systems. Subsequently, in section 3.3, we will focus onto the different processes within an atomic system, and onto how the balance equations can be cast into a matrix-vector notation. In section 3.4 two aspects of dealing with the set of balance equations are treated: the quasi steady state solution, and the cut-off technique. The general rules from section 3.2 are cast into convenient matrix-vector notation describing the three CRM tasks in section 3.5, and in section 3.6 they are generalized to more complex systems. Accommodation of external processes into the matrix notation is treated in section 3.7. In section 3.8 we outline a method to deal with a system subjected to radiation transport, demonstrating the promotion of an LC level to a TS level. In the last section (3.9) some remarks about time dependent CRMs are given.

3.2 General exploration

We will start with a global exploration based on a case study of optically thin plasmas. Initially, the most basic system consisting of two TS levels is treated. Subsequently, the extension to a three level system will be described.

3.2.1 Two level system

Consider a plasma in *steady state* that consists of one atomic species with the corresponding ion of charge number $Z = 1$. This ion only exists in the ground level. The system of excited levels between the atomic and ionic ground level is denoted by the *excitation space* or *the system*, in short. We assume that only the ground levels of the atom and ion are transport sensitive (TS), cf. Fig. 3.1 and 3.2.

Ionizing system

Figure 3.1 gives a representation of a purely ionizing atomic system [23] for which the density at the high energy side, the ζ-side* , is set equal to zero: $n(\zeta) = 0$. This can be reached, or better, approached, if the efflux of ion-electrons pairs is extremely large; for instance due to fast diffusion created by large density gradients. This efflux at the high-energy side demands (in steady state) for an influx at the α-side and must be supported by an (net) ionization flux in the excitation space leading to the $\alpha \rightarrow \zeta$ conversion. An important observation is that the transport

* To be as general as possible we use the symbol ζ for the high energy output side. In case of an atomic system it represents the ion ground level, so $\zeta = +$. The opposite side, the low energy input side, is denoted by α.

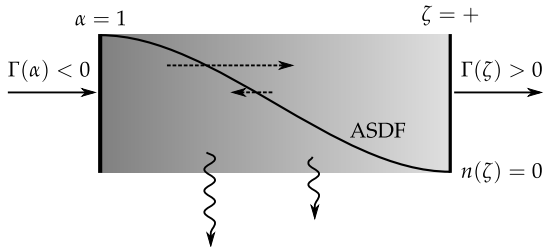


Figure 3.1. *A sketch of an ionizing atomic system with two entries: the atom ground level α, and ion ground level ζ. Only these levels (α, ζ) are transport sensitive (TS). We assume to deal with a purely ionizing situation so that, due to the large efflux of ion-electron pairs, the density at the ζ-side equals zero: $n(\zeta) = 0$. This efflux demands (in steady state) for an influx at the α-side and a flux in the excitation space that leads to the $\alpha \rightarrow \zeta$ conversion. This conversion is accompanied by the generation of radiation. Both conversion and radiation are ruled by the electron excitation kinetics, EEK. As illustrated by the dashed arrows, the EEK conversion is the net result of several excitation and de-excitation processes. The shape of the ASDF is determined by the density at the α entry level and the EEK, and because of the linear nature of the system, we expect a relation between the ground level α and an arbitrary internal level of the form $n^{\alpha}(i) = R_{i\alpha} n(\alpha)$. It can be understood that the radiative energy losses are directly proportional to the ground state density (cf. Eqn.* (3.3)*).*

efflux Γ (in $\mathrm{m^{-3}\,s^{-1}}$) at the input must (in absolute value) be the same as that of the output side, so that:

$$\Gamma(\alpha) = -\Gamma(\zeta) \qquad \text{or} \qquad \Gamma(\alpha) + \Gamma(\zeta) = 0. \tag{3.1}$$

We will use the sign-convention that Γ is reckoned positive for the case of outward transport so that, in this case, $\Gamma(\zeta) > 0$ whereas $\Gamma(\alpha) < 0$, see Fig. 3.1. This relation (Eqn. (3.1)) will be denoted by the *system flux balance*. In this example of a purely ionizing atomic plasma the ASDF is determined by the density at the entry level α and the electron excitation kinetics, EEK. Due to the linear nature of the system, we can expect a relation between the ground level α and an arbitrary internal level of the form:

$$n^{\alpha}(i) = R_{i\alpha}\, n(\alpha). \tag{3.2}$$

This "relation function" R is the result of various forward and backward processes and heavily depends on the properties of the electron gas $\{e\}$: the density n_e and temperature T_e. Furthermore, due to the fact that the population of all radiating levels is directly proportional to that of the entry level α, we may assume that also the radiation output is directly proportional to $n(\alpha)$. Thus, the power density ($\mathrm{W\,m^{-3}}$) at which photons are created can be written as*:

$$\varepsilon_f^{+} = L^{\alpha}\, n(\alpha), \tag{3.3}$$

* The following notation convention is used: ε stands for the power density ($\mathrm{W\,m^{-3}}$), the subscript e, f, and c refer to the electron $\{e\}$, photon $\{f\}$, and chemistry $\{c\}$ energy reservoirs, respectively.

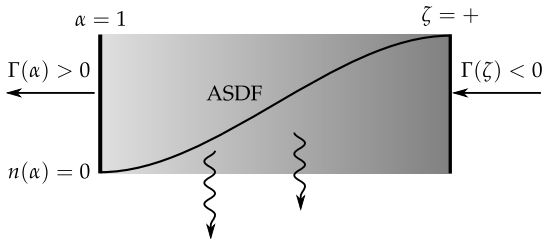


Figure 3.2. *A sketch of a recombining atomic system with the same entry levels as those given in Fig. 3.1. In this purely recombining case we have $n(\alpha) = 0$. The ASDF is now determined by the density at the ζ entry and the EEK. We expect an ASDF of the form $n^{\zeta}(i) = R_{i\zeta}n(\zeta)$.*

where L^{α} is called the *specific effective emissivity* cf. Van Dijk et al. [24]. The $+$ sign added as a superscript indicates that it is a gain term for the photon field $\{f\}$. Apart from the radiation also the conversion will cost energy since the internal energy of the ζ particles is larger than that of the α particles. Using the sign convention given above it is obvious that the power density at which "chemistry" is added to the plasma reads:

$$\varepsilon_c^+ = E(\alpha)\Gamma(\alpha) + E(\zeta)\Gamma(\zeta), \tag{3.4}$$

where $E(\alpha)$ and $E(\zeta)$ are the internal energies of the atom (ion) in level α and ζ.

Since both ε_c^+ and ε_f^+ originate from electron excitation we may assume that the power density ε_e^- at which the electrons $\{e\}$ *lose* energy must satisfy the following relation:

$$\varepsilon_e^- = \varepsilon_f^+ + \varepsilon_c^+ \qquad \text{or} \qquad \varepsilon_e^+ + \varepsilon_f^+ + \varepsilon_c^+ = 0 \tag{3.5}$$

Equation (3.5) treats the reservoirs on an equal footing and is similar to the Kirchhoff junction rule:

> *The sum of the currents in all the branches flowing from a junction equals zero.*

Here, current is replaced by power density, the branches by the reservoirs, while the role of the junction is played by the excitation space. Equation (3.5) will be denoted by the *system energy balance*.

Recombining system

Figure 3.2 gives a sketch of the situation opposite to that of Fig. 3.1, namely that of a purely recombining atomic system. The same two entry levels ($\alpha = 1$ and

The superscript $-$ or $+$ indicates whether the energy transfer is regarded as an effective loss ($-$) or gain ($+$) for the reservoir indicated by the subscript; so, evidently, $\varepsilon_x^+ = -\varepsilon_x^-$. If the sign is absent we assume that we deal with a gain term; i.e., $\varepsilon_x \equiv \varepsilon_x^+$.

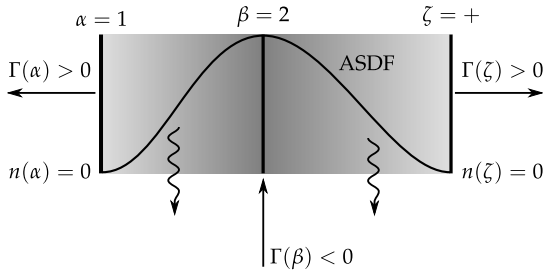


Figure 3.3. *A sketch of an atomic system in which level β, in this case $\beta = 2$ (the first excited level) is promoted from LC to TS level. It thus forms a new entry for the excitation space. The contribution of the population of β on the ASDF can be seen by setting the densities at the other entry levels equal to zero, which will result in $n^{\beta}(i) = R_{i\beta}n(\beta)$.*

$\zeta = +$) are involved but now ζ is the input and α the output side, which means that $\Gamma(\alpha) > 0$ and $\Gamma(\zeta) < 0$. In this situation the conservation laws in Eqns. (3.1) and (3.5) are also expected to be valid. In this purely recombining situation the ground state density is set to zero ($n(\alpha) = 0$) so that the ASDF and the radiation generation are determined by the density at the ζ entry (together with the EEK). Thus the ASDF and the line radiation will have the form:

$$n^{\zeta}(i) = R_{i\zeta}n(\zeta), \tag{3.6}$$

and

$$\varepsilon_f = L^{\zeta}n(\zeta). \tag{3.7}$$

3.2.2 Three level system

In Fig. 3.3 the first excited level $\beta = 2$, is promoted from LC to TS level and thus forms a new entry of the excitation space. The reasons for such a promotion can be that the timescale of ordinary transport (diffusion and convection) of β may be in the same order as that of the local chemistry. The influence of this extra source at the entry β on the ASDF is found by setting the densities at the other entries equal to zero. We expect for the ASDF and the radiation power density the following forms:

$$n^{\beta}(i) = R_{i\beta}\, n(\beta), \tag{3.8}$$

and

$$\varepsilon_f = L^{\beta}\, n(\beta). \tag{3.9}$$

The system flux balance, cf. Eqn. (3.1), can be generalized to:

$$\Gamma(\alpha) + \Gamma(\beta) + \Gamma(\zeta) = \sum_{\xi}\Gamma(\xi) = 0, \tag{3.10}$$

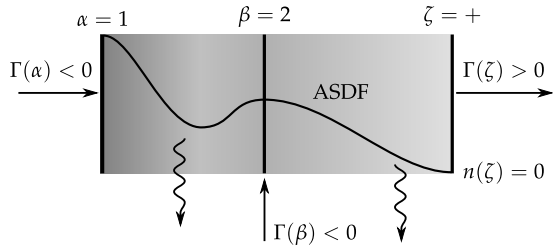


Figure 3.4. *A sketch of an atomic system in which the excitation space is fed by levels α and β.*

where the index ξ runs over the collection $\{\alpha, \beta, \zeta\}$. The associated power density will be of the form:

$$\varepsilon_c^+ = E(\alpha)\Gamma(\alpha) + E(\beta)\Gamma(\beta) + E(\zeta)\Gamma(\zeta) = \sum_{\xi} E(\xi)\Gamma(\xi). \tag{3.11}$$

Alternatively, we define the vectors $\mathbf{E}$ and $\mathbf{\Gamma}$ containing the energy values and efflux rates of the levels, so we can write the sum as a dot product:

$$\varepsilon_c^+ = \mathbf{E} \cdot \mathbf{\Gamma} = \mathbf{E}^t \cdot \mathbf{\Gamma}^t, \tag{3.12}$$

where ^{t} denotes TS levels, which is valid since for LC levels (denoted by ^{l}) $\Gamma^l = 0$. The system energy balance will retain the form given by (3.5).

3.2.3 Generalization

In Fig. 3.4 the three entry levels are simultaneously in action. The EEK makes the conversion between α, β, and ζ via the excitation space possible. In this general situation we may expect system energy and flux conservation laws of the same form as Eqns. (3.5) and (3.10). The power density associated to radiation will now be of the form:

$$\varepsilon_f^+ = L^\alpha n(\alpha) + L^\beta n(\beta) + L^\zeta n(\zeta) = \sum_{\xi} L^\xi n(\xi), \tag{3.13}$$

where we also introduce vectors to store the specific emissivities and densities: $\mathbf{L}$ and $\mathbf{n}$. We can then write ε_f as:

$$\varepsilon_f^+ = \mathbf{L}^t \cdot \mathbf{n}^t. \tag{3.14}$$

The density of an LC level i is given by:

$$n(i) = \sum_{\xi} n^\xi(i) = \sum_{\xi} R_{i\xi}\, n(\xi), \tag{3.15}$$

which can be written in a general matrix-vector form by:

$$\mathbf{n}^l = \boldsymbol{R}^{lt}\,\mathbf{n}^t, \tag{3.16}$$

where the densities of the LC levels (superscript l) are expressed in the densities of the TS levels (superscript t) by use of the matrix $\boldsymbol{R}^{lt}$. Thus the species (or levels) forming the system $\{s\}$ can be split up into the collections of entry (or TS) levels $\{t\}$ and the internal (or LC) levels $\{l\}$; i.e., $\{s\} = \{t\} \cup \{l\}$.

The factual form of the ASDF as described by the $\boldsymbol{R}$-matrix depends on the joint action of all the various transition processes. These are given by individual transition frequencies D defined such that $n(j)D(j,k)$ is the number of $j \to k$ transitions per unit of time.

3.2.4 Further generalization

If a CRM is used to determine the source of a plasma transport model we are not interested in all the details of the various processes. More important is to know the *effective* conversion between the entry levels, for which the internal levels serve as intermediates. Thus, what counts are the effective conversion coefficients. For example, $J(\alpha,\beta)$ is defined such that $n(\alpha)J(\alpha,\beta)$ is the number of effective processes per unit of time and volume by which α is converted into β. Effective means that apart from the direct processes from α to β, we also take the transport through the system (over the internal levels) into account. The *net effective conversion rate* from α to β is given by $n(\alpha)J(\alpha,\beta) - n(\beta)J(\beta,\alpha)$, where the adjective *net* refers to the result of *forward minus backward* reactions. The effective conversion coefficients can be stored in a matrix $\boldsymbol{J}$, which will be treated in section 3.6.

This concludes the general exploration of the tasks of an atomic CRM. We have found the general structure of the ASDF (3.16), the radiative power density associated with the transport of radiation (3.14), and found the structure of the energy and mass balance. We also took the opportunity to introduce the nomenclature and the matrix representation. However, the actual form of the ASDF, the effective conversion coefficients and source terms are still unknown. In the next sections we will see how these are composed out of the rate coefficients of the various elementary processes.

3.3 The system of coupled particle balance equations

The central role in the description of the relation between transport and elementary processes is played by the set of particle balance equations. Before we study the *set as a whole* we will first investigate the form of a *single* equation. It is the 0-th moment of the Boltzmann transport equation and expresses how the density $n(p)$ of a level p is determined by transport and various populating and depopulating

processes; it reads:

$$\underbrace{\frac{\partial}{\partial t} n(p)}_{\text{accumulation}} + \underbrace{\nabla \cdot \Big(n(p)\, \mathbf{v}(p) \Big)}_{\text{transport}} =$$

$$\underbrace{n_e\, n_1\, K(1,p)}_{\text{electron excitation from ground}} \overset{\mathcal{B}_1}{-} \underbrace{n(p)\, n_e\, K(p,1)}_{\text{electron de-excitation to ground}} \tag{3.17a}$$

$$+ \underbrace{\sum_q n_e\, n(q)\, K(q,p)}_{\text{electron (de-)excitation from } q} \overset{\mathcal{B}}{-} \underbrace{n(p)\, n_e \sum_q K(p,q)}_{\text{electron (de-)excitation to } q} \tag{3.17b}$$

$$+ \underbrace{n_e^2\, n_+\, K(+,p)}_{\text{two electron recombination}} \overset{\mathcal{S}}{-} \underbrace{n(p)\, n_e\, K(p,+)}_{\text{electron induced ionization}} \tag{3.17c}$$

$$+ \underbrace{\sum_l n(l) B(l,p) \rho_\nu}_{\text{absorption}} \overset{\mathcal{P}_l}{-} \underbrace{n(p) \sum_l [A(p,l) + B(p,l)\, \rho_\nu]}_{\text{emission (spont. + stim.)}} \tag{3.17d}$$

$$+ \underbrace{\sum_u n(u)\, [A(u,p) + B(u,p)\, \rho_\nu]}_{\text{cascade (spont. + stim.)}} \overset{\mathcal{P}_u}{-} \underbrace{n(p) \sum_u B(p,u)\, \rho_\nu}_{\text{photo excitation}} \tag{3.17e}$$

$$+ \underbrace{n_e\, n_+\, [K_{\text{rad}}(+,p) + B(+,p)\, \rho_\nu]}_{\text{radiative recombination (spont. + stim.)}} \overset{\mathcal{P}_+}{-} \underbrace{n(p) B(p,+)\, \rho_\nu}_{\text{photo ionization}}, \tag{3.17f}$$

where $\mathcal{B}$ refers to the Boltzmann balance, $\mathcal{S}$ the Saha balance, and $\mathcal{P}$ the Planck balance. Upper and lower levels are denoted by u and l respectively, while q refers to any other level ($q \neq p$), and 1 is the ground level. The symbol $K(p,q)$ represents the rate coefficient for the transition $p \rightarrow q$ induced by electron collisions. The symbols $B(p,u)$ and $B(u,p)$ are the coefficients for absorption and stimulated emission, respectively, while $A(u,l)$ refers to the transition probability for spontaneous decay ($u \rightarrow l$). The symbol ρ_ν represents the spectral energy density (in $\mathrm{J\,m^{-3}\,Hz^{-1}}$).

The structure of the particle balance in Eqn. (3.17) demonstrates that the accumulation, $\partial n(p)/\partial t$, and efflux, $\nabla \cdot (n(p)\mathbf{v}(p))$, of a species p is coupled to the various (collisional C and radiative R) production and destruction processes. These CR processes are grouped in forward and corresponding backward processes according to so-called proper balances [25] of the types Boltzmann ($\mathcal{B}$: excitation–de-excitation), Saha ($\mathcal{S}$: ionization–two electron recombination) or Planck ($\mathcal{P}$: absorption–emission). Disequilibrium of one of these balances (rhs) either leads to accumulation, will invoke transport (lhs), or demands for a complementary dis-

equilibrium of another balance. For instance the escape of radiation will lead to the disequilibrium of a balance of the Planck-type. This has to be compensated, for instance with the non-equilibrium state of a Boltzmann balance (more excitation than de-excitation) and/or the influx of excited species. A distinction was made between the Boltzmann balance $\mathcal{B}_1$ of p with the ground level ($p = 1$), Eqn. (3.17a), and those with other, higher levels, Eqn. (3.17b). We also distinguish between Planck balances to lower levels $\mathcal{P}_l$, higher levels $\mathcal{P}_u$ and the continuum $\mathcal{P}_+$.

Planck equilibrium is in most cases not easy to establish, in the sense that (line) emission can easily escape from the plasma and is not compensated by (re)absorption. Therefore it is useful to introduce the effective radiative transition probability A^* which is defined such that Eqn. (3.17d) is replaced by the *effective* decay:

$$\underbrace{n(l)\,B(l,p)\,\rho_\nu}_{\text{absorption}} - \underbrace{n(p)\,[A(p,l) + B(p,l)\,\rho_\nu]}_{\text{emission (spontaneous + stimulated}} \equiv \underbrace{-n(p)\,A^*(p,l)}_{\text{effective emission}}. \tag{3.18}$$

The same can be done for $\mathcal{P}_u$, Eqn. (3.17e), and $\mathcal{P}_+$, Eqn. (3.17f).

3.3.1 Reordering

The various terms in Eqn. (3.17) can also be ordered differently. For every level p we can define a transition frequency:

$$F(p,q) = n_e\,K(p,q) + A^*(p,q), \tag{3.19}$$

which describes the frequency at which level p is depopulated in favor of level q. It consists of two terms; one proportional to the electron density n_e, reflecting the electron induced transitions, and a constant term which refers to the (effective) radiative transition. This structure, which among other reasons is the consequence of the definition of A^* in Eqn. (3.18), is essential for EEK plasmas, in which electrons are the only agents ruling the traffic in excitation space.

When the ion level is the originating level ($p = +$), the transition frequency is defined by:

$$F(+,q) = n_e^2\,K(+,q) + n_e\,K_{\text{rad}}(+,q). \tag{3.20}$$

We can now combine production terms for level p:

$$P(p) = \sum_{q\neq p} n(q)\,F(q,p) + n_+\,F(+,p) \tag{3.21}$$

and the destruction factor:

$$D(p) \equiv F(p,p) \equiv \sum_{q\neq p} F(p,q) = n_e\,K(p) + A^*(p), \tag{3.22}$$

where we use $n_e\,K(p) \equiv n_e\,\sum_q K(p,q)$ and $A^*(p) \equiv \sum_l A^*(p,l)$.

Using these combining terms we get for Eqn. (3.17):

$$\underbrace{\frac{\partial}{\partial t}n(p)}_{\text{accumulation}} + \underbrace{\nabla\cdot\Big(n(p)\,\mathbf{v}(p)\Big)}_{\text{transport}} = \underbrace{P(p)}_{\text{production}} - \underbrace{n(p)\,D(p)}_{\text{destruction}}, \tag{3.23}$$

An even more abstract form for Eqn. (3.17), equating the transport to the (local) chemical source, is given by:

$$\underbrace{\mathcal{T}(p)}_{\text{transport}} = \underbrace{\mathcal{S}(p)}_{\text{source}}, \tag{3.24}$$

where evidently

$$\mathcal{T}(p) \equiv \frac{\partial}{\partial t}n(p) + \nabla\cdot\Big(n(p)\,\mathbf{v}(p)\Big) \equiv n(p)\,F_t(p), \tag{3.25}$$

and

$$\mathcal{S}(p) \equiv P(p) - n(p)\,D(p). \tag{3.26}$$

So the efflux and accumulation are represented as a transport frequency F_t, showing that accumulation can be viewed as transport in time and efflux as transport in space. Note that in contrast to Eqns. (3.25) and (3.26), Eqn. (3.24) is not a definition but an expression demonstrating the competition between transport and chemistry. The source term $\mathcal{S}(p)$ has to be computed using *local* chemistry (i.e. elementary processes) whereas the transport $\mathcal{T}(p)$ follows from (gradients in) the density and velocity field and the transient behavior of the plasma.

3.3.2 Matrix notation

It is clearly demonstrated in Eqn. (3.17) that the chemical production of a certain level p depends on the population of all other levels. Therefore a plasma model should contain a set of coupled balance equations in the form of Eqn. (3.17). The combined set can be cast in matrix form:

$$\mathbf{T} = \mathbf{S} \equiv \boldsymbol{F}\mathbf{n}, \tag{3.27}$$

where the chemistry is written as a matrix-vector multiplication. The elements of vector $\mathbf{T}$ are of the form given in Eqn. (3.25), one for each level, vector $\mathbf{n}$ is a vector composed of level densities, and matrix $\boldsymbol{F}$ a matrix containing the transition

frequencies defined as*†‡:

$$F_{pq} = F(q,p) \qquad \text{for } p \neq q \tag{3.28}$$

$$F_{pp} = -F(p,p) \equiv -\sum_{q \neq p} F(p,q) \qquad \text{on the diagonal} \tag{3.29}$$

Since the transition frequency is a summation of electron excitation kinetics (EEK) and radiative transitions (see Eqn. (3.19)), the frequency matrix $\boldsymbol{F}$ can be decomposed into a collisional and radiative part:

$$\boldsymbol{F} = {}^{e}\boldsymbol{F} + {}^{f}\boldsymbol{F}, \tag{3.30}$$

where ${}^{e}\boldsymbol{F}_{pq} = n_e\, K(q,p)$, and ${}^{f}\boldsymbol{F}_{pq} = A^*(q,p)$. We will now continue to discuss how this matrix equation can be simplified.

3.4 Simplification

The set of balance equations described by Eqn. (3.27), can be simplified in two ways. First, for many levels the chemistry is so fast compared to transport, that we can disregard the latter. This leads to a distinction between LC and TS levels. Second, we can restrict the number of levels that are taken into account, by the assumption that the occupation of highly excited levels can be described analytically.

3.4.1 Local chemistry versus transport sensitive

For many plasma conditions transport is relatively slow and in most cases the inequality $F_t(p) < 10^6\ \mathrm{s}^{-1}$ holds for all species. However, for excited species the elementary reactions are fast and the destruction factor is of the order $D(p) \approx 10^7\ \mathrm{s}^{-1}$ or higher. If we divide Eqn. (3.24) by the destruction frequency we get:

$$\frac{P(p)}{D(p)} - n(p) = \frac{n(p)\,F_t(p)}{D(p)} \approx 0 \qquad \text{or} \qquad n(p) \approx \frac{P(p)}{D(p)} \tag{3.31}$$

This leads to a procedure known as the Quasi Steady State Solution (QSSS) [8]. The densities of excited states, for which (3.31) is fulfilled, are only determined by local chemistry LC. This simplification does not hold for the ground states of

* According to mathematical convention, F_{qp} is the element of matrix $\boldsymbol{F}$ positioned on row q and column p. At that location we find the frequency at which p populates q, which, by CRM convention, is given by $F(p,q)$. That is why $F_{qp} = F(p,q)$.

† q and p run over all levels in the system including the ion level ("+").

‡ The minus sign in the definition of F_{pp} is related to that placed before the destruction factor in Eqns. (3.17) and (3.26). The summation in the definition of the so-called total destruction rate $F(p,p) = D(p)$ runs over all other levels $q \neq p$, cf. Eqn. (3.22).

the atom and ion. These species are called transport sensitive and can be seen as the entries to the excitation space. Therefore, they will be denoted by entry or TS levels, the other levels are *internal* or LC levels.

In the past many CRMs with a two-entry structure were constructed. In section 3.6 we will give a prescription of how the structure of a CRM can be extended to situations in which more than two TS levels are needed.

3.4.2 Cut-off procedure

A further simplification is obtained by reducing the number of LC-levels, which leads to a reduction of the dimension of the vector equation (3.27). This can be done using the *cut-off* method that is based on the scaling of the destruction frequencies as a function of the principal quantum number* p. The electron collisional excitation rate coefficients follow the principal quantum number scaling by [25]:

$$K(p) \propto \mathsf{p}_p^4 \, Z^{-3}, \tag{3.32}$$

and for transition probabilities the scaling is:

$$A(p) \propto \mathsf{p}_p^{-5} \, Z^4. \tag{3.33}$$

The approximations demonstrate that for increasing p_p we rapidly enter the domain of the excitation space where the collisional part dominates the radiative part in the total destruction term in transition frequency (3.19). This happens when $p > p_{\mathrm{CR}}$ where p_{CR} is the so-called CR-boundary given by the condition $n_e \, K(p_{\mathrm{CR}}) = A^*(p_{\mathrm{cr}})$. The CR-boundary is determined approximately by the expression:

$$p_{\mathrm{cr}}^9 \approx 9 \times 10^{23} \, \frac{1}{n_e} \, Z^7, \tag{3.34}$$

where Z is the charge number of the core, and the electron density n_e must be given in m^{-3}. This boundary condition shows that for increasing n_e the boundary level p_{CR} shifts towards lower values.

It can be shown that for a wide range of conditions both the ASDF and the flux in the collisional part of the excitation space (thus $p > p_{\mathrm{CR}}$) can be treated analytically (see section 4.3.5). These analytical expressions make it possible to construct a cut-off procedure so that the number of excited states that must be treated numerically can be reduced drastically. Thus in general, a CRM can be constructed with a *numerical* bottom and an *analytical* top and since only the numerical part has to be solved by Eqn. (3.27) we can reduce the dimension of this

* Note that we use the symbol p among others to label the levels and thus the equations of type (3.17). We use the notation p to refer to principal quantum numbers (pqn) defined as $\mathsf{p}_p = Z\sqrt{Ry/E_{p+}}$ where E_{p+} is the ionization potential of level p, Z the charge of the core, and $Ry \approx 13.6\,\mathrm{eV}$ the Rydberg energy.

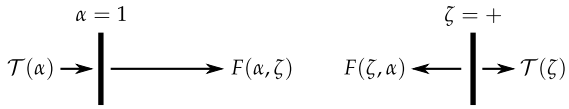


Figure 3.5. *The system of two levels being at the same time the two entries of the excitation space and their interacting transition frequencies.*

matrix-vector equation drastically. This cut-off technique will also be available in the CRM implementation described in chapter 4.

Finally we would like to mention that there are conditions for which the whole ASDF can be cast in an analytical form, thus for both the lower radiative ($p < p_{CR}$) and the upper collisional domain ($p > p_{CR}$). This pure analytical expression was found to give a fairly good description of the ionization flow through the chain of ionic systems, ranging from, I ($Z = 1$) to X ($Z = 10$) of strongly ionizing Xe and Sn pinch plasmas as used for lithography [26]. However, to be as general as possible and to be prepared for the description of the influence of molecular processes and radiative transfer on atomic excitation spaces we will study the general structure of numerical CR Models.

3.5 Structure and tasks of a CRM

This section gives a systematic description of the structure and tasks of a numerical CRM. It follows a step-by-step approach starting with the simplest system, a system of 2 levels that are both entries of an atomic excitation space. Conversion between these two can only take place by *direct* processes. This simple form of the excitation space will be denoted by a {2-entry/2-level} system. After that we will add in section 3.5.2 one extra "internal" level so that a {2-entry/3-level} system is obtained. We will use the notation convention that entry levels are denoted by Greek symbols (α, β, ...), whereas Roman symbols (i, j, ...) are used for the internal levels.

3.5.1 A {2-entry/2-level} system

Consider a two-level atomic system with $\alpha = 1$ and $\zeta = +$; the ground state of the atom and ion ground respectively. Due to electron excitation kinetics the conversion $\alpha \to \zeta$ (ionization) will take place in a certain plasma region. The local surplus of $n(\zeta)$ and defect of $n(\alpha)$ created by EEK leads to transport. This is illustrated in Fig. 3.5, where the competition between *transport* in configuration space ruled by $\mathcal{T}(\alpha)$ and $\mathcal{T}(\zeta)$ and the traffic through the system (excitation space), the *chemistry*, is shown.

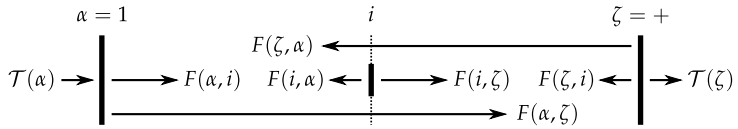


Figure 3.6. *The system of three levels, two entries, and their interacting transition frequencies.*

The set of balance equations 3.24 only contains two terms:

$$\mathcal{T}(\alpha) = \mathcal{S}(\alpha) \qquad \mathcal{S}(\alpha) \equiv -n(\alpha)\,F(\alpha,\alpha) + n(\zeta)\,F(\zeta,\alpha) \tag{3.35a}$$

$$\mathcal{T}(\zeta) = \mathcal{S}(\zeta) \qquad \mathcal{S}(\zeta) \equiv \;\; n(\alpha)\,F(\alpha,\zeta) - n(\zeta)\,F(\zeta,\zeta), \tag{3.35b}$$

whereas $\mathcal{T}(\alpha)$ and $\mathcal{T}(\zeta)$ are given by Eqn. (3.25).

In this case the chemistry is rather simple, and since there are only two levels, all destruction processes of α will lead to population of ζ and vice versa, so $F(\alpha,\alpha) = F(\alpha,\zeta)$, and also $F(\zeta,\zeta) = F(\zeta,\alpha)$. This means that $\mathcal{S}(\alpha) = -\mathcal{S}(\zeta)$ and subsequently $\mathcal{T}(\alpha) + \mathcal{T}(\zeta) = 0$, which was already announced in (3.1).

The total system density $n_s = n(\alpha) + n(\zeta)$ is therefore ruled by:

$$\frac{\partial}{\partial t} n_s + \nabla \cdot (n_s\,\mathbf{v}_s) = 0, \tag{3.36}$$

where $n_s\,\mathbf{v}_s = n(\alpha)\,\mathbf{v}(\alpha) + n(\zeta)\,\mathbf{v}(\zeta)$. The source term for α is the net recombination and for ζ it is the net ionization.

3.5.2 A {2-entry/3-level} system

The 2-entry/2-level system given above is extremely simple and in fact a CRM is not needed since only *direct* processes between α and ζ were involved. The next step towards a more complex and realistic system is the addition of an internal level to the system as shown in Fig. 3.6.

In this case the balance equations (3.24) with their respective source terms are:

$$\mathcal{T}(\alpha) = \mathcal{S}(\alpha), \qquad \mathcal{S}(\alpha) = -n(\alpha)F(\alpha,\alpha) + n(i)\,F(i,\alpha) + n(\zeta)F(\zeta,\alpha) \tag{3.37a}$$

$$0 = \mathcal{T}(i) = \mathcal{S}(i), \qquad \mathcal{S}(i) = \;\; n(\alpha)\,F(\alpha,i) - n(i)\,F(i,i) + n(\zeta)F(\zeta,i) \tag{3.37b}$$

$$\mathcal{T}(\zeta) = \mathcal{S}(\zeta), \qquad \mathcal{S}(\zeta) = \;\; n(\alpha)\,F(\alpha,\zeta) + n(i)\,F(i,\zeta) - n(\zeta)F(\zeta,\zeta) \tag{3.37c}$$

An essential aspect of this {2entry/3-level} system is that the density of level i is not sensitive for transport phenomena: $\mathcal{T}(i) = 0$, which is true as long as $F_t(i) \ll F(i,i)$, see Eqn. (3.31). Thus contrary to the levels α and ζ, which being located at the entrance of the system are influenced by densities on other plasma locations via transport, the occupation of level i only depends on the local chemistry (LC) and thus on the densities of the entrance levels.

3.5.3 The tasks of a CRM

Now we will concentrate on the three different tasks of a CRM, namely providing tools and ingredients for the calculation of:

1. the atomic state distribution function, ASDF;
2. the effective conversion frequencies;
3. source terms of the energy equations.

The study of the *task allocation* will be guided by the simple {2-entry/3-level} structure given in the previous subsection. Insights gained in this way will be used to understand more complex situations. Intuitively derived matrix relations will be justified in section 3.6.

The ASDF construction

The ASDF construction can take place if we know how the densities of *internal* states are related to those of the entry levels. For the simple {2-entry/3-level} case we can use Eqn. (3.37b) and find that:

$$n(i) = F(i,i)^{-1}\, F(\alpha,i)\, n(\alpha) + F(i,i)^{-1}\, F(\zeta,i)\, n(\zeta), \tag{3.38}$$

showing the same structure as (3.15), where also the density of the internal level(s) was determined by the addition of mutually independent contributions from each of the entry levels.

From (3.38) $n^{\alpha}(i)$ is found by setting $n(\zeta) = 0$:

$$n^{\alpha}(i) = F(i,i)^{-1}\, F(\alpha,i)\, n(\alpha). \tag{3.39}$$

This demonstrates that the contribution to $n(i)$ by level α is a balance between production from α and the total destruction factor at i. The same holds (mutatis mutandis) for the contribution by ζ to i.

Just like Eqn. (3.15), this can also be written in matrix format as in Eqn. (3.16), where now $\mathbf{n}^{l}$ only contains level i, and $\mathbf{n}^{t}$ contains levels α and ζ. Matrix $\boldsymbol{R}^{lt}$, that relates the densities of the internal levels to those of the entry levels, is in this case a 1×2 matrix. The general structure of $\boldsymbol{R}$ will be given in section 3.6, where it will be shown that the translation of $F(i,i)^{-1}F(\alpha,i)$ into the corresponding matrix form is rather straightforward.

The $\boldsymbol{R}$ matrix can be extended such that a mapping from the t-subspace to the entire excitation space can be effectuated, by:

$$\boldsymbol{R}^{st} = \begin{pmatrix} \boldsymbol{I}^{tt} \\ \boldsymbol{R}^{lt} \end{pmatrix}, \tag{3.40}$$

so that:

$$\mathbf{n}^s = \boldsymbol{R}^{st}\,\mathbf{n}^t, \tag{3.41}$$

where ^{s} denotes the entire excitation space (TS and LC levels)*, or $\{s\} = \{t\} \cup \{l\}$. Matrix $\boldsymbol{I}^{tt}$ is the identity matrix leaving entry levels unaltered. Equation (3.40) gives a simple matrix expression for ASDF.

The effective conversion frequencies

The {2-entry/3-level} system shows that apart from *direct* ionization and recombination also *stepwise* processes involving the intermediate level i are present. The combination of direct and stepwise processes is the effective conversion frequency J. The effective conversion frequency for ionization can be found by substituting equation (3.38) into (3.37c), showing that the contribution to the source of species ζ by species α is $n(\alpha)\,J(\alpha,\zeta)$, with:

$$J(\alpha,\zeta) = F(\alpha,\zeta) + F(i,\zeta)\,F(i,i)^{-1}\,F(\alpha,i). \tag{3.42}$$

This shows that the effective conversion frequency $J(\alpha,\zeta)$ equals the direct conversion frequency $F(\alpha,\zeta)$ representing the direct process $\alpha \to \zeta$, plus a fraction of the frequency $F(\alpha,i)$ representing the process $\alpha \to i$. The fraction equals the fraction of the total destruction of i, $F(i,i)$, that leads to the transition $i \to \zeta$: i.e., $F(i,\zeta)F(i,i)^{-1}$.

The effective recombination ($\zeta \to \alpha$) can be found by exchanging α and ζ in (3.42). The *net* conversion rate through the system in the direction from $\alpha \to \zeta$ is given by:

$$S(\zeta) = n(\alpha)\,J(\alpha,\zeta) - n(\zeta)\,J(\zeta,\alpha), \tag{3.43}$$

which in matrix form reads:

$$\mathbf{S}^t = \boldsymbol{J}^{tt}\mathbf{n}^t. \tag{3.44}$$

This shows that the sources of the entry levels can be expressed in J-coefficients and only depend on the densities of the TS levels.

In fact the J-coefficients are generalizations of the direct F-frequencies and by applying these effective conversion coefficients, we get a transformation of the {2-entry/3-level} system back to the {2-entry/2-level} system of section 3.5.1. Because only two entries are involved in this example, we have that $J(\alpha,\alpha) = J(\alpha,\zeta)$; that is, all conversions of α lead (direct or indirect) to a production of ζ. If there are more than two entries, such as in the example given in Fig. 3.4, it is useful to define for each entry level the total conversion rate; for instance $J(\alpha,\alpha) = J(\alpha,\zeta) + J(\alpha,\beta)$. This quantity can be found on the diagonal of the matrix $\boldsymbol{J}^{tt}$

* In cases for which it is clear from the context that we deal with the entire excitation space we omit index ^{s}. For instance $\mathbf{n}^s \equiv \mathbf{n}$ and $\boldsymbol{F}^{ss} \equiv \boldsymbol{F}$.

that stores the coefficients for the effective conversion frequencies. In line with Eqn. (3.29) we find on the diagonal for entry level ξ:

$$J^{tt}_{\xi\xi} = -J(\xi,\xi) = \sum_{\upsilon\neq\xi} J(\xi,\upsilon), \tag{3.45}$$

where υ runs over all possible entry levels.

Equation (3.44) has a structure that is comparable to the general source matrix equation $\mathbf{S} = \boldsymbol{F}\mathbf{n}$ (Eqn. (3.27)). But in contrast to Eqn. (3.27), Eqn. (3.44) does not operate on the whole excitation space but on the subspace spanned up by the entry levels. The vector at the left-hand side $\mathbf{S}^t$ contains the chemical sources for the entry levels solely. Thus $\mathbf{S}^t$ has the same dimension as the vector $\mathbf{n}^t$, namely the number of transport sensitive levels.

Energy conversion

Here we will derive important tools that can be used in the electron energy balance for EEK plasmas and we anticipate to conditions for which other excitation agents (like photons and heavy particles) are important as well. We start with the {2-entry/3-level} example. Multiplying each particle balance of Eqn. (3.37) with the energy of the level in question we get:

$$-E_\alpha\, n(\alpha)\, F(\alpha,\alpha) + E_\alpha\, n(i)\, F(i,\alpha) + E_\alpha\, n(\zeta)\, F(\zeta,\alpha) = E_\alpha\, \mathcal{T}(\alpha) \tag{3.46}$$

$$E_i\, n(\alpha)\, F(\alpha,i) - E_i\, n(i)\, F(i,i) + E_i\, n(\zeta)\, F(\zeta,i) = 0 \tag{3.47}$$

$$E_\zeta\, n(\alpha)\, F(\alpha,\zeta) + E_\zeta\, n(i)\, F(i,\zeta) - E_\zeta\, n(\zeta)\, F(\zeta,\zeta) = E_\zeta\, \mathcal{T}(\zeta). \tag{3.48}$$

Adding up these equations, and using (3.29) for the total destruction factor we get:

$$\sum_\nu E_{\alpha\nu}\, n(\alpha)\, F(\alpha,\nu) + \sum_\nu E_{i\nu}\, n(i)\, F(i,\nu) + \sum_\nu E_{\zeta\nu}\, n(\zeta)\, F(\zeta,\nu) = E_{\alpha\zeta}\, \mathcal{T}(\zeta), \tag{3.49}$$

where we have used $E_{pq} = E_p - E_q$, and on the right-hand side the fact that $\mathcal{T}(\zeta) = -\mathcal{T}(\alpha)$. In vector notation this can be written as:

$$\mathbf{E}\cdot(\boldsymbol{F}\mathbf{n}) = \mathbf{E}\cdot\mathbf{T}, \tag{3.50}$$

which was already announced in Eqn. (3.12). Here too, we can replace $\mathbf{E}\cdot\mathbf{T}$ with $\mathbf{E}^t\cdot\mathbf{T}^t$, since non-entry levels have a transport term equal to zero. Equation (3.50) can be used in the case of multiple entries.

The general form given in the matrix notation (3.50) is very compact and facilitates the algebra of the various summation procedures considerably. However, in some cases it is preferable to retain an expression like Eqn. (3.49). For instance, one should realize that each energy distance in this equation appears two times, once positive, for the energy increasing processes (excitation/ionization), and the other

time negative (de-excitation/recombination). By grouping the terms in pairs of forward and corresponding backward processes, we can write Eqn. (3.49) as:

$$\sum_{k<j}\sum_{j} E_{kj}\Big(n(k)\,F(k,j) - n(j)\,F(j,k)\Big) = E_{\alpha\zeta}\,\mathcal{T}(\zeta). \tag{3.51}$$

If we then decompose the transition frequencies into a collisional and radiative part, $F = {}^{e}F + {}^{f}F$, we get:

$$\sum_{k<j}\sum_{j} E_{kj}\Big(n(k)\,{}^{e}F(k,j) - n(j)\,{}^{e}F(j,k)\Big) + \sum_{k<j}\sum_{j} E_{kj}\Big(n(k)\,{}^{f}F(k,j) - n(j)\,{}^{f}F(j,k)\Big) = E_{\alpha\zeta}\,\mathcal{T}(\zeta). \tag{3.52}$$

Again, we can write this in matrix vector form by using $\boldsymbol{F} = {}^{e}\boldsymbol{F} + {}^{f}\boldsymbol{F}$:

$$\mathbf{E}\cdot({}^{e}\boldsymbol{F}\mathbf{n}) + \mathbf{E}\cdot({}^{f}\boldsymbol{F}\mathbf{n}) = \mathbf{E}\cdot\mathbf{T}, \tag{3.53}$$

or

$$\varepsilon_{e}^{-} + \varepsilon_{f}^{-} = \varepsilon_{c}^{+}, \tag{3.54}$$

which proves the energy conservation announced in section 3.2.1, cf. Eqn. (3.5). The term at the right-hand side $\varepsilon_{c}^{+} = \mathbf{E}\cdot\mathbf{T}$ is indeed identical to that given in Eqn. (3.12). Concerning the left-hand side we must realize that due to the definition $\varepsilon_{e}^{-} = \mathbf{E}\cdot({}^{e}\boldsymbol{F}\mathbf{n})$ and $\varepsilon_{f}^{-} = \mathbf{E}\cdot({}^{f}\boldsymbol{F}\mathbf{n})$ these power densities can be seen as the energy investment of electrons and photons into the excitation space. For the electrons this is the net result of excitation minus de-excitation, whereas for the photons it is the net result of absorption minus emission. Thus in this representation the electrons and photons are equivalent agents, both ruling the excitation space by energy investment. Equation (3.54) states that the power density associated to the investment done by electrons and photons into the system is the same as that associated to the efflux of radicals* (chemistry). Treating electrons, photons, and radicals on an equal footing would lead to a Kirchhoff-like expression, cf. Eqn. (3.5):

$$\varepsilon_{e}^{-} + \varepsilon_{f}^{-} + \varepsilon_{c}^{-} = 0. \tag{3.55}$$

Note that by introducing the effective decay probability $A*$, cf Eqn. (3.18), the interaction of the radiation field is presented as an effective emission process, so that $\varepsilon_{f}^{-} = \mathbf{E}\cdot({}^{f}\boldsymbol{F}\mathbf{n})$ is negative. This can more easily be seen by inspecting the double sum in Eqn. (3.52) where all the frequencies ${}^{f}\boldsymbol{F}(k,j)$, for which $k < j$, are zero, so that only the negative terms $n(j)\,{}^{f}F(j,k)$ survive.

* Note that we use the term radical in a general way. It is meant as a species effectuating a reaction in the atomic system.

We can use the expression for the ASDF (3.40) to derive an expression for the power density of the radiation gain term:

$$\varepsilon_f^+ = -\mathbf{E} \cdot ({}^f\boldsymbol{F}\mathbf{n}) = \mathbf{L}^t\mathbf{n}^t, \tag{3.56}$$

where we use the definition:

$$\mathbf{L}^t = -\mathbf{E} \cdot ({}^f\boldsymbol{F}^{ss}\boldsymbol{R}^{st}) \tag{3.57}$$

a vector consisting of the specific effective emissivities, see section 3.2.1. So, the $\mathbf{L}^t$ vector allows us to express the plasma emission in the densities of the TS levels (vector $\mathbf{n}^t$). The matrix $\boldsymbol{R}^{st}$ is already known, since it is needed for task 1; the determination of the ASDF, Eqn. (3.16).

Similarly, we can derive an expression for the energy associated to the transport of radicals:

$$\varepsilon_c^+ = \Big(n(\alpha)\, J(\alpha,\zeta) - n(\zeta)\, J(\zeta,\alpha)\Big) E_{\alpha\zeta}, \tag{3.58}$$

or in matrix-vector notation:

$$\varepsilon_c^+ = \mathbf{E}^t \cdot \mathbf{T}^t = \mathbf{E}^t \cdot (\boldsymbol{J}^{tt}\mathbf{n}^t). \tag{3.59}$$

As the J matrix is known in the framework of the CRM, task 2 (Eqn. (3.42)), we can express the power density associated to the transport of charged particles again in densities of the TS levels.

The energy-coefficients given here are essential ingredients for the electron energy equation of EEK plasmas. They can be used to convert the inelastic terms of the "energy" moment of the Boltzmann transport equation in which the excitation space is involved. However, it should be realized that there are inelastic processes in which the excitation space is not (completely) involved; such as the radiative losses in free-free and free-bound transitions. Therefore, the derivation given in this section provides only specific tools for the energy balance. It is not a complete derivation. The energy balance for non-LTE plasmas is beyond the scope of this treatment.

The examples given until now are related to EEK plasmas where the electron gas $\{e\}$ is the only leading agent of excitation and thus radiation generation and radical production. In a general treatment we must admit other agents on an equal footing as $\{e\}$ so that Eqn. (3.53) changes into:

$$\sum_a \mathbf{E} \cdot ({}^a\boldsymbol{F}\mathbf{n}) = \mathbf{E} \cdot \mathbf{T}, \tag{3.60}$$

where the summation extends over all the agents $[a = \{e\}, \{f\}, \{h\}, \ldots]$: that is electrons, photons, the heavy particles and possible other active species. This means that extra branches can and must be added in the Kirchhoff-like scheme.

3.6 Generalization

In the preceding section we occasionally predicted how the insights obtained by the 3-level example can be extended towards the general case of a block of LC levels and more than two entries; i.e. TS levels. Especially the generalization of task 3 was found to be rather straightforward. However, some of the tasks of the general CRM were postponed and will be treated here by giving a short overview of this general case. More details can be found in [24] and [9].

The source equations were given in Eqn. (3.27) by the matrix notation $\mathbf{S} = \boldsymbol{F}\mathbf{n}$. For the general case of an LC-block and more than two TS levels it is useful to re-arrange the elements of vectors $\mathbf{S}$ and $\mathbf{n}$ in sub-vectors and matrix $\boldsymbol{F}$ in blocks. The following structure is obtained:

$$\begin{pmatrix} \mathbf{S}^t \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}^{tt} & \boldsymbol{F}^{tl} \\ \boldsymbol{F}^{lt} & \boldsymbol{F}^{ll} \end{pmatrix} \begin{pmatrix} \mathbf{n}^t \\ \mathbf{n}^l \end{pmatrix} \qquad \text{or} \qquad \begin{pmatrix} \mathbf{S}^t \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}^{tt}\,\mathbf{n}^t + \boldsymbol{F}^{tl}\,\mathbf{n}^l \\ \boldsymbol{F}^{lt}\,\mathbf{n}^t + \boldsymbol{F}^{ll}\,\mathbf{n}^l \end{pmatrix}, \tag{3.61}$$

where the sub-vectors $\mathbf{n}^t$ and $\mathbf{n}^l$ consist respectively of densities of the entry and internal levels. Parallel we have the division of the source vector into $\mathbf{S}^t$ and $\mathbf{S}^l$ for which, anticipating on the transport vector, the latter was set equal to zero; i.e. $\mathbf{S}^l = \mathbf{0}$. The four $\boldsymbol{F}$ sub-matrices are related to the four different types of excitation traffic routes, between the possible combinations of t and l blocks. For instance, $\boldsymbol{F}^{tl}$ refers to the traffic between blocks $t \to l$, while $\boldsymbol{F}^{ll}$ relates to the internal traffic between the LC levels. Comparing this matrix expression with that of the simple three level system makes the following generalization plausible.

Task 1: The ASDF

The ASDF still contains as many components as there are entry levels. In the 2-entry case we had two of them, now this number is in principle unlimited. Solving the lower line of the matrix representation, Eqn. (3.61), we get the following generalization of Eqn. (3.40):

$$\mathbf{n}^l = -\left(\boldsymbol{F}^{ll}\right)^{-1} \boldsymbol{F}^{tl}\,\mathbf{n}^t. \tag{3.62}$$

In comparison to the three level case given by Eqn. (3.38), we see that the transformation to the matrix representation given here is rather straightforward; we only need to replace the F frequency by the appropriate sub matrix of $\boldsymbol{F}$. The counter intuitive minus sign in the matrix equation corresponds to the fact that $F_{pp} = -F(p,p)$, see Eqn. (3.29).

In the form of Eqn. (3.40), the ASDF is given by:

$$\mathbf{n}^s = \boldsymbol{R}^{st}\mathbf{n}^t \qquad \text{with} \qquad \boldsymbol{R}^{st} = \begin{pmatrix} \boldsymbol{I}^{tt} \\ -(\boldsymbol{F}^{ll})^{-1}\,\boldsymbol{F}^{tl} \end{pmatrix}. \tag{3.63}$$

Task 2: Effective conversion rates

The effective conversion rates are represented in the $\boldsymbol{J}$ matrix, Eqn. (3.44), with elements defined by (3.42). It gives the effective conversion rate between each pair of entry levels and is a square matrix with a rank equal to the number of entry levels. We can use the definition of the four sub-matrices of the transition matrix as defined in (3.61), to obtain $\boldsymbol{J}$, which is given by:

$$\boldsymbol{J} = \boldsymbol{F}^{tt} - \boldsymbol{F}^{lt} \left(\boldsymbol{F}^{ll}\right)^{-1} \boldsymbol{F}^{tl}. \tag{3.64}$$

As was the case with the ASDF in Eqn. (3.63), the minus sign originates from Eqn. (3.29).

Task 3: Energy source terms

The matrix equations for the energy power density of the photons and radicals were already given in Eqns. (3.56) and (3.59). In generalized form they do not change, the only issue is that a generalized form of $\boldsymbol{L}^t$, Eqn. (3.57), and $\boldsymbol{T}^t$ is needed. The result of task 1 is used for matrix $\boldsymbol{L}^t$, and the result of task 2 for matrix $\boldsymbol{T}^t$. We will give the results here for completeness, starting with the power density for radiation losses:

$$\varepsilon_f^+ = -\mathbf{E} \cdot \left({}^f\boldsymbol{F}\mathbf{n}\right) = -\mathbf{E} \cdot \left({}^f\boldsymbol{F}^{ss}\boldsymbol{R}^{st}\right) \cdot \mathbf{n}^t = \mathbf{L}^t \cdot \mathbf{n}^t, \tag{3.65}$$

where we have used Eqn. (3.63).

The power density associated to the radicals is given by:

$$\varepsilon_c^+ = \mathbf{E}^t \cdot \mathbf{T}^t = \mathbf{E}^t \cdot \left(\boldsymbol{J}\mathbf{n}^t\right), \tag{3.66}$$

where Eqn. (3.64) is used.

3.7 Further generalization

We have already seen that, due to its linear nature, the $\boldsymbol{F}$-matrix can be split up for instance into $\boldsymbol{F} = {}^e\boldsymbol{F} + {}^f\boldsymbol{F}$, as in Eqn. (3.30). This leads to a linear split up in the $\boldsymbol{R}$ (Eqn. (3.63)) and $\boldsymbol{J}$ (Eqn. (3.64)) matrices so that the implementation is straightforward. In the same way it might be needed to admit other excitation agents to the same atomic system. The consequences with respect to the energy equations were already discussed in section 3.5.3. However, apart from admitting more agents along the same channels it might also be needed to implement completely different sources. For instance if we want to construct a CRM for *atomic* nitrogen we have to include all kinds of processes between excited atomic nitrogen states. We can start the construction of such a model by describing the effect

of the following excitation agents: the group of electrons, photons, and heavy particles. However, for a better description of reality we have to add sources that are molecular of origin. For instance the dissociative recombination channel:

$$N_2^+ + e \rightarrow N^* + N, \tag{3.67}$$

will offer an extra and important source of excited nitrogen atoms. This means that our formalism has to be extended so that the solution of more general source matrices can be treated as well. The source vector as defined in Eqn. (3.27) is extended with an extra term:

$$\mathbf{S} = \boldsymbol{F}\mathbf{n} + \mathbf{S}_{\text{ext}}, \tag{3.68}$$

which can be split up, as in Eqn. (3.61), to:

$$\begin{pmatrix} \mathbf{S}^t \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}^{tt} & \boldsymbol{F}^{lt} \\ \boldsymbol{F}^{tl} & \boldsymbol{F}^{ll} \end{pmatrix} \begin{pmatrix} \mathbf{n}^t \\ \mathbf{n}^l \end{pmatrix} + \begin{pmatrix} \mathbf{S}^t_{\text{ext}}(\mathbf{n}^t, \mathbf{n}^l) \\ \mathbf{S}^l_{\text{ext}}(\mathbf{n}^t, \mathbf{n}^l) \end{pmatrix}. \tag{3.69}$$

The matrix algebra related to this extension is rather straightforward and can be found in [27]. Here, we will only give the results for the ASDF and radiation losses (tasks 1 and 3).

The external source term is dependent on TS densities and LC densities, and typically non-linear, requiring a numerical solution procedure. If the external source term for LC levels only depends on TS levels, or $\mathbf{S}^l_{\text{ext}} = \mathbf{S}^l_{\text{ext}}(\mathbf{n}^t)$, an analytical description is possible.

For the ASDF we can solve Eqn. (3.69), which gives the result for the internal levels:

$$\mathbf{n}^l = -(\boldsymbol{F}^{ll})^{-1} \left[\boldsymbol{F}^{tl}\, \mathbf{n}^t + \mathbf{S}^l_{\text{ext}}(\mathbf{n}^t) \right]. \tag{3.70}$$

The density vector of the complete system is then, following Eqn. (3.63):

$$\mathbf{n} = \begin{pmatrix} \boldsymbol{I} \\ -(\boldsymbol{F}^{ll})^{-1}\, \boldsymbol{F}^{tl} \end{pmatrix} \mathbf{n}^t + \begin{pmatrix} \mathbf{0} \\ -(\boldsymbol{F}^{ll})^{-1} \end{pmatrix} \mathbf{S}^l_{\text{ext}}(\mathbf{n}^t) \tag{3.71}$$

$$\equiv \boldsymbol{R}^{st}\, \mathbf{n}^t + \boldsymbol{Q}\mathbf{S}^l_{\text{ext}}(\mathbf{n}^t) \equiv \mathbf{n}_{\text{exc.sp.}} + \mathbf{n}_{\text{ext}}. \tag{3.72}$$

This result, combined with Eqn. (3.65), gives the power density for radiation losses:

$$\varepsilon_f^+ = -\mathbf{L}^t \cdot \mathbf{n}^t + \mathbf{E} \cdot \left({}^f\boldsymbol{F}^{ss} \boldsymbol{Q} \mathbf{S}_{\text{ext}}(\mathbf{n}^t) \right). \tag{3.73}$$

An application of this extended source method will be given in the next section which is devoted to the effect of radiation transport.

3.8 Level sensitive to radiation transport

In the previous section an outline was presented for a general CRM with more than two entries. Here we will give an example of the application of such a system, namely a system containing a level that is sensitive to radiation transport. Consider the {2-entry/3-level} system as discussed in section 3.5.2. The density of the internal (LC) level i is determined by the balance equation:

$$\mathcal{S}(i) = n(\alpha)\, F(\alpha, i) - n(i)\, F(i, i) + n(\zeta)\, F(\zeta, i) = 0 \tag{3.74}$$

The right-hand side of this equation was put equal to zero since the fluid transport processes, usually with time scales in the order 10^{-4} s, are supposed to be much slower than those associated to elementary processes. Therefore the elementary processes have to compensate each other on each location; the overall source term $\mathcal{S}(i)$ must be zero. So the density of level i depends solely on the local chemistry.

Now suppose that i is a resonant level, which decays to the ground level under the emission of a photon. The decay probability $A(i, \alpha)$ of this elementary process is often in the range 10^7 s^{-1} which justifies the reasoning sketched above. However, due the large density of the ground level it is likely that a substantial part of the emitted radiation will be reabsorbed, which implies that the reverse process of absorption will lead to a high population frequency for the resonant level. The transport of radiation can be seen as a transport of radiative species and the corresponding transport frequency is much larger than that of material transport, such as generated by convection or diffusion.

Normally the emission and re-absorption of radiation is accounted for by using the *escape factor*, changing $A(i, \alpha)$ into $A^*(i, \alpha) = \theta_{i\alpha} A(i, \alpha)$ (see Eqn. (3.18)):

$$n(i)\, \theta_{i,\alpha}\, A(i, \alpha) = n(i)\, A(i, \alpha) - [n(\alpha)\, B(\alpha, i) - n(i)\, B(i, \alpha)]\, \rho_\nu. \tag{3.75}$$

The above equation gives the definition of the escape factor θ. In fact this is a misnomer since it suggests that we deal with the chance that a photon emitted at a certain plasma location escapes from the plasma. Thus one would expect that $0 < \theta \leq 1$, which is not the case, in general. What θ really gives is the *normalized* net *emission factor*; that is the difference between spontaneous emission and absorption normalized on emission. Since stimulated emission is accounted for as negative absorption it is possible that $\theta > 1$ for those conditions for which stimulated emission prevails (see also table 4.1). On the other hand, if there is more absorption than emission we can get $\theta < 0$. Thus θ does not simply provide the escape probability, it accounts for the net effect of emission and absorption.

Now suppose that the radiation field is not uniform, so that ρ_ν in Eqn. (3.75) is spatially dependent. In that case we can not use a constant escape factor and the transport as induced by this inhomogeneity is very important. This means that the associated transport frequency is in the order of $A(i, \alpha)$. This implies that

$A^*(i,\alpha)$ in Eqn. (3.74) has to be replaced by the right-hand side of Eqn. (3.75) so that Eqn. (3.74) changes to:

$$n(\alpha)\,F^{\blacklozenge}(\alpha,i) - n(i)\,F^{\blacklozenge}(i,i) + n(\zeta)\,F(\zeta,i) = \\ n(i)\,A(i,\alpha) - [n(\alpha)\,B(\alpha,i) - n(i)\,B(i,\alpha)]\,\rho_\nu \quad (3.76)$$

In this equation we have corrected $F(i,i)$ into $F^{\blacklozenge}(i,i)$ and $F(\alpha,i)$ into $F^{\blacklozenge}(\alpha,i)$, a direct consequence of replacing the constant A^* by its form given in Eqn. (3.75). At the left-hand side the LC processes are retained, such as (de-)excitation due to electron collisions. At the right-hand side we will have a (radiative) transport term which can be treated as the $\mathbf{S}_{\text{ex}}$ in Eqn. (3.68). It can be seen that level i is no longer an LC level but sensitive to transport; more precisely radiation transport. Another consequence is that a comparable $\mathbf{S}_{\text{ex}}$ term has to be present at the right-hand side of the balance equation for the ground level α. But there is more; due to the fact that ρ_ν is determined by emission of other plasma parts, we can no longer retain a local solution.

A solution procedure that can be employed is know as the Ray-Trace Control-Volume (RTCV) method and consists of two steps [28, 29]. The method starts with a given plasma composition of which the radiation intensity field and thus ρ_ν can be determined using a ray-trace (RT) method. This ρ_ν provides (new) values of θ, that is to say the right-hand side of Eqn. (3.76), and in fact the value of $\mathbf{S}_{\text{ex}}$. Subsequently a control volume (CV) method is used to solve the system of the combined modules* of fluid plus chemistry as prescribed by the new CRM. The resulting new plasma composition provides ingredients to compute the radiation field again so that the first step can be repeated. This obviously is an iterative procedure.

The first step the ray-trace module takes, is to make a selection and discretization of the spectral region of interest. Next, a network of lines (rays) through the plasma has to be constructed. The number of lines largely depends on symmetry aspects. Along these lines the intensity evolution is computed using the radiation transfer equation:

$$\frac{dI_\nu(\nu)}{d\nu} = j_\nu - k(\nu)\,I_\nu(\nu), \quad (3.77)$$

where $I_\nu(\nu)$ is the spectral intensity (in $\mathrm{W\,m^{-2}\,sr^{-1}\,Hz^{-1}}$) for the direction along the lines through the plasma. Each line starts outside the plasma with $I_\nu(\nu) = 0$. While entering the plasma $I_\nu(\nu)$ will grow due to the emission generated in the outer plasma layers. Proceeding further along the line the intensity will increase but (depending on the k-value) absorption will realize an energy transfer from the ray to the plasma. The computation (in the first step) of the evolution of I_ν along

* The term "modules" refers to the fact that this model was implemented in the plasimo plasma modeling framework. In plasimo complex models can be constructed by combining building blocks called modules; fluid and chemistry being just two of the many available. See also chapter 5.

the rays is of course prescribed by the j and k values. These are closely related to the A and the B in Eqns. (3.75) and (3.76). The value of ρ_ν can be determined at the nodal points of each control by computing the direction-average of the I_ν values for the different (selected) directions.

3.9 Time dependence

In the preceding discussion, the Quasi Steady State Solution was used to express the three tasks of a CRM in simple expressions. Considering the ASDF, the densities of the LC levels, vector $\mathbf{n}^l$, were mapped to the densities of the TS levels, vector $\mathbf{n}^t$, meaning that a solution requires fixed densities of the TS levels. We are however also interested in a full time dependent system, in which the system of balance equations is:

$$\frac{\partial \mathbf{n}(t)}{\partial t} + \nabla \cdot (\mathbf{n}(t)\mathbf{v}(t)) = \mathbf{S}(t). \tag{3.78}$$

The transport can be moved to the right-hand side, treating it like an additional (destructive) source term $\mathbf{S}_{\text{tr}}$:

$$\frac{\partial \mathbf{n}(t)}{\partial t} = \boldsymbol{F}(t)\mathbf{n}(t) - \mathbf{S}_{\text{tr}}(t) + \mathbf{S}_{\text{ext}}(t), \tag{3.79}$$

where also external sources are included, making the system potentially non-linear. The densities of all the levels at $t = 0$ are known (for instance from a QSS solution), making it an initial value problem. Due to the complexity of the set of balances and their source terms this usually requires the set of equations to be solved numerically as a function of time.

In case all source terms in Eqn. (3.79) are independent of time, it can still be useful to study the response of a disturbance in the equilibrium ASDF as a function of time.

Time dependence can enter equation (3.79) through the transport and external source vectors, but also the transition frequency matrix can be time dependent. Here, we will only discuss the latter.

The most obvious manner in which the transition frequency matrix can be time dependent is through the electrons. Both the electron density, n_e, and the Electron Energy Distribution Function (EEDF), f_{eedf} can be time dependent, resulting in a time dependence of the rate coefficients, since:

$${}^{e}F_{pq}(t) = n_e(t)K(p,q)(t) = n_e(t) \int_{E_{pq}}^{\infty} \sigma_{pq}(E')\, f_{\text{eedf}}(E',t)\, \sqrt{E'/2m}\, dE'. \tag{3.80}$$

This will be used in a CRM of an Ar plasma induced by Extreme UltraViolet radiation, presented in chapter 7.

In chapter 8 we will present a time dependent CRM for Laser Induced Fluorescence (LIF) experiments in Ar. In that CRM, time dependence is the result of a short laser pulse that excites an excited level to another, higher level. The model starts with a steady state solution, calculated using QSSS. While the plasma conditions remain the same, a single, time dependent transition is added to the transition matrix. This transition represents the excitation (absorption) of the lower level and de-excitation ((stimulated) emission) of the upper level by the laser pulse. With the model the evolution in time of the ASDF, and dominant processes in restoring the steady state can be studied.

As such, the time dependent CRM is no longer a tool to provide a fluid model with parameters for a reduced chemistry, it is an independent model useful to study the various chemistry aspects in a system.

3.10 Conclusion

Collisional radiative models play an important role in the description of the non-equilibrium chemistry in plasmas. Basically we have seen that there are two schemes: one in which transport of radiation is not important and another in which it forms an essential part of the transport-chemistry interaction. In the first case we can run the CRM module separate from the fluid module and use the CRM output to get transport coefficients and source terms for the fluid equations. In the second case this is not possible and an iterative procedure has to be constructed in order to come to a proper and self-consistent description of the trinity: fluid, chemistry and radiation. In the first category we find apart from the laboratory plasma as based on noble gases (e.g., Ar and He), also the reactive plasmas for which the local chemistry is extremely important. In the second category we meet with plasmas for light generation. Since these plasmas are intended to generate large radiative fluxes it is obvious that this implies a central role for radiation transfer. In many cases it is insufficient to employ a simple method with a constant escape factor. Instead a method is needed in which the light emitting species is not treated as LC level but as being sensitive to transport.

It has been shown how the task package of the CRM has to be prepared for this job and how the matrix representation can be employed to tackle the tedious algebra associated to extra sources in excitation space. This mathematical procedure can be used in the future to modify existing atomic CRMs such that molecular processes can be treated as well. This will extend the present application domain of atomic CRMs towards the plasmas found in many modern applications.

In case time dependency is included the solution procedure is quite different. Time dependence can enter the system through level densities not being at their equilibrium value or through the properties of the agents that change over time. Since we are no longer dependent on a QSS solution (apart from possibly using

it for the initial values) we have more freedom in processes that are taken into consideration; i.e., non linear.

In the following chapters (chapters 4 and 5) an implementation of both versions will be presented. This entails the construction of the matrices and vectors as presented in section 3.6. Additionally, the time dependent CRM will be applied in two different types plasmas; in chapter 7 to EUV induced Ar plasmas, and in chapter 8 to LIF experiments in Ar plasmas.

Chapter 4

Description of a general CRM code

4.1 Introduction

In chapter 2 we introduced the models that will be dealt with in this thesis: Collisional Radiative Models (CRM) and Global Plasma Models (GPM). In chapter 3 the tasks of a CRM were described, and in this chapter the first step towards an implementation will be taken. The goal is to create a general CRM code in the plasimo framework. Certain implementation specifics will be described in the next chapter (5). Here we will outline a mathematical description of the calculations the CRM should perform (section 4.2), and we will treat the construction of the transition-matrix, which holds all interactions between the levels in the model (section 4.3).

4.2 Model

The tasks of the CRM were described in detail in the previous chapter. For the steady state model they are to calculate:

1. the atomic state distribution function (ASDF);
2. the effective conversion rates;
3. the source terms for the energy equation.

For the time dependent model the goal is to calculate the species densities as a function of time. When these are known, other characteristics such as the spectral

emission, can easily be determined. In all cases the system is described by 0-th moment of the Boltzmann transport equation, which for a single species reads:

$$\frac{\partial}{\partial t} n_s + \nabla \cdot (n_s \mathbf{v}_s) = \mathcal{S}_s \tag{4.1}$$

where n_s, $\mathbf{v}_s$, and $\mathcal{S}_s$ are the density, velocity, and source term of species s, respectively. In chapter 2 it was shown how this leads to the treatment in species space and reaction space; the first being relevant for CRMs, the second is the basis of a GPM. In chapter 3 it was shown how the treatment in species space leads to solutions for the task of a CRM. Here we will give a brief outline.

4.2.1 Quasi Steady State (QSS)

In the Quasi Steady State solution we distinguish between Transport Sensitive (TS) and Local Chemistry (LC) species or levels. For the TS levels, which often have a relatively high density, the chemical processes impacting that density compete with transport mechanisms. In contrast, the chemical processes of LC levels are so fast that transport can be neglected. The result is that in equation (4.1) the left-hand side is zero for LC levels, meaning that $\mathbf{S} = \mathbf{0}$. Furthermore, when we only take linear processes into account we can write:

$$\mathbf{S} = \boldsymbol{F}\mathbf{n}, \tag{4.2}$$

where $\mathbf{S}$ and $\mathbf{n}$ are vectors containing the sources and densities of the levels, while $\boldsymbol{F}$ is a matrix composed of the frequencies describing the processes between the levels. From the matrix $\boldsymbol{F}$ and the densities of the TS levels the three tasks of the CRM can be determined, see section 3.6. Mathematically, the three tasks are expressed by:

1. ASDF: $\mathbf{n} = \begin{pmatrix} \boldsymbol{I} \\ -(\boldsymbol{F}^{ll})^{-1}\boldsymbol{F}^{tl} \end{pmatrix} \mathbf{n}^t$,
2. effective conversion rates: $\boldsymbol{J} = \boldsymbol{F}^{tt} - \boldsymbol{F}^{lt}(\boldsymbol{F}^{ll})^{-1}\boldsymbol{F}^{tl}$,
3. energy source terms: $\varepsilon_f^+ = -\mathbf{E} \cdot ({}^f\boldsymbol{F}\mathbf{n})$,

where the frequency matrix split into four parts as defined in equation (3.61) is used, $\mathbf{n}$ and $\mathbf{n}^t$ are the density vector and the vector of TS densities, and $\mathbf{E}$ is a vector composed of the level energies. The prefix ^{f} attached to the transition matrix in item 3 denotes the fact that only the radiative transitions (where the photons are the agents) are included in this matrix.

These equations show, that when the matrix $\boldsymbol{F}$ is known, the desired information can easily be obtained with simple matrix-vector calculations.

4.2.2 Transient behavior

In the transient case, where the source vector $\mathbf{S}$ can be time dependent, the set of level density balance equations is treated as an initial value problem. This implies, of course, that the densities at the start of the simulation are known, which can be achieved by first running a QSS case. As in the QSS case, the source vector is written in linear form (4.2), and non-linear terms and transport terms can be added to the source vector, see Eqn. (3.68).

In general, an initial value problem for a vector containing a set of time dependent variables $\mathbf{y}(t)$ has the form:

$$\frac{d\mathbf{y}(t)}{dt} = \boldsymbol{f}(t, \mathbf{y}(t)), \tag{4.3}$$

and the initial value can be denoted by $\mathbf{y}(t_0) = \mathbf{y}_0$. The simplest way to get a solution for the time $t = t_0 + \Delta t$ is:

$$\mathbf{y}(t_0 + \Delta t) \doteq \mathbf{y}(t_0) + \Delta t\, \boldsymbol{f}(t_0, \mathbf{y}(t_0)). \tag{4.4}$$

This is the *forward* or *explicit* Euler method. When we apply this to our problem of the time dependent CRM, we have to replace vector $\mathbf{y}$ with the density vector $\mathbf{n}$ and the function $\boldsymbol{f}$ with the source vector $\mathbf{S}$. If we also write the source vector in linear form, as in equation (4.2), we get:

$$\mathbf{n}(t_0 + \Delta t) = \mathbf{n}(t_0) + \Delta t\, \boldsymbol{F}(t_0)\mathbf{n}(t_0). \tag{4.5}$$

We use the general $\mathbf{y}$ instead of $\mathbf{n}$ since in chapter 6 dealing with a GPM implementation, we will also use this solution method. In the GPM vector $\mathbf{y}$ not only contains the species densities but also the electron energy density.

Apart from the forward Euler method there are many more methods available offering increased accuracy, stability, and/or computational speed, such as Runge-Kutta methods. To obtain flexibility and offer various possibilities a general *stepper* scheme is implemented, so that different methods of taking a computational step can be used. In general the solution of a vector $\mathbf{y}$ after a time step Δt is determined by some function $\boldsymbol{g}$:

$$\mathbf{y}(t_0 + \Delta t) = \mathbf{y}(t_0) + \boldsymbol{g}(t_0, \Delta t, \mathbf{y}(t_0), \boldsymbol{f}). \tag{4.6}$$

The function $\boldsymbol{g}$ determines for which t and vector $\mathbf{y}$ the function $\boldsymbol{f}$ is evaluated. The function $\boldsymbol{g}$ can then be implemented for different solution methods, and in the case of forward Euler it simply reads:

$$\boldsymbol{g}(t_0, \Delta t, \mathbf{y}(t_0), \boldsymbol{f}) = \Delta t\, \boldsymbol{f}(t_0, \mathbf{y}(t_0)). \tag{4.7}$$

Evidently, the QSS solution and the time dependent solutions share the usage of the transition-matrix $\boldsymbol{F}$, which in the latter case is time dependent. When this

matrix is known the solution procedures are rather straightforward. The bulk of the implementation lies therefore in the construction of this matrix.

We will now continue with a description of the processes that can be included in this matrix. When all the processes are described, we will show how they form the transition matrix. The actual implementation details in the PLASIMO framework are treated in chapter 5.

4.3 Transition-matrix

The transition-matrix $\boldsymbol{F}$, that is defined by Eqn. (4.2), consists of frequencies describing the transitions between the different excited states. The current CRM version contains the following types of processes:

- electron (de-)excitation;
- electron ionization;
- two-electron recombination;
- radiative transitions (bound-bound);
- radiative recombination (free-bound).

The PLASIMO CRM code constructs the transition-matrix from building blocks representing these separate processes. We will now give a description of each in detail.

4.3.1 Electron excitation

In this process a particle (X) with a certain energy level (p) collides with an electron, resulting in a higher excited state (q) of the particle:

$$X_p + e + \left(E_{pq}\right) \overset{K(p,q)}{\rightarrow} X_q + e, \tag{4.8}$$

where E_{pq} is the energy difference between levels p and q, and $K(p,q)$ is the rate coefficient. The probability for this reaction is determined by the cross section $\sigma_{pq}(E)$, and the rate coefficient for this reaction is obtained by averaging the product of the cross section and velocity over the electron energy distribution function (EEDF):

$$K(p,q) = \langle \sigma v \rangle = \int_{E_{pq}}^{\infty} \sigma_{pq}(E')\, v(E')\, f(E')\, dE', \tag{4.9}$$

with E the electron energy, $f(E)$ the EEDF, and $v(E) = \sqrt{2E/m_e}$ the electron velocity, with m_e the electron mass. In this integration the lower limit is the threshold energy for the reaction. As we deal with excitation, that is endothermic, processes the internal energy of level q will be higher than that of the initial level p, so

$E_q > E_p$, and the threshold energy is $E_{pq} = E_q - E_p > 0$. In the reverse case of de-excitation (exothermic) the energy of the initial level is higher and the threshold energy is 0.

The cross sections $\sigma(E)$ and EEDFs $f(E)$ that are supported by the CRM are merely functions of energy and can be described in two ways: as look-up tables or as mathematical formulae with a certain set of parameters. The EEDFs that can be used are either Maxwellian or non-Maxwellian. A Maxwellian EEDF can be described by a single parameter: the electron temperature. The use of non-Maxwellian EEDFs of any form is supported; they can be described by look-up tables, representing multi-temperature, Druyvesteyn, or extremely non-Maxwellian (see chapter 7) EEDFs.

Cross sections are determined either by experiments or some form of calculations (Born-Bethe, Hartree-Fock). The results are published as look-up tables or as parameters for fit functions. In this CRM code both options are supported. Some common cross section fit functions are implemented in the CRM, such as a:

- step function;
- Pots fit;
- ion fit;
- Hartgers fit;
- Vriens-Smeets, for excitation and ionization;
- Drawin.

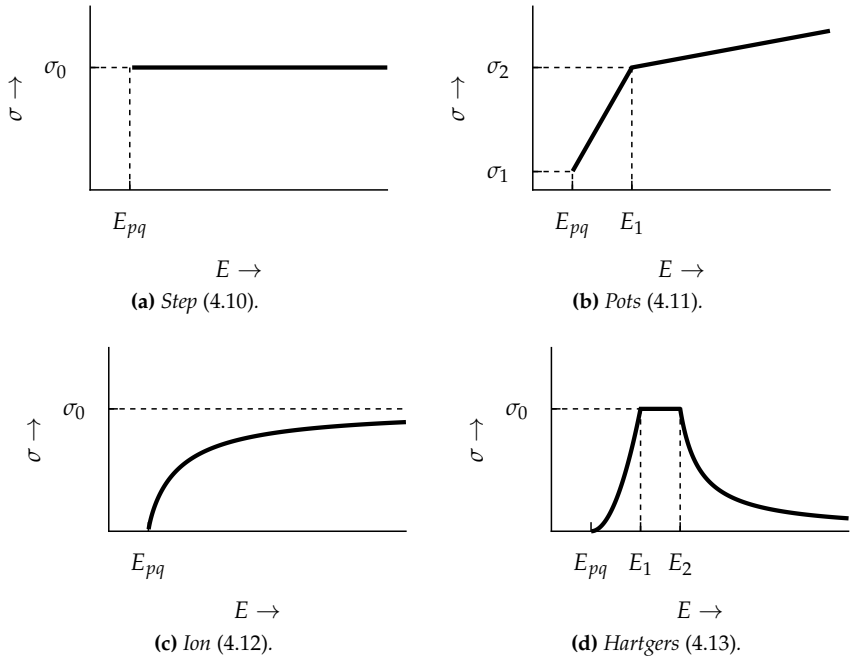
To use any one of these in a CRM only the appropriate fit parameters are required. Additionally, cross sections can be defined by custom fit functions (with a variable set of parameters) or, as stated before, by look-up tables.

Step function

The simplest function that can be used to describe a cross section is the step function (see Fig. 4.1a):

$$\sigma_{pq}(E) = \begin{cases} 0 & \text{for } E < E_{pq} \\ \sigma_0 & \text{for } E \geq E_{pq}, \end{cases} \tag{4.10}$$

meaning that the cross section is simply σ_0 when the energy exceeds the threshold energy E_{pq}.



(a) *Step* (4.10). **(b)** *Pots* (4.11). **(c)** *Ion* (4.12). **(d)** *Hartgers* (4.13).

Figure 4.1. *Examples of cross section functions.*

Pots fit

The simple step function can be given some more structure by using the following function, that will be referred to as the *Pots* function (see Fig. 4.1b):

$$\sigma_{pq}(E) = \begin{cases} 0 & \text{for } E < E_{pq} \\ \sigma_1 + s_1(E - E_{pq}) & \text{for } E_{pq} \leq E \leq E_1 \\ \sigma_2 + s_2(E - E_1) & \text{for } E > E_1, \end{cases} \tag{4.11}$$

which is defined by the values σ_1, σ_2, E_{pq}, E_1, s_1, and s_2. The first slope is determined by $s_1 = (\sigma_2 - \sigma_1)/(E_1 - E_{pq})$. By selecting $s_2 = 0$ and $\sigma_1 = \sigma_2$ (resulting in $s_1 = 0$) we get the step function where σ_0 of Eqn. (4.10) is replaced with σ_1 ($= \sigma_2$).

Ion fit

The ion fit cross section is often used for describing ionization cross sections, but can be used for excitation cross sections as well:

$$\sigma_{pq}(E) = \begin{cases} 0 & \text{for } E < E_{pq} \\ \sigma_0 \left(1 - \dfrac{E_{pq}}{E}\right)^n & \text{for } E \geq E_{pq}, \end{cases} \tag{4.12}$$

where σ_0 and n are the adjustment parameters. It can be seen as a smooth form of the step function as shown in Fig. 4.1c.

Hartgers fit

The following function is useful for fitting more complex shapes (see Fig. 4.1d), and is referred to as the *Hartgers* function:

$$\sigma_{pq}(E) = \begin{cases} 0 & \text{for } E < E_{pq} \\ \sigma_0 \left(\dfrac{E - E_{pq}}{E_1 - E_{pq}}\right)^n & \text{for } E_{pq} \leq E < E_1 \\ \sigma_0 & \text{for } E_1 \leq E < E_2 \\ \sigma_0 \dfrac{E_2 - E_3}{E - E_3} & \text{for } E \geq E_2, \end{cases} \tag{4.13}$$

where σ_0, n, E_1, E_2, and E_3 are the adjustment parameters, with $E_3 < E_2$ determining the curvature for $E > E_2$.

Vriens-Smeets

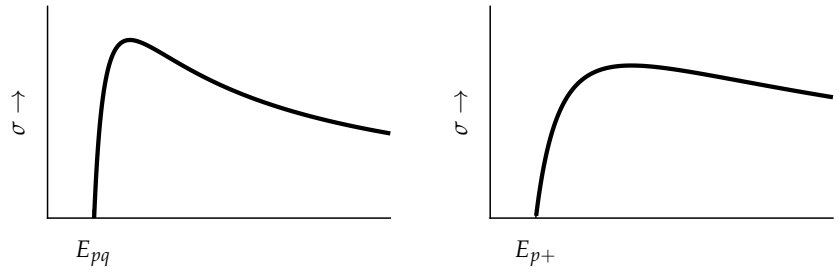
Vriens and Smeets [30] have combined experimental and theoretical results semi-empirically into practical formulae. They give cross sections and rate coefficients that are supposed to be applicable to electronic transitions between levels with only one electron in the outer orbit, such as transitions between excited levels or transitions from the ground state of alkali atoms. The following expression is given for electron excitation:

$$\sigma_{pq}(E) = 4\pi a_0^2 \frac{Ry}{E + \gamma_{pq}} \left[\mathcal{A}_{pq} \ln\left(\frac{E}{2Ry} + \delta_{pq}\right) + \mathcal{B}_{pq}\right], \tag{4.14}$$

where a_0 is the Bohr radius, and $Ry = 13.6\,\text{eV}$ the Rydberg energy. The three transition dependent constants are defined by*:

$$\mathcal{A}_{pq} = \frac{Ry}{E_{pq}} f_{pq}, \tag{4.15}$$

* Note that $\mathcal{A}_{pq}$ and $\mathcal{B}_{pq}$ are not to be confused with the Einstein coefficients that are employed in radiation transport, see Eqn. (4.45).



(a) *Excitation* (4.14). **(b)** *Ionization* (4.22).

Figure 4.2. *Cross section functions by Vriens and Smeets.*

with f_{pq} the absorption oscillator strength,

$$\mathcal{B}_{pq} = \frac{g_q}{g_{\text{sub}}} \frac{2Ry^2}{\mathsf{p}_q^3} \left(\frac{1}{E_{pq}^2} + \frac{4}{3} \frac{E_{p+}}{E_{pq}^3} + b_p \frac{E_{p+}^2}{E_{pq}^4} \right), \tag{4.16}$$

with

$$b_p = \frac{1.4 \ln \mathsf{p}_p}{\mathsf{p}_p} - \frac{0.7}{\mathsf{p}_p} - \frac{0.51}{\mathsf{p}_p^2} + \frac{1.16}{\mathsf{p}_p^3} - \frac{0.55}{\mathsf{p}_p^4}, \tag{4.17}$$

$$\delta_{pq} = \exp\left(-\frac{\mathcal{B}_{pq}}{\mathcal{A}_{pq}} \right) - 0.4 \frac{E_{pq}}{Ry}, \tag{4.18}$$

and

$$\gamma_{pq} = Ry \left[8 + 23 \left(\frac{\mathsf{p}_q - \mathsf{p}_p}{\mathsf{p}_p} \right)^2 \right] \times \left(8 + 1.1 \mathsf{p}_q |\mathsf{p}_q - \mathsf{p}_p| + \frac{0.8}{(\mathsf{p}_q - \mathsf{p}_p)^2} + 0.4 \frac{\mathsf{p}_q^{3/2}}{\sqrt{\mathsf{p}_q - \mathsf{p}_p}} |\mathsf{p}_q - \mathsf{p}_p - 1| \right)^{-1}. \tag{4.19}$$

In these equations p_p and p_q are the effective principal quantum numbers:

$$\mathsf{p}_p = Z \sqrt{\frac{Ry}{E_{p+}}}, \tag{4.20}$$

with Z the charge number of the core and E_{p+} the ionization potential of level p. The ion ground level is denoted by "+". By incorporating Z it is possible, in principle, to use the Vriens-Smeets expression for ionic systems, in which case the "+" denotes the ground level of the subsequent ionic level with core charge number $Z + 1$.

It should be noted that the extra weight factor g_q/g_{sub} in Eqn. (4.16) is not present in the original version as given in [30]. It was added in Benoy [31] to extend the applicability of the original formula to non-hydrogen atoms. In non-hydrogen atoms the (principal) quantum levels are split up into multiple levels. Because of this splitting the cross sections are weighted such that the sum of all factors g/g_{sub} for transitions from a certain level p to a certain higher level equals unity. If levels p and q have the same principal quantum number then g_{sub} is equal to g, otherwise g_{sub} equals the sum of the weights of all levels with the same principal quantum number q, to which excitation from p is possible.

A cross section of this kind is defined by two parameters: the weight g_{sub} and the radiative transition probability $A(q,p)$. The latter is required to calculate the oscillator strength (used in Eqn. (4.15)), using:

$$f_{pq} = \frac{m_e c^3 \varepsilon_0 h^2 g_q}{2\pi e^2 E_{pq}^2 g_p} A(q,p), \tag{4.21}$$

with m_e the electron mass, c the speed of light in vacuum, ε_0 the permittivity of free space, h the Planck constant, and e the elementary charge. Optionally, a hydrogen approximation by Johnson [32] can be used for the oscillator strength.

Additionally, Vriens and Smeets suggest a cross section for ionization from excited atoms and ground state alkali atoms:

$$\sigma_{p+}(E) = 4\pi a_0^2 \frac{Ry^2}{E + \alpha E_{p+}} \left(\frac{5}{3E_{p+}} - \frac{1}{E} - \frac{2E_{P+}}{3E^2} \right), \tag{4.22}$$

with $\alpha = 3.25$.

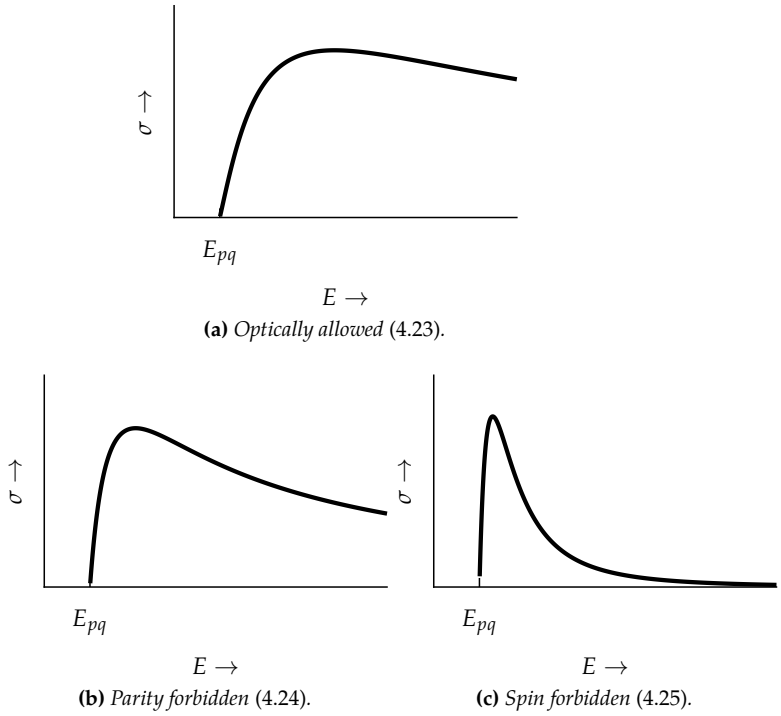
Plots of these two cross section functions are shown in Figure 4.2. Cross sections of this type were used by Benoy [33] for constructing a CRM of Ar, and by Van der Heijden [34] for H cross sections in a CRM of a TALIF experiment in a Ar-H plasma.

Drawin

Three types of semi-empirical cross sections (see Figure 4.3) were proposed by Drawin [35]. These are based on extensions of the Bethe formulae [36], and contain parameters for optically allowed, parity forbidden, and spin forbidden transitions between excited states. Parameters for Ar were calculated by Kimura et al. [37], whose results are the basis of the model by Vlček [38]. They are also used in this thesis in chapters 7 and 8, where time dependent CRMs of Ar plasmas will be presented.

For optically allowed transitions ($\Delta l = \pm 1, \Delta J = 0, \pm 1$ but not $J = 0 \rightarrow J = 0$) the expression is:

$$\sigma_{p1}^A = 4\pi a_0^2 \left(\frac{Ry}{E_{pq}} \right)^2 f_{pq}\, \alpha_{pq} \left(1 - \frac{E_{pq}}{E} \right) \frac{E_{pq}}{E} \ln \left(1.25 \beta_{pq} \frac{E}{E_{pq}} \right), \tag{4.23}$$



(a) *Optically allowed* (4.23).

(b) *Parity forbidden* (4.24).

(c) *Spin forbidden* (4.25).

Figure 4.3. *Semi empirical cross section functions by Drawin.*

where f_{pq} is related to the radiation probability via Eqn. (4.21). For parity forbidden transitions ($\Delta l \neq \pm 1$, $\Delta s = 0$) the cross section reads:

$$\sigma_{pq}^{P}(E) = 4\pi a_0^2 \, Q_{pq}^{P} \left(1 - \frac{E_{pq}}{E}\right) \frac{E_{pq}}{E}, \tag{4.24}$$

and for spin forbidden transitions ($\Delta s \neq 0$):

$$\sigma_{pq}^{S}(E) = 4\pi a_0^2 \, Q_{pq}^{S} \left(1 - \frac{E_{pq}^2}{E^2}\right) \frac{E_{pq}^3}{E^3}. \tag{4.25}$$

The coefficients α_{pq}, β_{pq}, Q_{pq}^{P}, and Q_{pq}^{S} are dimensionless fit parameters.

Custom function

Apart from these fit functions many publications use some other type of fit function to represent a range of cross sections using a set of parameters. In order to use these cross sections without much effort, cross sections can also be defined in the form of custom functions with a parameter set of variable size.

In the input file a mathematical expression must be entered describing the custom function. Use of these types of custom functions for cross sections does not require any compilation of C++ code. The functions are evaluated at *run time*, thereby offering a high degree of flexibility (see also section 5.4.3).

Look-up tables

The final option to define a cross section is by using a look-up table. In literature many cross sections are available in this form, be it from experimental results or from complex (quantum mechanical) calculations. The PLASIMO framework provides classes to parse and store this kind of data using simple text files. A high degree of freedom is available in the use of units of measurement, preventing the need for cumbersome conversion of data.

Rate coefficient

In principle, every rate coefficient has to be calculated using equation (4.9), where virtually any cross section can be used in combination with any EEDF. However, an important category is that of plasmas with a Maxwellian EEDF. In that case the EEDF can be described by one parameter: the electron temperature T_e. In literature many rate coefficients for electron excitation can be found written as some (usually exponential) function of T_e, for instance the rate coefficient for excitation of Ar from ground state to 4s from Kannari et al. [39]:

$$K(3p^6, 4s) = 5.0 \times 10^{-15}\, T_e^{0.74}\, \exp(-11.56/T_e), \tag{4.26}$$

with the electron temperature T_e in eV. Instead of defining a cross section for a collisional transition, a rate coefficient function can be defined similar to the *custom function* for the cross sections.

4.3.2 Electron de-excitation

Up to now we have dealt with the endothermic processes of excitation. The rate coefficient for the reverse processes of de-excitation can be determined by starting with the forward process and then employing the principle of detailed balancing [40]. When a collisional excitation transition is defined, an optional switch in the code determines whether the reverse process is implicitly included through detailed balancing. If it is not, the reverse process can be included by defining the reverse process explicitly.

In case of thermodynamic equilibrium the number of transitions in forward direction equals the number of transitions in backward direction:

$$n_e n_p^B K(p, q) = n_e n_q^B K(q, p), \tag{4.27}$$

where n_e denotes the electron density and n_p^B, and n_q^B denote the densities of excited levels p and q according to a Boltzmann distribution:

$$\frac{n_q^B}{g_q} = \frac{n_p^B}{g_p} \exp\left(-\frac{E_{pq}}{k_B T_e}\right), \tag{4.28}$$

with k_B the Boltzmann constant. The forward and backward rate coefficients are then related through:

$$K(q,p) = \frac{g_p}{g_q} K(p,q) \exp\left(\frac{E_{pq}}{k_B T_e}\right). \tag{4.29}$$

This relation is used to calculate the rate coefficient of the reverse process when an electron collisional excitation transition is not defined using a cross section, but using a rate coefficient as function of the electron temperature (in the case of a Maxwellian EEDF).

We can use the principle of detailed balancing on a more elementary level to relate the cross section for the forward (σ_{pq}) and backward process (σ_{qp}). In thermal equilibrium the EEDF is Maxwellian and is described by:

$$f^M(E) = \frac{2\pi}{(\pi k_B T_e)^{3/2}} \sqrt{E} \exp\left(-\frac{E}{k_B T_e}\right). \tag{4.30}$$

When we use this Maxwellian EEDF in the calculation of the rate coefficients (equation (4.9)), but only for an energy between E and $E + dE$, and enter this in (4.27) we get:

$$n_e\, n_p\, \sigma_{pq}(E)\, v_e(E)\, f^M(E)\, dE = \\ n_e\, n_q\, \sigma_{qp}(E - E_{pq})\, v_e(E - E_{pq})\, f^M(E - E_{pq})\, dE. \tag{4.31}$$

When we combine this with (4.28) the result is:

$$\sigma_{qp}(E) = \frac{g_p}{g_q} \frac{E + E_{pq}}{E} \sigma_{pq}(E + E_{pq}). \tag{4.32}$$

Since the cross sections do not depend on the presence of thermodynamic equilibrium, equation (4.32) is valid under all conditions. This result shows that when the cross section for the forward process $\sigma_{pq}(E)$ is known, the cross section for the reverse process is also known. When the EEDF is not Maxwellian, the cross section for the reverse process is used in Eqn. 4.9 to calculate the de-excitation rate.

4.3.3 Ionization and recombination

If the energy of an electron is high enough, a collision with an atom in excited level p can cause ionization:

$$X_p + e + (E_{p+}) \overset{K(p,+)}{\rightarrow} X_+ + e + e. \tag{4.33}$$

Identical to electron collisional excitation the rate coefficient is dependent on a cross section, which is now called the ionization cross section. Its value is calculated using equation (4.9).

The opposite process of recombination can occur in two* ways: two-electron recombination, the reverse of (4.33), or radiative recombination. In case of equilibrium between ionization and two-electron recombination, the Saha relation holds:

$$\frac{n_p}{g_p} = \frac{n_p^S}{g_p} = \frac{n_e}{g_e}\frac{n_+}{g_+}\left(\frac{h^2}{2\pi m_e k_B T_e}\right)^{3/2} \exp\left(\frac{E_{p+}}{kT_e}\right), \tag{4.34}$$

where n_p^S is the Saha density of level p. The rate coefficient for two-electron recombination is then:

$$K(+,p) = K(p,+)\,\frac{g_p}{2g_+}\left(\frac{h^2}{2\pi m_e k_B T_e}\right)^{3/2} \exp\left(\frac{E_{p+}}{k_B T_e}\right), \tag{4.35}$$

where we have used $g_e = 2$.

The second recombination mechanism, radiative recombination, does not require the extra "spectator" electron:

$$X_+ + e \overset{K_{\rm rad}(+,p)}{\rightarrow} X_p + h\nu_{p+}, \tag{4.36}$$

where the excess energy produces a photon. The following expression taken from Van der Mullen [22] is used for the rate coefficient:

$$K_{\rm rad}(+,p) = \frac{n_p^S}{n_e n_+}\gamma\, Z^4\, \frac{1}{\mathsf{p}_p} \int_{\epsilon_p}^{\infty} \frac{\exp(-\epsilon)}{\epsilon}\, h(\mathsf{p}_p, \epsilon/\epsilon_p) d\epsilon, \tag{4.37}$$

where $h(\mathsf{p}, y)$ is the Gaunt factor†, and

$$\epsilon_p = \frac{E_{p+}}{k_B T_e}, \tag{4.38}$$

and

$$\gamma = \frac{8\,\alpha^4\, c}{3\pi\,\sqrt{3}\,a_0}, \tag{4.39}$$

with

$$\alpha = \frac{e^2}{4\,\pi\,\varepsilon_0\,\hbar\,c}. \tag{4.40}$$

* In the implementation described here, two-electron and radiative recombination are included. Molecular Assisted Recombination, described in section 4.3.8, is a third mechanism, but it is not included in this code.

† The Gaunt factor is approximately 1.

The CRM code provides two ways to include ionization in a model. It can be explicitly defined for every relevant level, or a "default" ionization contribution can be used in which case the ionization is included for every defined level using the cross section for ionization by Vriens and Smeets (4.22). Whether two-electron recombination and radiative recombination are included in the model, is also controlled by global options. When either of them is enabled a frequency is added to the relevant element of the transition-matrix for every level that is included in the model.

4.3.4 Radiative transitions

Transitions between excited levels can also occur due to emission and absorption of radiation. Three processes can be distinguished between an upper (u) and lower level (l):
spontaneous emission:

$$X_u \overset{A(u,l)}{\rightarrow} X_l + h\nu_{ul}, \tag{4.41}$$

absorption:

$$X_l + h\nu_{ul} \overset{\rho_\nu B(l,u)}{\rightarrow} X_u, \tag{4.42}$$

and stimulated emission:

$$X_u + h\nu_{ul} \overset{\rho_\nu B(u,l)}{\rightarrow} X_l + h\nu_{ul} + h\nu_{ul}, \tag{4.43}$$

where $\nu_{ul} = c/\lambda_{ul}$ is the photon frequency, ρ_ν the spectral energy density, whereas $A(u,l)$, $B(l,u)$, and $B(u,l)$ are the Einstein coefficients for spontaneous emission, absorption, and stimulated emission, respectively.

In the framework of a CRM, and more in general of a zero-dimensional model, absorption and stimulated emission are problematic because of their non-local nature: one part of the plasma produces radiation which interacts with another part of the plasma. This problem is tackled by the use of an *escape factor* θ_{ul}, which is used to replace the transition probability $A(u,l)$ in equation (4.41) by (see Holstein [41]):

$$A^*(u,l) = \theta_{ul} A(u,l). \tag{4.44}$$

The aim of the escape factor is to approximate the influence of absorption and stimulated emission, and is determined by integration over the line profile as previously shown in Eqn. (3.75):

$$\theta_{ul} = 1 - \frac{1}{n_u A(u,l)} \left(n_l B(l,u) - n_u B(u,l) \right) \rho_\nu, \tag{4.45}$$

Depending on the dominant radiative processes, five situations can be identified as shown in table 4.1.

$\theta > 1$	Stimulated emission dominates.
$\theta = 1$	Absorption and stimulated emission can be neglected.
$0 < \theta < 1$	Partial absorption.
$\theta = 0$	No net radiative transfer.
$\theta < 0$	Absorption exceeds local emission.

Table 4.1. *Relation between escape factor values and dominating radiative processes.*

Since escape factors are so heavily dependent on the geometry of the plasma, they can not be calculated by the CRM. Instead they are considered input data. Therefore, in the CRMs a radiative transition is defined by the spontaneous emission probability and an escape factor. In section 3.8 a case was described where the escape factors were dependent on the location in the plasma, requiring an iterative solution procedure.

4.3.5 Cut-off procedure

The number of levels in an atom that have to be dealt with can be very large, but is limited. In reasonable approximation, the radius of an atom excited to a level p with principal quantum number p_p and core charge number Z is:

$$a_p = \frac{a_0 \mathsf{p}_p^2}{Z}. \tag{4.46}$$

When we compare this with the distance between electrons for a given density:

$$d_{ei} = n_e^{-1/3}, \tag{4.47}$$

we see that the maximum excited level has the principal quantum number [42]:

$$\mathsf{p}_{\max} = \sqrt{\frac{Z}{a_0\, n_e^{1/3}}}\,. \tag{4.48}$$

For a density of $n_e = 10^{18}\,\mathrm{m}^{-3}$ and $Z = 1$ it results in $\mathsf{p}_{\max} \approx 140$.

Another approach is to equate the atom radius to the Debye length [43]:

$$\mathsf{p}_{\max} = \sqrt{\frac{Z}{a_0}\sqrt{\frac{\varepsilon_0 k_B T_e}{n_e e^2}}}. \tag{4.49}$$

For equal Z and n_e, and $T_e = 1\,\mathrm{eV}$ this results in $\mathsf{p}_{\max} \approx 390$.

For practical reasons it is necessary to somehow limit the amount of levels that are taken into account. The obvious solution of simply neglecting higher

levels can, however, lead to inaccurate results, since the stepwise ionization along those excluded higher levels can play a large role. A more elegant procedure was introduced by Van der Mullen [22]. In this cut-off procedure a highest excited level, the cut-off level, is chosen, so that the cut-off level and all levels above it are collisionally dominated, and in the so-called *hot region*, which means that the average electron energy is higher than the ionization energy of those levels. The levels in this region obey the Excitation Saturation Balance (ESB), meaning that the atoms are repetitively excited until the excited electrons reach the continuum and the atom is ionized.

This process is taken into account by an additional ionization contribution to the rate coefficients for excitations from levels below the cut-off level to levels above the cut-off level:

$$K^*(p,+) = K(p,q)\left(1 - \frac{\mathrm{p}_p^6}{\mathrm{p}_q^6}\right). \tag{4.50}$$

Since the levels above the cut-off level are no longer included in the model, this rate coefficient is no longer attributed to the transition from p to q but from p to $+$.

To determine the cut-off level, Eqn. (3.34) can be used.

Whether this procedure is used in the CRM is determined by a global option that defines the cut-off level, by using the name of the intended level.

4.3.6 Optimization

The rate coefficient for collisional transitions is calculated by using equation (4.9). This calculation is performed by numerical integration using the trapezoidal rule, a "one size fits all" solution that can handle non-smooth cross sections and/or EEDFs. The CRM supports any form of EEDF, which is demonstrated in chapter 7 where an EEDF that is strongly peaked around 76 eV is used.

Because the numerical integration is relatively computationally expensive, the rate calculation is of special interest. In many plasmas the EEDF is, or can be considered to be, Maxwellian, so by introducing the dimensionless parameter $x = E/k_B T_e$ equation (4.9) becomes:

$$K(p,q) = \sqrt{\frac{8k_B T_e}{\pi m_e}} \int_{E_{pq}/k_B T_e}^{\infty} \sigma(x k_B T_e)\, x \exp(-x)\, dx, \tag{4.51}$$

After re-parameterization this leads to:

$$K(p,q) = \sqrt{\frac{8k_B T_e}{\pi m_e}} \exp(-x_0) \int_0^{\infty} h(x' + x_0) \exp(-x')\, dx', \tag{4.52}$$

with:

$$h(x) = x\,\sigma(x k_B T_e), \tag{4.53}$$

and

$$x_0 = \frac{E_{pq}}{k_B T_e}. \tag{4.54}$$

The integral in (4.52) can efficiently be calculated using the n-point Gauss-Laguerre quadrature [44, eq. 25.4.45]:

$$\int_0^\infty h(x)\exp(-x)\,dx \doteq \sum_{i=1}^{n} w_i h(x_i), \tag{4.55}$$

with roots x_i and corresponding weights w_i. Since this method prevents the expensive evaluation of the exponential term* it is much more efficient.

Further optimization can be achieved in case an analytical solution is available for equation (4.51). For instance, when the cross section function is a step function (4.10), the rate coefficient is:

$$K(p,q) = \sqrt{\frac{8k_B T_e}{\pi m_e}}\,\sigma_{pq}\,(1+x_0)\,\exp(-x_0), \tag{4.56}$$

so no form of numerical integration is necessary. Section 5.4.4 describes how these optimizations are implemented in a transparent way.

4.3.7 Matrix composition

The processes described so far constitute the transition frequency matrix $\boldsymbol{F}$ with elements:

$$F_{qp} = \begin{cases} -\left[\sum_{i\neq p} n_e K(p,i) + \sum_{i<p} A^*(p,i)\right] & \text{for } p = q \\ n_e K(p,q) + A^*(p,q) & \text{for } p > q \\ n_e K(p,q) & \text{for } p < q,\ q = + \\ n_e^2 K(p,q) + n_e K_{\text{rad}}(p,q) & \text{for } p = + \\ -\left[\sum_{i<+} n_e^2 K(p,i) + \sum_{i<+} n_e K_{\text{rad}}(p,i)\right] & \text{for } p = q = + \end{cases} \tag{4.57}$$

Note that this means that the sum of every column equals 0.

The transition matrix will be built up in the following steps:

*The cross section function must be sufficiently smooth to achieve accurate results.

1. We start with a zero matrix.
2. If default ionization is enabled, elements F_{+p} for all p are filled with the default ionization frequency, using the cross section by Vriens and Smeets 4.22.
3. All collisional transitions are added, either from a cross section or a rate coefficient. However, there are several options:
   - If the cut-off procedure is employed and the upper level is above the cut-off level, the frequency $K^*(p,+)$, see equation 4.50, is added to element F_{+p}.
   - If the cut-off procedure is not relevant for the transition (i.e., both the lower and upper levels are below the cutoff level) the transition frequency can either replace or be added to the already present frequency in element F_{qp}. This is dependent on an *override* switch, which is present in the definition of every collisional transition. This can be useful when certain default ionization rates are to be overruled by specific rates.
   - A second switch determines whether *detailed balancing* is used, so whether the frequency for the reverse process is added to element F_{pq}. The *override* switch will also be in effect for the reverse process.
4. At this stage the frequencies for ionization are present in $\boldsymbol{F}$, so now (if it is included) the two-electron recombination frequency can be set for F_{p+}, see equation 4.35.
5. If radiative recombination is included, its frequency, see equation 4.37, is added to F_{p+}.
6. All radiative transitions are added to the frequency matrix, by adding the frequency $A^*(p,q)$, see Eqn. (4.44), to F_{qp}.
7. In the last step the diagonal elements are calculated by summing every column: $F_{pp} = -\sum_{q \neq p} F_{qp}$.

4.3.8 Additional processes

In section 3.7 a method was described to extend the source vector of the system of species density balances with an additional extra source vector (Eqn. (3.68)). This extension enables the possibility to include certain heavy particle processes into the CRM.

In chapter 8 this is used in a time dependent CRM for LIF experiments in Ar plasmas to include (de-)excitation processes by the Ar ground state atoms. The extra source vector is constructed from rate coefficients and added to the transition matrix.

A process that must be mentioned, but is not implemented in the current version of the CRM, is Molecular Assisted Recombination (MAR). In this process ions recombine in two consecutive steps to an excited state, for instance in an Ar plasma:

$$Ar^+ + Ar + Ar \overset{K_{\mathrm{MAR},1}}{\rightarrow} Ar_2^+ + Ar, \tag{4.58}$$

followed by:

$$Ar_2^+ + e \overset{K_{\mathrm{MAR},2}}{\rightarrow} Ar^* + Ar + (E). \tag{4.59}$$

The second process is so quick that the first process is the limiting step and the process can be described by $K_{\mathrm{MAR},1}$. Effectively, this is comparable to a radiative recombination reaction:

$$Ar^+ + e \overset{K_{\mathrm{MAR}}}{\rightarrow} Ar^* + (E), \tag{4.60}$$

with source term $\mathcal{S}_{\mathrm{MAR}} = n_i n_e K_{\mathrm{MAR}}$.

4.4 Conclusion

Once the transition matrix is constructed the solution procedures as described at the start of section 4.2 can be employed. Now that we know how the transition matrix is assembled and solved, we can start with the treatment of the implementation details in the PLASIMO framework.

Chapter 5

Implementation of a CRM code in the plasimo framework

5.1 Introduction

In the previous chapter the basic functionality of the CRM code was described. The aim is to implement this functionality as a model plug-in for the plasimo framework. In this chapter several specifics of this implementation will be treated. We will start with a short overview of the history of plasimo (section 5.2). Since plasimo is developed in C++, it is necessary to give some background about this programming language, and how its facilities are used in plasimo. We will therefore continue with an introduction to C++ and some constructs as they are used in plasimo (section 5.3). Once the relevant C++ aspects have been described, the structure of the CRM model plug-in will be treated, along with the construction of the transition-matrix (section 5.4). This will also give some insight into the structure of plasimo itself. Section 5.5 is devoted to the implementation of the solution procedure.

5.2 History of plasimo

plasimo, an acronym that stands for PLAsma SImulation MOdel, is the name of the plasma modeling framework that has been under continuous development in the plasma groups at the faculty of applied physics of the Eindhoven University of Technology since the 90's of the previous century.

Before plasimo came into existence, a lot of expertise was already present in the field of CRMs, see for example Van der Sijden and Pots [45], Van der Mullen [25], and Hartgers [46], on which the CRM implementation in this work is based. The main goal of these models was the interpretation of spectroscopy.

The desire rose to construct more complex plasma models covering the three plasma aspects *configuration*, *transport* and *chemistry*: a "Grand Model". This undertaking was started by Emile de Jong, Dany Benoy, and Frank Fey under guidance of Joost van der Mullen. Originally, the *Pascal* programming language rather than the more commonly used *Fortran* was chosen for this task, since Pascal supports a more modular approach. From the beginning, modularity has been a key concern to the plasimo-team.

The original Pascal code was able to model two different plasmas: an atmospheric Inductively Coupled Plasma (ICP) in Ar and a cascaded arc; a DC driven plasma source. The code implemented generalized ortho-curvilinear coordinates to support geometrically pinched configurations.

plasimo first came into existence in 1992 as a re-implementation of this code in the C programming language by Dany Benoy. This first version, the basis of Benoy's thesis [31], was also used and extended by Ger Janssen (awarded with a PhD degree in 2000) for his thesis [47]. In addition to Ar (atomic) chemistry, he explored molecular chemistry in the form of H_2 and O_2 plasmas. Furthermore, Janssen studied a Microwave Induced Plasma (MIP) system as used in the manufacturing of optical fibers.

To achieve an even greater degree of modularity than is practical in C, plasimo was then re-implemented in the C++ programming language by Jan van Dijk in close cooperation with Harm van der Heijden, Bart Hartgers, and Kurt Garloff. The object-oriented approach of C++ makes it more easy to reuse and extend existing pieces of code. In an effort to make the control of plasimo models more user friendly, thereby also increasing its usability in an industrial environment, a Graphical User Interface (GUI) was added. This modeling platform that enables a user to conveniently manipulate the hierarchical structure of the model, controls the calculation, and provides facilities for monitoring the running calculation, is the current form of plasimo.

The C++ version of plasimo was used by Van Dijk for his thesis (2001, [27]) to study a QL lamp manufactured by Philips. Van der Heijden (thesis 2003, [3]) used the platform for his study of radiation transport in a sulfur lamp and Hartgers (2003, [48]) for the time dependent modeling of the plasmas in fluorescence lamps. Others who made extensive use of plasimo and extended its functionality, were Colin Johnston for the study of sulfur lamps (thesis 2003, [4]), Bart Broks for plasmas for laser wakefield acceleration (thesis 2006, [49]), Mark Beks for elemental diffusion in HID lamps (thesis 2008, [50]), Michiel van den Donker for deposition using microwave plasmas (thesis 2008, [14]), and Manuel Jiménez for microwave induced plasmas (thesis 2011, [13]).

Starting in 1997, during the development of plasimo, another modeling code was being developed in the EPG (Elementary Processes in Gas discharges) group of the applied physics department of the TU/e by Gerjan Hagelaar under supervision of Frits de Hoog. This code, called md2d for Micro-Discharge 2-Dimensional and the subject of his thesis (2000, [2]), was specifically aimed at modeling plasma-electrode interactions. In contrast to plasimo this code does not include flow and only supports rectangular grids. Furthermore, because md2d has to deal with space charges it is equipped with a Poisson equation solver.

When Van der Mullen moved from the ETP (Equilibrium and Transport in Plasmas) group to the EPG group, elements of the plasimo platform were introduced in md2d. md2d was rewritten in C++ by Van Dijk (Post Doc at the time) and PhD student Wouter Brok, and equipped with the same GUI as plasimo. One of the topics in Brok's thesis (2005, [51]) dealt with the modeling of plasma ignition. The md2d code (in C++) was extended and used by Diana Mihailova (thesis 2010, [52]) for her research on sputtering hollow cathode discharges.

In addition to fluid modeling, Brok and Van Dijk constructed the basis for the Monte Carlo module in plasimo. This Monte Carlo code was used by Marc van der Velden (thesis 2008, [53]) in his study of radiation generated plasmas. The results of this modeling study, with some adjustments, are also used in this thesis. In chapter 7 a CRM of Extreme UltraViolet (EUV) driven plasmas will be presented. The goal of the CRM is the interpretation of time resolved spectroscopical measurements, and uses plasma parameters (electron density and energy distribution function) provided by the Monte Carlo code. Thus, from the roots of plasimo, lying in the field of CRMs, we have come full circle.

Since the translation of md2d in C++, it has been gradually merged with plasimo, thereby creating a platform for constructing a wide range of models of many different types of plasmas: LTE or non-LTE, steady-state or transient, flowing or non-flowing, with or without space charges. Different transport mechanisms can be applied, be it in the form of particle driven (Monte Carlo), fluid, or ruled by ray-tracing. The platform offers a user friendly interface through the GUI, but also through the use of structured, descriptive input files. Chemistry, geometrical configuration and input power can easily be defined and manipulated.

One of the key features of the plasimo framework is its modularity: a complex model is constructed from building blocks, some of which are also used in other models. In this work a general CRM code is presented that makes use of several of these building blocks. The code is based on a previous CRM code by Hartgers [46] that was suitable for Quasi Steady State (QSS) calculations. Apart from an implementation of this code in the plasimo framework, it is extended to perform transient simulations through the use of time dependent electron densities and electron energy distribution functions (EEDF). This also enables the simulation of laser induced fluorescence experiments [54], similar to TALIF simulations by Van der Heijden [34], which will be the subject of chapter 8). Details of this

implementation will be described in the following sections.

In chapter 6 a more general zero dimensional modeling code will be presented that is also implemented in the plasimo framework. This code does not focus on the radiative and collisional processes within an atomic system as does the CRM code, but on time dependent complex chemistry combined with external power input. This code for constructing Global Plasma Models (see chapter 2) is based on the *RateLab* model by Jiménez [13], which in turn was based on *PyRate* by Van den Donker [14].

5.3 C++: C with classes

The CRM is implemented as a model plug-in using the plasimo framework, written in the C++ programming language. C++ was developed by Bjarne Stroustrup starting in 1979 as an enhancement of the C programming language. The language was originally called *C with classes*, demonstrating the fact that the enhancements are in the form of *classes*, allowing for *object oriented programming* (OOP). The object oriented approach allows for a strong modular design, which has since its incarnation been one of the main concerns of plasimo; classes are used on all levels in plasimo, from the most basic level, i.e., the models themselves, to elements like solvers, species, and many more. The flexibility of plasimo is further increased by the use of a *plug-in* system, i.e., executable code that is dynamically loaded when necessary.

To demonstrate the benefits of classes we will show an implementation of a code for dealing with cross sections. In the examples only a simple step cross section (see Eqn. (4.10))* will be treated so we can focus on C++ specifics. In gradual steps we will introduce concepts that are used throughout plasimo, most notably *classes*, *polymorphism*, and *plug-ins*. Most C++ specific elements will be explained in the text, for some more background information the reader is referred to appendix A.

5.3.1 A simple cross section code

The most straightforward way of applying a cross section somewhere in a program is by simply placing the code at the required position as shown in listing 5.1. While this code is certainly valid and does what it is intended to do, it is far from practical. In a larger program the cross section might be needed at multiple different locations in the program. In this simple example the cross section is calculated in lines 15 through 20 using an `if-else` construct, and in line 24 using a *ternary*

* All step function cross sections in this chapter will have a threshold of 10 eV and step height of 10^{-20} m^2.

```
1  // reserve variable for threshold and initialize to 10 eV
2  double threshold = 10;
3
4  // reserve variable for step function height and
5  // initialize to 1e-20 m^2
6  double stepheight = 1e-20;
7
8  // reserve variable for cross section
9  double cs_value;
10
11 // reserve variable for energy and initialize to 20 eV
12 double energy = 20;
13
14 // get the cross for the energy of 20 eV
15 if (energy > threshold) {
16   cs_value = stepheight;
17 }
18 else {
19   cs_value = 0.0;
20 }
21
22 // reserve another variable for a cross section and
23 // use ternary operator for initialization
24 double cs_value_other = (energy > threshold) ? stepheight : 0.0;
```

Listing 5.1. *Code snippet showing the calculation of two cross section values when the cross section is described by a step function (see Eqn.* (4.10)*), with the threshold at 10 eV and a height of 10^{-20} m^2. Note, that no physical units are used. In this case the energy is given in electronvolt and the cross section in square meter. Two consecutive forward slashes "//" denote a comment. Everything starting from the slashes till the end of the line is ignored by the compiler. As a convention comments are either on the same line as a statement or the line(s) preceding a statement. More information about the definition and initialization of variables can be found in appendix A.*

operator*. Anywhere, where this cross section would be needed this code would have to be copied, making extension and maintenance of the code rather laborious.

An improvement can be made by implementing the step cross section as a *function*†, as shown in listing 5.2. This implementation of a cross section as a function

* A ternary operator is a compact form of an `if-else` construct. The statement between equal-sign and question mark is evaluated and if it is true (so it must represent a boolean value) the value between question mark and colon is returned, otherwise the statement (here, a value) following the colon.

† More information about functions can be found in appendix A. The most important features of a function are that it can be recognized by the parentheses (both in definition and call sequence) to hold

```
// definition of the cross section function
double CrossSectionStep(double e) {
  double threshold(10.0);
  double stepheight(1e-20);
  double cs_result = 0.0;
  if (e > threshold) {
    cs_result = stepheight;
  }
  return cs_result;
}

...

// section of code where the function is called returning
// the cross section value

// energy for which we want the cross section
double energy = 20; // at 20 eV
double cs_value = CrossSectionStep(energy);
```

Listing 5.2. *Code snippet with the same functionality as listing 5.1, but here the cross section calculation is moved to a separate function. More information about functions can be found in appendix A.*

is already much more convenient; whenever the cross section is needed only the function has to be called. Maintenance is also simplified since only the code in one location (the cross section function) is affected. For cross sections more complex than this simple step function the benefits can be quite substantial.

In general, though, different cross sections will be needed; cross sections with different thresholds and different heights. It is possible to define a different function for each of these cross sections, or to extend the cross section function with additional parameters; one for the threshold and one for the height. A better way is to utilize the main feature of C++: classes.

5.3.2 A simple class

A class is a construct used in object-oriented programming that offers a combination of data storage and functionality. Data can be stored in so called *member variables* (or simply *members*) and the functionality is provided by *methods* or *member functions*. Classes are variable types just like the built-in types of C++, such as `int` and `double`*. A class is only the *description* of the construct, an actual *instantia-*

the arguments, and that it returns a value (if the return type is `void` it effectively does not return a value).

* Since classes are types they can also be used as arguments and return value of functions.

tion of a class is called an *object*. The data members of a class can be any data type including other classes. The member functions are subroutines that offer some kind of functionality, usually involving manipulation of its data members. Just like normal functions in C++, member functions can have arguments (between parentheses) and always have a return value.

Access to the members (data and functions) of a class can be regulated by the use of *access specifiers*. The access specifier `public` means that whenever access to the object is available, access to its members is also available. Members can also be `private`, meaning that they are only accessible to other members of the same class.

Listing 5.3 shows a simple class for the step function cross section. The threshold and height are both stored as private data members* in lines 30 and 31. The class holds several methods (recognizable by the parentheses), three of which require more explanation:

- `CrossSectionStep(...)`
- `~CrossSectionStep()`
- `operator()(...)`

Every class has two special member functions: the *constructor* and the *destructor*. As their names imply, they are called when an instance of the class is constructed and destructed. They can be recognized by the fact that they have the same name as the class itself, and that they do not have a return type. The destructor carries the additional prefix "`~`".

In the case of the step cross section, the constructor (line 7) requires two input values: the threshold and the height (or value) of the cross section. The constructor stores these values in its appropriate members so that they can be accessed at a later time by other member functions. The task of the destructor (line 12) is to make sure that no "loose ends" will remain when an instance is destroyed. This usually means that dynamically allocated memory, allocated by the constructor or other member functions, is released.

The third special function is in line 15. This is an example of *operator overloading*: the ability to define the behavior of operators on classes. Since classes are types just like the built-in types, we can use operators on them. The behavior of operators on built-in types is defined in the language definition. For example, when the addition operator "`+`" is applied to two variables of type `double` the result is a `double` holding the sum of the two initial `doubles`. However, it is up to the programmer to define what the behavior is of an operator when it is applied to a class. Every combination of operator and class that is used in a program must

*Commonly, the names of the variables containing member data are prefixed with "`m_`" to emphasize that they are *member* variables.

```
1  // name of the class, this defines a new type
2  class CrossSectionStep {
3    // members that are publicly accessible
4    public:
5      // the constructor, accepting two values,
6      // that define the step function
7      CrossSectionStep(double threshold, double cs) {
8        m_threshold = threshold;
9        m_cs = cs;
10     }
11     // the destructor, that does not have to do anything
12     ~CrossSectionStep() { }
13     // function call operator, that accepts an energy as input
14     // and returns a double value (the cross section)
15     double operator()(double eps) {
16       // ternary operator
17       return (eps < threshold()) ? 0.0 : sigma_above_th(eps);
18     }
19     // function returning the threshold level
20     double threshold() {
21       return m_threshold;
22     }
23     // function returning the cross section above
24     // the threshold level
25     double sigma_above_th(double eps) {
26       return m_cs;
27     }
28   // members that are not publicly accessible
29   private:
30     double m_threshold;
31     double m_cs;
32 };
```

Listing 5.3. *Code snippet showing a class for a step function cross section. The private member variables are initialized in the body of the constructor, lines 8 and 9.*

```
1 double step_thresh = 10.0; // in eV
2 double step_height = 1e-20; // in m^2
3 // definition of a CrossSectionStep object, given the two
4 // previously defined values
5 CrossSectionStep my_cs_step(step_thresh, step_height);
6 // call the function operator, returning the cross section value,
7 // i.e., 1e-20 m^2 at 20 eV
8 double cs_val = my_cs_step(20);
```

Listing 5.4. *Code snippet showing an example of the use of the* `CrossSectionStep` *class as defined in listing 5.3.*

be defined*. In the step cross section class the behavior of the *function call* operator is defined, this means that an object can be used as if it is a function, see line 8 in listing 5.4.

The other two member functions of the `CrossSectionStep` class (lines 20 and 25) behave like normal functions. The `threshold()` function returns the value of the private variable†, and the `sigma_above_th()` function returns the cross section value for energies exceeding the threshold energy; in this case simply returning the height of the step function.

Listing 5.4 shows how the class can be used. First, an instance of a step cross section of height $10^{-20}\,\mathrm{m}^2$ at a threshold of 10 eV is declared. Then the cross section at 20 eV is requested from the cross section object. When the function `operator()` is called the code first checks whether the input energy is above the threshold energy. If it is above threshold the function for returning the actual cross section value is called, otherwise 0.0 is returned (line 17).

The advantage of using a class for providing the functionality of calculating a cross section over a normal function is that a class maintains *state*, i.e., a class can store data that can be used to influence its behavior. Here, the threshold and height of the cross section are stored in the object upon declaration, and determine the returned cross section value when it is called.

To implement all the cross section types that have been discussed in section 4.3, one could follow the procedure in which for each type a different class is defined. Only the constructor and the implementation of the `sigma_above_th()` function would have to be changed. The downside of this approach is that it is very impractical to use a collection of these cross sections: since every class of a cross section is of a different type, they can not be accessed through a common interface, even though they look so much alike. This can be solved by using class hierarchy, a combination of *base* and *derived* classes, and the concept of *polymorphism*.

* More information about operator overloading can be found in appendix A.

† In this way *public* access is available to a *private* variable.

5.3.3 Class derivation and polymorphism

OOP offers the possibility to derive a class from another; a so called *parent* class. The result is that the derived class *inherits* nearly all members and methods* of the parent class. Thus the members and methods of the parent class are added to the specific members and methods of the derived class. The result is that all classes, derived from the same parent class share the parent's class interface. A derived class can override member functions of the parent class upon implementation. This is called *polymorphism*.

In listing 5.5 an example is given of class derivation. It is used to define the step function cross section as a derivative of a general parent cross section class. The first part of the listing describes the base class `CrossSectionBase`, the second gives the derived class `CrossSectionStep`. The class `CrossSectionBase` provides the basic functionality a cross section class should have: to return the threshold energy and the cross section for the desired energy. Whereas previously in listing 5.3 the method `sigma_above_th()` implemented the calculation of the cross section; in case of a step function simply returning the height. Here, another function is called: `do_sigma_above_th()`.

The first notable thing of the declaration of the `do_sigma_above_th()` function is the keyword *virtual*, meaning that this function can be overridden by a function of a derived class. The second thing is that the function is appended with "`= 0`". This means that the function is *pure virtual* so derived classes *must* contain an implementation of this specific function†. The pure virtual function is defined in a `protected` section, which is similar to `private` with the addition that derived classes also have access to members defined in this domain.

The `CrossSectionStep` class is derived from the base class as is shown by its declaration in line 29‡. The derived class also has a constructor (like any class) and implements the `do_sigma_above_th()` function.

Before the body of its constructor is executed, the constructor of the base class is invoked. In contrast to the previous implementation of the step cross section (listing 5.3), a single variable is used for the input parameter of the constructors. Rather than a threshold and step height value, a single variable of type `plNode` is passed to the constructor, that can be seen as a data container. The threshold and step height value are extracted from this single variable; the step height in the derived class (line 36), the threshold in the base class (line 7). The reason behind using a `plNode` data structure is that the base class is used by different derived cross section classes, that each require a different set of parameters for their initialization. In plasimo the `plNode` class is used for storage of, and access

*Except the constructor, destructor, and `operator=()` member function.

†It also means that the base class is *abstract*; i.e. there can be no instance of the class. If an instance would be allowed this would pose a problem since it is not known what should be done by a call to the pure virtual function.

‡Parent classes are denoted by a colon ("`:`").

```
// the base class
class CrossSectionBase {
  public:
    // the base class constructor, loading a plNode object from
    // which the threshold is extracted and stored
    CrossSectionBase(plNode node) {
      m_threshold = node("threshold");
    }
    ~CrossSectionBase() { }
    double operator()(double eps) {
      return (eps < threshold()) ? 0.0 : sigma_above_th(eps);
    }
    double threshold() {
      return m_threshold;
    }
    double sigma_above_th(double eps) {
      return do_sigma_above_th(eps);
    }
  // members only accessible by derived classes
  protected:
    // function returning cross section value is pure virtual
    // must be implemented in derived class
    virtual double do_sigma_above_th(double eps) = 0;
  private:
    double m_threshold;
};

// the derived class, colon indicates the parent class
class CrossSectionStep : CrossSectionBase {
  public:
    // constructor first calls constructor of base class...
    CrossSectionStep(plNode node)
     : CrossSectionBase(node) {
      // ...and then extracts the step function height from
      // the plNode object and stores it
      m_cs = node("cs");
    }
  // the virtual function must be implemented as protected
  protected:
    // implementation of the pure virtual function
    virtual double do_sigma_above_th(double eps) {
      return m_cs;
    }
  private:
    double m_cs;
};
```

Listing 5.5. *Code snippet showing a cross section base class and a derived class implementing a step function.*

```
1 // create step cross section object
2 CrossSectionStep my_cs_step(cs_node);
3 // create pointer of base type pointing to object of derived type
4 CrossSectionBase* p_cs = &my_cs_step;
5 // call function operator via pointer to base class
6 // cs_val = 1e-20 m^2 at 20 eV
7 double cs_val = (*p_cs)(20);
```

Listing 5.6. *Code snippet showing an example usage of the* `CrossSectionStep` *class and its base class as defined in listing 5.5. Previous to the code shown here a variable* `cs_node` *of type* `plNode` *must have been defined and provided with the data for the threshold (10) and step height (*10^{-20}*).*

to the model configuration. In depth treatment of this class is beyond the scope of this text*, but what is important here is that it is a flexible system that can deal with hierarchical data. Here, it is used to store and access the threshold and height of the cross section step function. Other derived classes would retrieve other data from a `plNode` object, but what they have in common is that they only require one, identically typed variable as input: a `plNode` object. Note, that all derived cross section classes have in common that they have a threshold. Logically, extracting the threshold value from the `node`-object, saving its value in a private member, and returning its value on request, is implemented in the base class.

Listing 5.6 shows how the set of classes can be used. Just as previously in listing 5.3 a `CrossSectionStep` object is defined in line 2, only now the initialization data is provided through a `plNode` object. Then a *pointer*† to the step cross section object is declared and initialized. Note, that the pointer is of the type of the base class. The only members of the `CrossSectionStep` object that can be accessed through the pointer are the ones defined in the base class, but that is sufficient, since only the function call operator is required. The function call operator of the base class is called in line 7. In listing 5.5 we can see that when the energy (given as argument) is above the threshold, the function `sigma_above_th()` is called which in turn calls `do_sigma_above_th()`. Automatically, the implementation of this function in the derived class `CrossSectionStep` is called, returning the cross section value.

As previously stated, the advantage of using class hierarchy (the base–derived classes construction) is that the derived classes can be accessed through a common interface. For a computer program that uses cross section objects it is only important that the threshold and cross section value can be retrieved (available through

*Documentation of the `plNode` class is available in the plasimo documentation. Additionally, many samples highlighting usage specifics are included in the source code distribution.

†A pointer is a data type that holds the address of another variable. It can be used as an alternative way to access a variable. More information can be found in appendix A.

the base class). How the cross section value is determined in the derived class is irrelevant.

Another advantage of inheritance is that it enables *code reuse*. For instance, every cross section class should have some functionality to return the threshold value. Here, the relevant code is only present in the base class, but therefore implicitly available in all derived classes. Whenever a change is required only one piece of code needs modification.

5.3.4 Self registering objects

As stated before, modularity is an important aspect of PLASIMO. One of the ways this is achieved is by the use of *plug-ins*. A plug-in is a set of software components that add certain abilities to another application*. In practice, a set of routines is combined into a *library* that can be loaded by the main PLASIMO application (or existing plug-in) when any of those routines is required. This concept is used extensively in PLASIMO; there are modules for grid generation, chemistry, solving electromagnetic problems, and many more. These modules can be combined to construct grand plasma models.

The problem of using basic inheritance is that when the object is created knowledge of the specific object must be available. This means that at *compile time* any derived class that can possibly be created must be known. It is not possible to create and load a new library with a different derived class; this would require maintenance and recompilation of the main application. This problem is dealt with through the use of *self registering objects* (SRO). This concept is heavily used in PLASIMO and described in the thesis by Van Dijk [27] and in detail in [55]. Here, we will give a short treatment.

Self registering objects are realized by extending the base and derived classes with a combination of an intermediate *factory* class and a set of *provider* classes: a *base provider* class and a set of *derived provider* classes; one for each derived model class (in our examples: cross sections). The task of the base and derived provider classes is that they can provide an object of the accompanying derived class type. The factory class has the task of tracking which provider classes are available and handing the correct derived provider object on request. In order to do so, provider classes first have to be registered by the factory class. As their name implies, self registering classes take care of this registration by themselves: the constructor of the derived provider object registers itself with the factory.

In view of the cross section class this means that a factory must be defined, as well as a base provider class for the cross section base class and derived provider

* Plug-ins are used in many software applications. An example is the use of plug-ins in web browsers. Web browsers are standalone applications, but their functionality can (often) be extended by plug-ins. Plug-ins can for instance enable the viewing of PDF files or various video formats inside the browser.

classes for derived cross section classes. Additionally, the cross section provider classes must register themselves at the factory. Derived cross section objects can then be created by requesting a cross section object from the factory by simply passing a string identifying the particular derived cross section type.

The complex process of defining the factory and provider classes can be cumbersome and error-prone. In plasimo most problems are avoided by a convenient set of *macros**. The definition of the cross section classes is shown in listing 5.7 and an example of how to use the factory in listing 5.8. Only a few adjustments of the previous code (listing 5.5) are required to use the mechanism of self registering objects. Lines 7 and 8 contain the declaration (by use of a macro) of the *base class provider*, giving it a name, and describing the format† of the constructor. In line 39 the derived step function cross section class is registered using a unique name ("Step"). In listing 5.8 this name is used to create an object of the specific type, in line 4. No knowledge of the specifics of the step cross section class is required; the interface of the cross section base class is sufficient for operation.

If another type of cross section is added, only a derived class with an accompanying "`REGISTER_PROVIDER`" line is required to register the derived class with the base provider, and to subsequently be able to create objects of that class. This code can reside in a separate library, and if dynamic linking is enabled, this code can be loaded whenever it is needed, for instance, as dictated by a configuration file (here, represented by the `node` object, in line 4 in listing 5.8), in practice a plasimo input file.

As previously mentioned, this plug-in mechanism is used throughout plasimo on all levels. At the most basic level it is used to create model plug-ins, of which the CRM is one.

5.4 The CRM as a plasimo model

In the previous section we showed how classes and plug-ins can be used to deal with cross sections in a modular way. Apart from cross sections there are many more aspects of a CRM, or any model in general, where modularity can be employed. In the CRM we can identify for instance:

- transitions: all transitions involve an originating and target level, they differ in for example the agent (electron, photon), and definition (rate coefficient, cross section).
- EEDFs: all EEDFs define the distribution function as function of energy, but some have distinct features such as Maxwellian EEDFs.

*A macro is a pattern that defines how a set of input values is mapped to an output value. Here it is used to define a set of classes from only a few strings, describing the name of the class and input for the constructor.

† The format of the constructor is the combination of types the constructor requires as arguments.

```
1  class CrossSectionBase {
2    public:
3      CrossSectionBase(plNode node) {
4        m_threshold = node("threshold");
5      }
6      // macro for declaring the provider
7      DECLARE_PROVIDER (CrossSectionBase, "CrossSection",
8        (plNode node), (node))
9      ~CrossSectionBase() { }
10     double operator()(double eps) {
11       return (eps < threshold()) ? 0.0 : sigma_above_th(eps);
12     }
13     double threshold() {
14       return m_threshold;
15     }
16     double sigma_above_th(double eps) {
17       return do_sigma_above_th(eps);
18     }
19   protected:
20     virtual double do_sigma_above_th(double eps) = 0;
21   private:
22     double m_threshold;
23 };
24
25 class CrossSectionStep : CrossSectionBase {
26   public:
27     CrossSectionStep(plNode node)
28      : CrossSectionBase(node) {
29       m_cs = node("cs");
30     }
31   protected:
32     virtual double do_sigma_above_th(double eps) {
33       return m_cs;
34     }
35   private:
36     double m_cs;
37 };
38 // macro for registering this class with the provider
39 REGISTER_PROVIDER(CrossSectionBase, CrossSectionStep, "Step");
```

Listing 5.7. *Code snippet showing the use of macros provided by* plasimo *for using self registering objects for the cross section class. Note, only three lines have been added, compared to listing 5.5: line 7 and 8 where the base provider is defined (this also creates the factory), and line 39 where the derived provider class is registered.*

```
1 // create a CrossSectionStep object, and
2 // store the pointer to that object
3 CrossSectionBase* my_cs =
4   CrossSectionBase::factory().get(node("Type")).create(node);
5 // call the function operator through the base class pointer
6 double cs_val = (*my_cs)(20); // sets cs_val to 1e-20;
```

Listing 5.8. *Code snippet showing how the factory can be used to request a cross section of a certain type. In order for this code to work with the definitions in listing 5.7, the* `node` *object must have the values for the cross section (threshold at 10 eV and height at* 10^{-20} m^2*), and the "*`Type`*" must be of "*`Step`*", see listing 5.11. The double colon ("*`::`*") on line 4 means that the* `factory()` *function is defined in the scope of the* **`CrossSectionBase`** *class. The* `factory()` *function gives access to the factory that implements a* `get()` *function. The* `get()` *function is provided with the specific type of derived class that is requested (the "*`Type`*" key in the* `node` *object which equals "*`Step`*"). The* `get()` *function then returns an object with the* `create()` *function that creates an object of the desired class, and returns a pointer to that object. The* `create()` *function requires the* `node` *object to pass it to the constructor of* `CrossSectionStep`*.*

In these two cases it is logical to implement a class for transitions and EEDFs and derive specialized classes from those.

In general any model can be designed to employ some form of modularity. In plasimo this is taken one step further; the models themselves are plug-ins. There are model plug-ins for heat balances, flows, LTE plasma, non-LTE plasmas, etc. The CRM is also implemented as a model plug-in.

5.4.1 The basic model

Model plug-ins are the most basic level of plasimo. All model plug-ins are instances of classes derived from the `plModelBase` class, see listing 5.9. This class has several `virtual` members that must be implemented in derived classes. The implementation of the `update()` and `Run()` members is shown in listing 5.10.

plasimo uses the `plModelBase` class in the following way:

1. the plasimo application loads an input file* into a `plNode` object;
2. plasimo creates an object of a `plModelBase` derived class passing the `plNode` object containing the input file to the relevant constructor;

* There are different plasimo applications: a console application, an application with a Graphical User Interface (GUI), and an experimental web interface. The console application is the simplest: it loads an input file given as argument on the command line and runs the model. There is no user interaction and the program quits when the model is finished. The GUI application is equipped with an editor for plasimo input files. In addition models can be loaded and run by use of control buttons, and graphs of calculated data can be shown real-time. The web interface version aims to provide the same functionality as the GUI through the use of a web browser.

```
1  class plModelBase {
2  public:
3      // constructor requires a plNode object passed as reference
4      plModelBase(plNode & tree);
5      // macro for declaring the provider
6      DECLARE_PROVIDER( plModelBase, "Model",
7          (plNode& node), (node) )
8      virtual ~plModelBase();
9      virtual void Prepare() = 0;
10     virtual bool CanGo() = 0;
11     virtual bool Finished() = 0;
12     virtual bool OutputNeedsUpdate() = 0;
13     virtual void UpdateOutput();
14     void update();
15     void Run();
16 protected:
17     virtual void do_update();
18 };
```

Listing 5.9. *Snippet showing the most important functions of the `plModelBase` class. In lines 6 and 7 the `DECLARE_PROVIDER()` macro is used so that derived classes can register themselves. Note, several member functions are declared pure virtual, i.e. they are not implemented in this base class, this is denoted by "=0". The functions `update()` and `Run()` on lines 14 and 15 are not implemented in this listing but in listing 5.10.*

3. plasimo calls the `Run()` member of `plModelBase`.

Listing 5.10 shows how the `Run()` member calls the members of the class `plModelBase`. It is a simple loop that starts when `CanGo()` returns `true` and calls `update()` until `Finished()` returns `true`. The other members `UpdateViews()`, `OutputNeedsUpdate()`, and `UpdateOutput()` are intended to deal with output and presentation of calculation results at the end of every call to `update()`.

A model plug-in is derived from `plModelBase` and must therefore implement its pure virtual members. Additionally, a model plug-in can override the member `do_update()` if the model involves some form of iteration; as shown in listing 5.10, the `update()` member only logs and performs a call to the `do_update()` member.

5.4.2 The CRM model plug-in

The CRM model plug-in should perform the following steps:

1. read an input file describing the model;
2. create the transition-matrix representing the processes described by the input file;

```
1  void plModelBase::Run() {
2      // start the model loop, while CanGo is true and
3    // Finished is not true
4      while (CanGo() && !Finished()) {
5          // update the model
6          update();
7          // when we are done or when update of output is needed...
8          if (Finished() || OutputNeedsUpdate()) {
9              // ...update the output and...
10             UpdateOutput();
11             // ...update the views
12             UpdateViews(true);
13         }
14     }
15     // update the views
16     UpdateViews(true);
17     // if everything went correctly Finished should be true...
18     if(!Finished()) {
19         // ...otherwise report
20         Log(3) << "Simulation ended prematurely." << std::endl;
21     }
22 }
23
24 void plModelBase::update() {
25   // report new iteration
26     Log(1) << "plModelBase: Updating model." << std::endl;
27   // call member function of derived class
28     do_update();
29 }
```

Listing 5.10. *Implementation of the* `Run()` *and* `update()` *member of the* `plModelBase` *class. Note the scope operator "*`::`*" in lines 1 and 24. This refer to the fact that the definition of the functions are found in the namespace of the* `plModelBase` *class. This is to avoid confusion with possible other functions with the same name. In lines 20 and 26 messages are sent to the* `Log`*-object. The number is the log-level, which denotes the "urgency" of the message, a higher number being more urgent.*

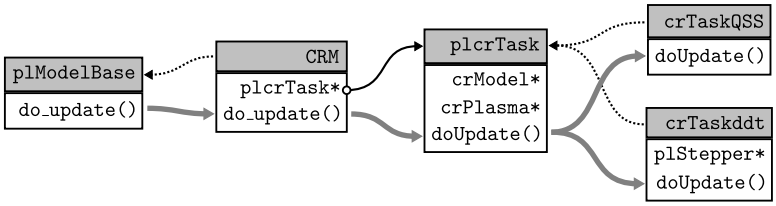


Figure 5.1. *Diagram of the basis of the CRM model plug-in C++ class structure. Grey boxes give the name of the classes and the attached white boxes (some of) their members. The solid black line denotes a pointer, the dotted lines indicate the base class of derived classes, and the thick gray lines show the call sequence. The member* `do_update()` *of class* `plModelBase` *is the member function on line 14 in listing 5.9. It is called on line 28 in listing 5.10. A call to this function will eventually end up at the* `doUpdate()` *member function of either* `crTaskQSS` *or* `crTaskddt`*.*

3. use the transition-matrix to calculate a solution, guided by parameters in the input file;

4. save and/or present the results.

Reading the input file in step 1 means that a file describing the levels, transitions (radiative and collisional), plasma parameters (electron density, EEDF, etc.), and other settings (solver, start and end time for the transient model, etc.) is read. The combination of steps 2, 3, and to some extent 4, form a single iteration and can be performed multiple times when the CRM describes a transient model. Comparing these to the main loop of plasimo (lines 4 through 14 in listing 5.10) it is clear that steps 2, 3, and 4 are implemented in the `do_update()` member function (line 28). Logically, reading the input file (step 1) is implemented in the constructor of the CRM model plug-in*.

As stated before, the CRM supports two kinds of calculations (steady state and transient) that share the creation of the transition-matrix. The difference lies in how this matrix is applied to achieve a solution. This is reflected in the way the CRM is implemented as a plasimo model plug-in, see Fig. 5.1. A `CRM` class is derived from `plModelBase`, and an extra class `plcrTask` is defined. `CRM` contains a pointer to a `plcrTask` object, and all pure virtual functions of `plModelBase` overridden by `CRM` call appropriate members of the `plcrTask` class. Classes can be derived from `plcrTask` for different purposes; here, a quasi steady state task (`crTaskQSS`) and transient task (`crTaskddt`).

The `plcrTask` class contains a pointer to a `crModel` and a `crPlasma` object (the corresponding classes are not depicted in Fig. 5.1). The `crPlasma` object holds all relevant plasma parameters, most notably the electron density and EEDF. The

* The term *reading* is not quite correct. The input file is read by the plasimo application and converted into a `plNode` data structure. This data structure is read by the constructor.

`crModel` object is the core of the model plug-in. It holds all the information about the levels and transitions, and provides accessor functions for retrieval of this information and for obtaining the transition-matrix. The `crModel` class also has a pointer to the `crPlasma` object to access any parameter it might need to calculate transition frequencies (see also Fig. 5.3). Manipulation of the transition-matrix is done through the `crPlasma` object. In the standard transient model the transition-matrix is time dependent because of the time dependency of the electron density and EEDF. Whenever the transition-matrix is requested of the `crModel` object, it uses the actual settings as they are present in the `crPlasma` object.

The first thing the constructor of `plcrTask` does is to create a `crModel` object. Upon calling the constructor the position in the input file that contains the description of the levels and transitions is passed. This is achieved using a reference* to a `plNode` object.

5.4.3 Input file

Listing 5.11 shows a part of an input file for a simple Ar CRM, shown schematically in Fig. 5.2. It consists only of the Ar atom and ion ground level and one excited level. The listing shows the format that is used for plasimo input files. It is a combination of sections demarcated by curly brackets, and *key–value* pairs. The key–value pairs are called *leafs*†, and both the leafs and sections are called *nodes*; the sections being nodes with subnodes. Lines 5, 10, and 15 are examples of the start of a section; a `Level`-section. Each `Level`-section contains three nodes, which are in this case leafs, because they do not contain subnodes: the name, energy distance to the ground level, and statistical weight. In these leafs the first part is the *key* (`Name`, `Energy`, and `Weight`), the second part the *value*. The two collisional transitions included in this model show the two ways in which the rate for a collisional transition can be described: by a cross section or by a rate coefficient. When a cross section is used to define a collisional transition, Eqn. 4.9 is used to calculate the rate coefficient. Otherwise, the defined expression (line 30).

The `Type` leaf in line 38 is the key that is used in line 4 of listing 5.8 to obtain the `CrossSectionStep` class from the `CrossSectionBase` factory. All lines in listing 5.11 containing a `Type` leaf are used for the purpose of obtaining the appropriate class from a factory. Lines 28 and 36 are passed to the `crTransCol` factory (see Fig. 5.3) that will return a `TransColRate` and `TransColCS` object respectively. The `Type` leaf in line 1 is used to request the appropriate `plModelbase` derived object from the factory.

*A reference is similar to a pointer in that they are both used to refer to another variable. The difference is that a pointer is a variable on its own, and can therefore be uninitialized, it then simply points to nothing. A reference can be seen as another name for an object. This means that whenever a reference is created that the object it refers to must exist. More information on references can be found in appendix A.

†A leaf can have multiple values.

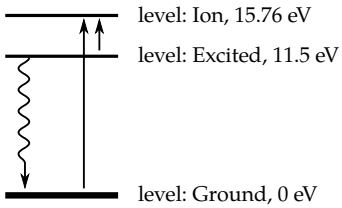


Figure 5.2. *Diagram depicting the model described in listing 5.11. Only three levels are included, two collisional transitions, and one radiative transition.*

As is evident from the listing, the file format is very descriptive and offers great flexibility through its hierarchical structure, which is similar to for instance XML.

The plasimo framework offers the *plParser* module, which is a library of classes that can be used to process and manipulate these kind of input files. At the heart of the plParser module lies the `plNode` class. When plasimo loads an input file, the data in the input file is stored in a nested structure of `plNode` objects, that reflects the structure of the input file. The `plNode` class is equipped with iterators and accessor functions to traverse the structure. Furthermore, it offers a very convenient feature that deals in a transparent manner with physical units. In listing 5.11 several variables are defined in a certain unit. When the value of a leaf is requested from the relevant `plNode` object, a string containing the desired physical unit can be passed. The unit is then automatically converted from the unit as defined in the input file to the desired unit*. For instance, when a variable called `node` refers to one of the `Level` sections in the input file, the energy value of the level expressed in Joule can be retrieved by the simple code:

```
double energy = node("Energy")["J"];
```

The levels and transitions defined in the input file determine what the transition-matrix looks like. The `crModel` constructor stores the levels, radiative transitions, and collisional transitions in STL† containers. The class structure is shown in figure 5.3. The radiative (`crTransRad`) and collisional (`crTransCol`) class are both derived from the general transition class `crTrans`. The originating and target level of a transition are indices into the list of levels the model contains.

The `frequency()` member of a radiative transition simply returns the probability‡. In case of a collisional transition it calls the `do_frequency()` member, which either calculates the frequency from a cross section and EEDF, as in equation (4.9), or it is calculated by evaluating a mathematical expression, such as line 30 in listing 5.11. The `plMathFunction` class is provided by plasimo. It is

*An error is generated when conversion is not possible

†Standard Template Library, a collection of generic algorithms, containers, functors, and iterators.

‡The probability is actually multiplied by an escape factor, which is also a member of a `crTransRad` class, but this is not shown in the diagrams.

```
Type  CRM

Atom  {
  Name  Argon
  Level {
    Name  Ground
    Energy  0*eV
    Weight  1
  }
  Level {
    Name  Excited
    Energy  11.5*eV
    Weight  12
  }
  Level {
    Name  Ion
    Energy  15.76*eV
    Weight  6
  }
  RadiativeTransition {
    From  Excited
    To  Ground
    Probability 1e8*s^-1
  }
  CollisionalTransition {
    From  Ground
    To  Excited
    Type  Rate
    Rate  {
      Expression  5e-15*Te^(0.74)*exp(-1.34e5/Te)
    }
  }
  CollisionalTransition {
    From  Excited
    To  Ion
    Type  CrossSection
    CrossSection  {
      Type  Step
      Sigma 1e-20*m^2
    }
  }
}
```

Listing 5.11. *Part of a* plasimo *input file describing the atomic system for a simple CRM.*

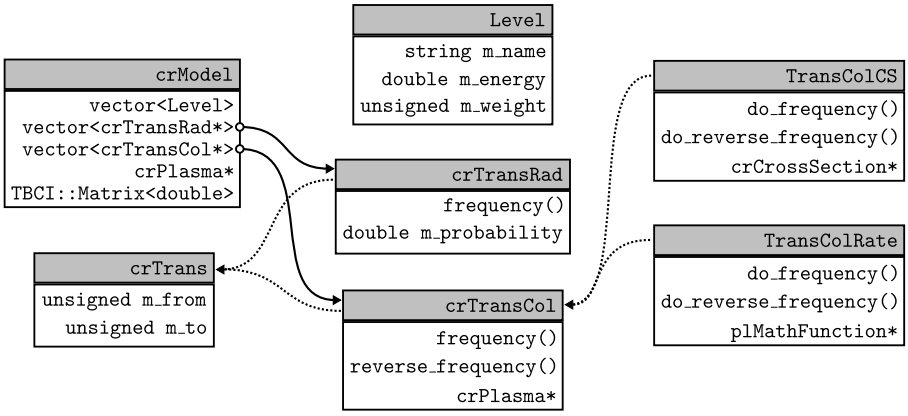


Figure 5.3. *Diagram of C++ class structure of the* `crModel` *class. The gray boxes give the name of the class, (some of) the members and member functions are listed in the corresponding white boxes. The solid lines denote a pointer to an object, the dotted lines show the base–derived class hierarchy. For some class members not only the type but also the name of the variable (starting with "*`m_`*") is shown.*

a class that can parse general mathematical expressions which can be evaluated at *run time*. The same class is used for custom cross section functions.

The `crModel` class contains a member of type `TBCI::Matrix<double>`. This is a matrix class (`double` refers to the data type of the matrix elements) that is provided by the TBCI library [56]. TBCI stands for Temporary Base Class Idiom, it is a C++ library of numerical base classes which implement basic data structures such as complex numbers, vectors, and various types of matrices (full, band, sparse, etc.). Additionally, some matrix solvers (e.g. LU-decomposition) and interfaces to netlib libraries such as CLAPACK and SUPERLU are included. The library achieves very high performance and enables the use of easily readable code. For example, when `A` is a matrix and `x`, `y`, and `z` are vectors, the following valid code can be used:

```
z = A * x - y;
```

TBCI matrices and vectors are used in the CRM model plug-in for all calculations involving the transition frequency matrix.

5.4.4 Optimization

In section 4.3.6 we already discussed a way to optimize the calculation of the rate coefficients for collisional transitions. Here, we will elaborate on its implementation.

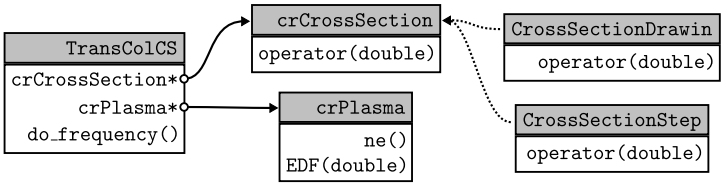


Figure 5.4. *Diagram of C++ class structure for calculating the collisional transition frequency. Grey boxes give the names of the classes and the attached white boxes their members. The solid lines denote a pointer to an object, the dotted lines show the base class of a derived class.*

The naive approach is shown in a diagram in figure 5.4; the rate coefficient is calculated by numerical integration*. To determine the integrand a cross section and an energy distribution function are required. The cross section is acquired through a pointer to a `crCrossSection` object that provides the value of the cross section as function of energy through its `operator()` member. For every cross section described in section 4.3.1 a derived class is available (only two are shown in the diagram for clarity). The EEDF is present in a `crPlasma` object, which is also accessible by means of a pointer. The calculation of the integral is implemented using the trapezoidal rule, a "one size fits all" solution, that can handle non-smooth cross sections and/or EEDFs.

The key of the optimization strategy is that rate coefficients involving certain combinations of EEDF and cross section can be calculated efficiently; i.e., instead of a general numerical integration scheme, a more efficient scheme can be used, or even an analytical solution is available. The implementation is realized by the use of *polymorphism* and is outlined in the diagram in Fig. 5.5. The benefit of the optimization is that it does not require a lot of code, however, the cost is increased complexity.

Instead of the single `EDF` class a derived class `EDFMaxwellian` is available in case the EEDF of the plasma is Maxwellian. An additional class `RateCoef` is introduced that handles the calculation of the rate coefficient. By using *overloading*, the `RateCoef` class contains a specialized member for calculating the rate coefficient for every possible type of EEDF (here, only Maxwellian and non-Maxwellian). Furthermore, the `crCrossSection` class is extended with a member called `CalcRateMaxwellian()`, that performs the calculation in case the EEDF is Maxwellian, using the Gauss-Laguerre quadrature rule. When the rate coefficient can be calculated analytically (as for the step function cross section), the member function is overloaded by the derived cross section class, that implements the calculation. The flow of the calculation is now as follows:

*The plasimo framework includes classes for calculating integrals using a number of different algorithms.

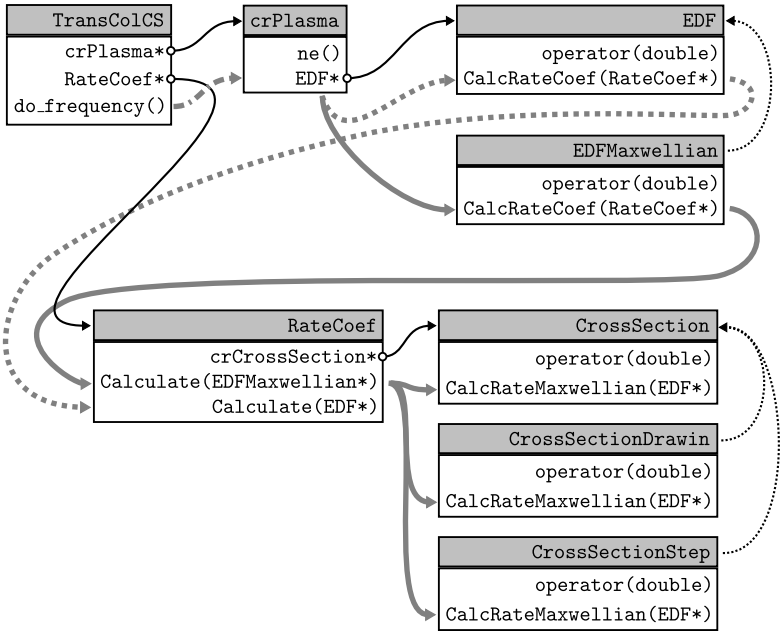


Figure 5.5. *Diagram of C++ class structure for optimized calculation of the collisional transition frequency. Grey boxes denote classes and the attached white boxes their members. The black solid lines denote a pointer to an object, the black dotted lines show the base class of a derived class, the thick gray lines show the flow of execution in case of a Maxwellian EDF (solid) and any other EDF (dotted).*

1. `do_frequency()` calls the `CalcRateCoef()` member of the `EDF` object in the `crPlasma` object. A pointer to the `RateCoef` object is passed when calling the member.

2. The `CalcRateCoef` function then calls the `Calculate()` member of the `RateCoef` object it was passed. In calling it passes the `this` pointer.

3. Depending on the type of the `this` pointer, which in this case points to either an object of type `EDF` or `EDFMaxwellian`, the `Calculate()` member with the appropriate signature is called.

   a) If the EEDF is not Maxwellian `RateCoef.Calculate()` calculates the rate coefficient using the trapezoidal rule.

   b) If the EEDF is Maxwellian the `CalcRateMaxwellian()` member of the cross section object is called. The `CalcRateMaxwellian()` member of the base class calculates the rate coefficient using the Gauss-Laguerre quadrature.

```
1 TBCI::Matrix<double> F_ll_inv = TBCI::lu_invert(F_ll);
2 TBCI::Matrix<double> J = F_tt - F_lt * F_ll_inv * F_tl;
```

Listing 5.12. *Code fragment showing the calculation of matrix* $\boldsymbol{J}$ *equation* (3.64). *The matrices* `F_tt`, `F_tl`, `F_lt`, *and* `F_ll` *have already been created.*

c) However, if this member is overridden, the implementation in the derived class is executed, thus enabling calculation of an analytical solution (as for the step function cross section).

This structure can easily be extended to support more types of energy distribution functions. All that is needed is an `EDF` derived class and an overloaded `RateCoef.Calculate()` member. If for certain cross sections an additional optimization can be achieved, the `CrossSection` class can be expanded with an appropriate member function.

5.5 Solution procedure

The code described so far is used to load the input file and create a transition frequency matrix from it. The matrix can now be used to either calculate a Quasi Steady State solution or start an iterative solution of a transient model. For each case a different class derived from `plcrTask` is available, see figure 5.3.

5.5.1 Quasi Steady State

In the QSS solution the ASDF, effective conversion rates, and source terms for the energy equation are calculated from the simple expressions listed in section 4.2.1. Because the TBCI library is used for matrix and vector types this can be accomplished using short, readable code. In listing 5.12 the calculation of matrix $\boldsymbol{J}$ is shown. On the first line the inverse of $\boldsymbol{F}_{ll}$ is calculated using a function provided by the TBCI library. In the second line $\boldsymbol{J}$ is calculated by a simple expression, which exactly matches equation (3.64).

The calculation of the QSS solutions is implemented in the `crModel` class. The reason for this is that the results can be used in different `plcrTask` classes. In the QSS task this is used to calculate the QSS solutions of an atom for a range of plasma parameters (n_e, T_e). The results can be saved in the form of a look-up table. For instance, a look-up table of effective conversion rates can be used in a plasimo fluid model that only takes into account the TS levels. The transient task class can use the QSS solution provided by `crModel` for calculating the initial densities.

The `crTaskQSS` class can calculate solutions for different electron density, electron temperature, and TS level densities by manipulating the `crPlasma` class.

5.5.2 Transient

The time dependent solution is implemented in the `crTaskddt` class. The type of problem that has to be solved is an initial value problem. For this type of problem many solution strategies are known. To accommodate a variety of solution schemes the solver is implemented as a plug-in: the *stepper*. In general, a stepper (**g**) should give the solution of a set of ordinary differential equations after a certain time step (Δt) given a set of initial values ($\mathbf{y}(t_0)$) and a function that provides the source terms of the differential equations (**f**) as in equation (4.6):

$$\mathbf{y}(t_0 + \Delta t) = \mathbf{y}(t_0) + \mathbf{g}(t_0, \Delta t, \mathbf{y}(t_0), \mathbf{f}(t', \mathbf{y}(t'))), \tag{5.1}$$

where the stepper determines at which time t' the source function is evaluated. In case of the CRM, vector **y** is the vector of level densities **n**, and the vector containing the source terms of the differential equation is the product of the transition frequency matrix and density vector $\mathbf{f}(t') = \boldsymbol{F}(t')\mathbf{n}(t')$, so

$$\mathbf{n}(t_0 + \Delta t) = \mathbf{n}(t_0) + \mathbf{g}(t_0, \Delta t, \mathbf{n}(t_0), \boldsymbol{F}(t')\mathbf{n}(t')). \tag{5.2}$$

The stepper should be implemented in a general way so it can be used as a module in other PLASIMO models*. The actual implementation is achieved in a slightly different way: instead of calculating and returning the step vector **g**, the stepper also takes care of the addition, so it immediately returns $\mathbf{n}(t_0 + \Delta t)$.

From equations (5.1) and (5.2) it is evident that the stepper class should have an interface accepting a time, a time step, a data vector, and a function that accepts a time and data vector, and that the class should somehow return a data vector.

5.5.3 Callback using a functor

Passing a function as an argument can be realized using a *callback*. In C++ (and in C) a reference to a function (or member function) can be passed in a function call. We will demonstrate this mechanism with two simple programs shown in listings 5.13 and 5.14†.

The code in listing 5.13 demonstrates a callback for a member function. The `call()` member function of the `Caller` class can be used to call any function in the `Printer` class as long as the signature (input and output types) fits.

However, listing 5.13 also shows an important shortcoming. In order to call a member function of a certain class the caller must have knowledge of the class the function is a member of. When applied to the stepper, it means that the stepper must know about any class that might provide a function that is to be called by

*The stepper will also be used in the more general zero dimensional PLASIMO model described in chapter 6.

† Note that the examples are general pieces of code intended to demonstrate the mechanism and are separate from the CRM implementation.

```
#include <iostream>

class Printer {
  public:
    void print_a(int val) {
      std::cout << "Printer A: " << val << std::endl;
    }

    void print_b(int val) {
      std::cout << "Printer B: " << val << std::endl;
    }
};

class Caller {
  public:
    void call(Printer prn,
      void (Printer::*p_print)(int), int val) {
      (prn.*p_print)(val);
    }
};

int main() {
  Printer myprn;
  void (Printer::*p_print_a)(int) =
    &Printer::print_a; // p_print_a points to print_a
  void (Printer::*p_print_b)(int) =
    &Printer::print_b; // p_print_b points to print_b

  (myprn.*p_print_a)(1); // result: Printer A: 1
  (myprn.*p_print_b)(2); // result: Printer B: 2

  Caller mycall;
  mycall.call(myprn, p_print_a, 2); // result: Printer A: 2
  mycall.call(myprn, p_print_b, 1); // result: Printer B: 1
}
```

Listing 5.13. *Small program showing the use of a callback in combination with classes.*

the stepper (function **f** in equation (5.1)). To allow for the stepper to be able to call any function in any class without prior knowledge a special class is used: a *functor*.

A functor is a class that is used as a function. It must therefore have an implementation of the function call operator. The most important benefit of using a functor is that it can contain state, see section 5.3. This means that the functor can store the class and member function it has to call. The caller (here the stepper) only needs access to the general functor object, which in turn knows which member function of which class it has to call.

Listing 5.14 demonstrates the usage of a functor. The constructor of the class `Functor` stores the class and member function that have to be called. The function call operator simply calls the stored member function of the stored class. The `Caller` class does not need to know about the `Printer` class, but only about the general `Functor` class.

Although the functor in listing 5.14 already shows some generalization, it can be improved upon by using *templates*, see also appendix A. In listing 5.14 the functor only works for a specific class type with a specific member function signature. Templates allow for a general functor class, that can access any member function in any class, with any type as input and output. When the type of the functor is defined (as input type for the stepper and type of the functor object in the class calling the stepper) the input and output types are defined. When the functor object is created, the member function the functor has to call and the class it belongs to are passed to the constructor.

Listing 5.15 shows the stepper base class. Through the macro in line 5 it supports self registration of derived objects. The input parameters reflect those of function **g** in equation (5.1) (though in different order) with `h` representing the time step. In the implementation a reference to the output vector is also an input parameter. To support algorithms that can adjust the time step, the requested time step is passed as a reference. On return it contains the time step that was actually taken, and the return value of the stepper itself is the (suggested) time step for the next iteration.

5.5.4 Implemented steppers

By implementing a class derived from `plStepper` a specific stepper, or solver, algorithm can be implemented. This has been done for a few algorithms which we will shortly discuss.

```
#include <iostream>

class Printer {
  public:
    void print_a(int val) {
      std::cout << "Printer A: " << val << std::endl;
    }
    void print_b(int val) {
      std::cout << "Printer B: " << val << std::endl;
    }
};

class Functor {
  public:
    Functor(Printer* prn, void(Printer::p_print)(int))
     : m_prn(prn)
     , m_p_print(p_print)
    { }
    void operator()(int val) {
      (*m_prn.*m_p_print)(val);
    }
  private:
    Printer* m_prn;
    void (Printer::*m_p_print)(int);
};

class Caller {
  public:
    void call(Functor* func, double val) {
      (*func)(val);
    }
};

int main() {
  Printer myprn;
  Functor myfunc_a(&myprn, &Printer::print_a);
  Functor myfunc_b(&myprn, &Printer::print_b);
  Caller mycall;
  mycall.call(&myfunc_a, 1); // result: Printer A: 1
  mycall.call(&myfunc_b, 2); // result: Printer B: 2
}
```

Listing 5.14. *Small program showing the use of a functor for a callback mechanism.*

```
1  class plStepper
2  {
3    public:
4      plStepper(){}
5      DECLARE_PROVIDER (plStepper, "plStepper",
6        (const plNode& node), (node))
7      virtual ~plStepper(){}
8      double operator()(
9        TFunctor<TBCI::Vector<double>&, double,
10         TBCI::Vector<double>& >* f, // the function that is
               called
11       TBCI::Vector<double>& y_in, // input vector
12       TBCI::Vector<double>& y_out, // output vector
13       double t, // time of input vector
14       double& h) // requested time step
15     {
16         return doStep(f,
17         y_in,
18         y_out,
19         t,
20         h);
21     }
22   protected:
23     virtual double doStep(
24       TFunctor<TBCI::Vector<double>&, double,
25         TBCI::Vector<double>& >* f,
26       TBCI::Vector<double>& y_in,
27       TBCI::Vector<double>& y_out,
28       double t,
29       double& h) = 0;
30 };
```

Listing 5.15. *The stepper base class. Derived classes only need to implement the function* `doStep`*. Line 5 shows that the class is implemented as a plug-in. Parameters are passed to the different implementations through a node object, that can contain the initial step size, tolerances, etc. In this case the functor has a double and a TBCI vector of doubles as input and also a TBCI vector of doubles as output. They represent the time and input vector and output vector of function* **f** *in equation* (5.1)*. The stepper takes the function that is called as input, as well as a reference to a vector for the input and output values, the time of the input vector, and a request for the time step. The return value is the step size for the next iteration.*

```
class StepperForwardEuler : public plStepper
{
  public:
    StepperForwardEuler(const plNode& node)
     : plStepper()
     , m_h(node("TimeStep")["s"])
    { }
  protected:
    virtual double doStep(
    TFunctor2<TBCI::Vector<double>&, double,
      TBCI::Vector<double>& >* f,
    TBCI::Vector<double>& y_in,
    TBCI::Vector<double>& y_out,
    double t,
    double& h) {
      y_out = y_in + (*f)(y_in, t) * m_h;
      return m_h;
    }
  private:
    double m_h;
};

REGISTER_PROVIDER(plStepper,
  StepperForwardEuler, "ForwardEuler");
```

Listing 5.16. *Implementation of a forward Euler stepper. The time step is fixed and passed to the constructor via a* `plNode` *object.*

Forward Euler

Listing 5.16 shows the implementation of the simplest algorithm: the forward or explicit Euler scheme:

$$\mathbf{y}(t + \Delta t) = \mathbf{y}(t) + \mathbf{f}\left(t, \mathbf{y}(t)\right)\, \Delta t \tag{5.3}$$

In the case of the CRM the source term is linear $\mathbf{S}(t) = \boldsymbol{F}(t)\mathbf{n}(t)$ so this will result in:

$$\mathbf{n}(t + \Delta t) = \mathbf{n}(t) + \boldsymbol{F}(t)\mathbf{n}(t)\, \Delta t. \tag{5.4}$$

Note, that care must be taken when choosing the time step. Typically, the time step must be smaller than twice the reciprocal of the highest destructive frequency in matrix $\boldsymbol{F}$.

Runge-Kutta

In addition to forward Euler, several other methods have been implemented. The fourth order Runge-Kutta method [57], also known as simply RK4, is given by:

$$\mathbf{y}(t+\Delta t) = \mathbf{y}(t) + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right), \tag{5.5}$$

with

$$\mathbf{k}_1 = \Delta t\, \mathbf{f}(t, \mathbf{y}(t)) \tag{5.6a}$$

$$\mathbf{k}_2 = \Delta t\, \mathbf{f}(t + \frac{1}{2}\Delta t, \mathbf{y}(t) + \frac{1}{2}\mathbf{k}_1) \tag{5.6b}$$

$$\mathbf{k}_3 = \Delta t\, \mathbf{f}(t + \frac{1}{2}\Delta t, \mathbf{y}(t) + \frac{1}{2}\mathbf{k}_2) \tag{5.6c}$$

$$\mathbf{k}_4 = \Delta t\, \mathbf{f}(t + \Delta t, \mathbf{y}(t) + \mathbf{k}_3). \tag{5.6d}$$

This method is implemented with an adaptive step size Δt. By evaluating (see [58]):

$$\kappa = 2\left|\frac{\mathbf{k}_3 - \mathbf{k}_2}{\mathbf{k}_2 - \mathbf{k}_1}\right| \tag{5.7}$$

the step is adjusted. When the minimum value of κ falls below 0.01 the step size is halved, when the maximum value exceeds 0.05 the step size is doubled, otherwise it is left unchanged.

Dormand-Prince

The third algorithm that has been implemented is an embedded Runge-Kutta method, named Dormand-Prince [59]. In embedded methods two solutions are calculated with different order. The difference of these two results is considered as an error that is used to optimize the next step size. The Dormand-Prince method gives fifth order accuracy.

LSODA

The methods described so far are applicable to non-stiff problems. A more advanced stepper that also support stiff problems is the so called LSODA solver. This solver by Hindmarsh and Petzold [60, 61] holds an implementation of two methods: Adams-Moulton for non-stiff problems and Backward Differentiation Formula (BDF) for stiff problems. Both methods are multistep methods, meaning that results from multiple previous time steps are used. Since classes retain state between calls this does not pose a problem for the implementation. The derived stepper class can keep track of previous steps. The solver automatically switches between methods to select the optimal algorithm. Originally the solver was available in a Fortran implementation, for PLASIMO a translated version into C was implemented in a `plStepper` derived class.

5.5.5 Transient results

By using the stepper the evolution of the level densities is calculated over time. By extracting certain elements from the transition frequency matrix the rates for specific processes can be calculated. For instance, the radiative transitions in the transition matrix are represented by their transition probabilities A^*, see Eqn. (4.44). To calculate all the isolated radiative transition rates we create a matrix $\boldsymbol{R}$ with elements:

$$R_{rp} = A^*(p, q), \tag{5.8}$$

where p and q represent the upper and lower levels of the radiative transition, and r represents an index for the specific radiative transition. If there are M radiative transition in the model and N levels, matrix $\boldsymbol{R}$ has dimensions $M \times N$. Simply multiplying matrix $\boldsymbol{R}$ with the level density vector gives a vector of radiative transition rates. The same can be done for electron (de-)excitation collisions so relative (de)population mechanisms per level can be studied. In the GUI all level densities and individual collisional and radiative rates are plotted as a function of time. These results are also saved to file.

5.6 Conclusion

In this chapter the implementation of the plasimo CRM plug-in has been described. The process from a mathematical description of the transition frequency matrix and solution strategies to the way this has been realized using several tools present in the plasimo framework has been treated. The result is a general CRM code that can be applied to a wide range of atomic plasmas. Because of the graphical user interface that plasimo provides, it allows for user friendly manipulation of input parameters and convenient examination of results.

Chapter 6

Implementation of a GPM code in the plasimo framework

6.1 Introduction

In chapter 2 a classification of models for plasma chemistry was given. Collisional Radiative Models (CRM) were the subject of chapters 3, 4 and 5, where the tasks and a detailed description of the implementation was given. In this chapter we will focus on the implementation of a Global Plasma Model (GPM) code.

In a GPM volume averaged plasma parameters are calculated as a function of external control parameters, such as pressure, geometry, and input power. The GPM described in this chapter possesses characteristics of a Reaction Exploration Model (REM) meaning that the source terms are constructed directly from the rates of the reactions in the model. This in contrast to the CRM where the source terms are written as the product of a frequency and species density. In the GPM, the reactions can be initiated by any species included in the model, whereas in the CRM this is restricted to external agents (electrons and photons).

The GPM is based on the model *RateLab* by Jiménez [13], which in turn is based on the *PyRate* model by Van den Donker [14]. The goal is to model the species densities as a function of time, depending on external control parameters, mainly input power. In addition to the species densities, the electron energy balance is solved. As previously with the CRM, the implementation is realized using the plasimo framework.

We will continue this chapter with the construction of the balance equations that constitute the model in section 6.2. In section 6.3 some special cases for the

source terms will be treated, such as wall reactions. Section 6.4 contains specifics of the implementation of this model as a plasimo plug-in.

6.2 General equations

In a GPM the plasma volume is averaged resulting in a zero-dimensional model describing the *chemistry*. This means that there is no *transport* (in configuration space) within the model. We are solely investigating species densities, and how the rates of various chemical reactions affect them. Transport and configuration are reduced to rates (i.e. frequencies) that are applied to the relevant species.

6.2.1 Species balance

As with the CRM we start with the zeroth moment of the Boltzmann transport equation:

$$\frac{\partial n_s}{\partial t} + \nabla \cdot (n_s \mathbf{v}_s) = \mathcal{S}_s, \tag{6.1}$$

where n_s, $\mathbf{v}_s$, and $\mathcal{S}_s$ are respectively the density, mean velocity, and source term of species s. The transport term can be written using a transport frequency:

$$\mathcal{F}_s = \frac{1}{n_s} \nabla \cdot (n_s \mathbf{v}_s), \tag{6.2}$$

so that Eqn. 6.1 becomes:

$$\frac{\partial n_s}{\partial t} + \mathcal{F}_s n_s = \mathcal{S}_s. \tag{6.3}$$

For a system of species we can write this in vector notation:

$$\frac{\partial \mathbf{n}}{\partial t} + \boldsymbol{\mathcal{F}} \mathbf{n} = \mathbf{S}, \tag{6.4}$$

where $\boldsymbol{\mathcal{F}}$ is a diagonal matrix with elements (6.2), or when we neglect transport:

$$\frac{\partial \mathbf{n}}{\partial t} = \mathbf{S}. \tag{6.5}$$

In the CRM the source term was written as the product of a transition matrix and the density vector. This was possible, because the transitions were the result of external agents (electrons, photons), so only linear reactions were included. In the GPM we will deal with generic reactions such as:

$$\alpha A + \beta B \rightarrow \gamma C + \delta D, \tag{6.6}$$

or in general:

$$\sum_s \nu^{\mathrm{d}}_{s,r} X_s \overset{k_r(T)}{\rightarrow} \sum_s \nu^{\mathrm{p}}_{s,r} X_s, \tag{6.7}$$

which describes the *destruction* of species X_s^{d} and the *production* of species X_s^{p}, with ν_s^{d} and ν_s^{p} the stoichiometric coefficients, and $k_r(T)$ the temperature dependent rate coefficient of this reaction. The rate at which this reaction occurs is then:

$$R_r(\mathbf{n}, T) = k_r(T) \prod_s n_s^{\nu_{s,r}^{\mathrm{d}}}, \tag{6.8}$$

where the product runs over all species s, with densities n_s. The sum in Eqn. (6.7) and product in Eqn. 6.8 can run over all species included in the model. In case the species are not involved in the reaction the stoichiometric coefficient $\nu_{s,r}$ equals 0, so the species is not included in the sum and its element in the product equals 1. This will be convenient in the general implementation.

The rate of a reaction is always expressed in $\mathrm{m}^{-3}\,\mathrm{s}^{-1}$, so the unit of the rate coefficient depends on the number of species on the left-hand side in Eqn. (6.7) and their stoichiometric coefficient. The unit of k_r is $\mathrm{m}^{3N_r-1}/\mathrm{s}$, with $N_r = \sum_s \nu_{s,r}^{\mathrm{d}}$.

The balance equation (Eqn. (6.3)) of every species involved in a reaction will receive a source term from that reaction, defined by the net stoichiometric coefficient multiplied by the rate:

$$\mathcal{S}_{s,r} = \left(\nu_{s,r}^{\mathrm{p}} - \nu_{s,r}^{\mathrm{d}}\right) R_r. \tag{6.9}$$

The cumulative source of all reactions for a species can be written as:

$$\mathcal{S}_s = \sum_r \left(\nu_{s,r}^{\mathrm{p}} - \nu_{s,r}^{\mathrm{d}}\right) R_r = \mathbf{W}_s \cdot \mathbf{R}, \tag{6.10}$$

where $\mathbf{W}_s$ is a vector of net stoichiometric coefficients, one per reaction, and $\mathbf{R}$ is a vector of rates, again one per reaction. The vectors $\mathbf{W}_s$ of all species can be combined in a matrix $\boldsymbol{W}$, one on every row, so that the balance equation of the system, Eqn. (6.5), can be written as:

$$\frac{\partial \mathbf{n}}{\partial t} = \mathbf{S} = \boldsymbol{W}\mathbf{R}. \tag{6.11}$$

In the implementation described in this chapter, the definition of the species balance will take place in two steps:

1. Define all required species. For every species the name, chemical composition, and energy level must be defined.

2. Define all reactions. A reaction is defined by a chemical equation, in which only species defined in step 1 can be included, and a rate coefficient.

Reactions that do not involve transport and can be described as outlined above will be called *volume reactions*. Reactions in which special care has to be taken, for instance due to transport, will be treated in section 6.3. In addition to the species density balances the model includes the electron energy balance.

6.2.2 Energy balance

The electron energy balance is defined as:

$$\frac{\partial}{\partial t}\left(\tfrac{3}{2} n_e k_B T_e\right) = P_{\text{input}}(t) - Q_{\text{inelas}} - Q_{\text{elas}} + Q_{\text{extra}}, \tag{6.12}$$

with n_e the electron density, k_B the Boltzmann constant, T_e the electron temperature, P_{input} the input power density*, Q_{extra} extra energy source terms, and Q_{inelas} and Q_{elas} the energy losses from inelastic and elastic processes between electrons and heavy particles. The input power density P_{input} is a parameter of the model, and can take any form as function of time.

In processes involving an electron, the energy difference between left and right-hand side is attributed to the electrons. For instance when we have an ionization reaction of species X, where ion X^+ has an energy of 15 eV:

$$X + e + (E_{\text{ion}}) \overset{k_{\text{ion}}}{\rightarrow} X^+ + e + e, \tag{6.13}$$

the electrons lose $E_{\text{ion}} = 15\,\text{eV}$ per reaction. For this ionization reaction the inelastic source term will be:

$$Q_{\text{inelas,ion}} = E_{\text{ion}}\, R_{\text{ion}} = E_{\text{ion}}\, n_X n_e\, k_{\text{ion}}. \tag{6.14}$$

The total inelastic source term can be written as:

$$Q_{\text{inelas}} = \sum_r E_{e,r} R_r = \mathbf{E}_e \cdot \mathbf{R}, \tag{6.15}$$

where $\mathbf{E}_e$ is a vector of electron energy transfers, one per reaction.

The extra source term Q_{extra} can be used to add any reaction that does not fit into the general description of Eqn. (6.15) to the electron energy balance. An example of this is energy exchange by elastic collisions (Q_{elas}), which will be implemented as an extra source term (thereby essentially removing it as a separate term). This will be described in section 6.3.2.

6.2.3 Solution procedure

The goal of the model is to simultaneously solve the species density balances (Eqn. (6.11)) and electron energy density balance (Eqn. (6.12)) as a function of time. The power input density is a function of time and additional source terms, see next section, can also be time dependent. The result of this calculation will be the species densities and the electron energy density as a function of time. Additionally, the rates vector $\mathbf{R}$ is calculated for the same time steps as the densities, which allows for time dependent analysis of reaction paths.

* It is assumed all input power is directly coupled into the electron energy balance.

The set of balances is an initial value problem similar to the time dependent CRM in section 4.2.2. Consequently, the same solution procedure can be employed, where now the the vector $\mathbf{Y}$ consists of the species densities and electron energy density.

6.3 Special cases

When a reaction is defined as described in section 6.2, it implies three things:

1. every species defined in a balance equation must be defined, and for that species a density balance will be created;

2. every reaction will result in a source term for all species, which through the mechanism of Eqns. (6.10) and (6.11) will be limited to the species with a non-zero stoichiometric coefficient;

3. every reaction will result in an inelastic source term for the electron energy balance, limited to relevant reactions by vector $\mathbf{E}_e$ in Eqn. (6.15).

So defining only the reactions (and the required species) results in a set of balance equations (species density and electron energy density) with appropriate source terms. However, it is possible that reactions, especially when interaction with the surroundings is involved, will require some adjustments to fit into this general description. An example of this are simple wall reactions.

6.3.1 Wall reactions

The most basic form of a wall reaction is a particle diffusing to the wall. Translated to the zero-dimensional paradigm, it means that the density of the relevant species, say X, disappears at a certain frequency:

$$X \overset{k_{\text{diff}}}{\rightarrow} \qquad , \tag{6.16}$$

which violates the law of conservation of mass. A workaround for this problem is the introduction of a specialized *wall* species, X_{wall} that is identical to the original species, X. We can then write:

$$X \overset{k_{\text{diff}}}{\rightarrow} X_{\text{wall}}. \tag{6.17}$$

Although this introduces another species and therefore a balance equation, this will not influence the system, since this wall species only appears on the right-hand side of this reaction.

The rate coefficient describing the diffusion will in general be of the form (see Eqn. (2.6)):

$$k_{\text{diff}} = \frac{D}{\Lambda^2}, \tag{6.18}$$

with Λ the size of the plasma, and D the diffusion coefficient. The diffusion coefficient of species X is described by Eqn. (2.7):

$$D_X = \frac{2}{3N\sigma}\sqrt{\frac{k_B T_X}{\pi M_X}}, \tag{6.19}$$

where N is the density of the buffer gas, σ the collision cross section, and T_X and m_X the temperature and mass of species X. In case species X has a charge the diffusion changes to the ambipolar diffusion (see Eqn. (2.9)):

$$D_X^{\text{ambi}} = (1 + \frac{T_e}{T_h})D_X, \tag{6.20}$$

with T_e and T_h the electron and heavy particle temperature.

A more complex wall interaction can occur when some mechanism introduces new particles into the plasma. An example of this is the sputtering of the wall.

When ions reach the wall and are energetic enough they can sputter the surface which releases new particles into the plasma. For instance, the ions from an Ar plasma sputter a Cu surface*:

$$Ar^+ + Cu_{\text{wall}} \overset{k_{\text{sputter}}}{\rightarrow} Ar^+ + Cu. \tag{6.21}$$

However, defining this reaction will result in a balance for Cu_{wall} and computation of the rate requires its density by definition, Eqn. (6.8), but the rate should be independent of the "makeshift" species Cu_{wall}. There are several ways to deal with this issue, here we have opted for extra source terms.

6.3.2 Extra sources

The source terms for the particle balances are constructed from the reactions defined in the models. To these sources we can add extra source terms "by hand", i.e., by circumventing the automatic source term creation from the defined reactions. The calculation of the source term remains the same, as defined in Eqns. (6.8) and (6.9). But now, we can define the relevant species, i.e., the non-zero left-hand side, right-hand side, and net stoichiometric coefficients separately, so we do not have to worry about mass conservation of the individual reaction.

With respect to the sputtering example, we can define *target* species, in this case Cu, *source* species, here Ar^+, and the net stoichiometric coefficient for the

* Note that in this case the Ar ions return into the plasma after the sputter process.

target species $W_{\text{net}} = 1$. The declaration of the extra source term is completed with the definition of the rate coefficient. Effectively, we have then defined:

$$Ar^+ \overset{k_{\text{sputter}}}{\rightarrow} Ar^+ + Cu. \tag{6.22}$$

The source term is calculated from:

$$\mathcal{S}^{\text{extra}}_{\text{sputter}} = W_{\text{net}} k_{\text{sputter}} \prod_{\text{sources}} n_{\text{source}}, \tag{6.23}$$

which is added to the balance equation of the target species, Cu. This will be used in chapter 9, where we describe a model of a sputtering plasma.

Another, more common, case in which extra source terms are useful is the electron energy loss term for inelastic collisions:

$$X + e \overset{k_{\text{inelas}}}{\rightarrow} X + e. \tag{6.24}$$

Following the general description of generating source terms for the balance equations, this reaction has no effect: all net stoichiometric coefficients are zero, and since the left-hand and right-hand side energies are equal there is no energy exchange.

Since this process only affects the electron energy balance, extra source terms can also have the electron energy as target. For this process the sources of the extra source term are X and e, the target is the electron energy, and the stoichiometric coefficient is 1.

6.4 Implementation

The GPM must perform the following steps:

1. load an input file, describing the species, reactions, and any control parameters required by the model, such as input power density;
2. prepare data structures for the iterative solution procedure;
3. perform the iterative solution procedure;
4. save and/or present results.

The GPM is implemented as a model plug-in in the PLASIMO framework using the C++ programming language. Plug-ins in PLASIMO and more specifically model plug-ins are treated in chapter 5, so we refer to that chapter for details. Chapter 5 also contains a description of C++ classes, and how they are used in PLASIMO and the CRM. Here we will limit ourselves to the elements of the implementation that are directly linked to the steps the GPM should perform.

The GPM is implemented in a class derived from the `plModelBase` class. This class contains a number of data structures to store the species, reactions including the extra source terms, and variables required for the calculation. The class also implements a number of functions (as any `plModelBase` derived class should) the most important of which are the constructor and `do_update()` member function.

Steps 1 and 2 are performed by the constructor, which is automatically executed when an object of the GPM is created. Steps 3 and 4 are implemented in the `do_update()` member function. The main plasimo application is responsible for creating the GPM model object (and thereby calling the constructor so that all data structures are initialized), and for iteratively calling the `do_update()` function.

6.4.1 Model construction

The first two steps of the GPM are to construct the model and all required data structures as described in an input file. Essentially, the model *is* the set of balance equations for the species density and electron energy density. Listing 6.1 shows part of an input file, describing a simple system of three species (electrons, molecular and atomic oxygen) and a single reaction (molecular oxygen dissociation).

Since it is a plasimo input file, the same form of key-value pairs ordered in sections is used here as in the input file for the CRM (see listing 5.11). The first thing the main plasimo application does is to look for the `Type` key (see section 5.4.1), which can be found on line 1. plasimo then creates an instance of the model plug-in that is available from the model plug-in factory under the name `ZDM` which stands for "Zero Dimensional Model" (the current name of the GPM plug-in). The rest of the input file is then loaded by the constructor of the GPM plug-in.

Species and reactions are defined in separate sections: section `Species` (line 3) and `VolumeRelations` (line 32). Each species is defined in its own `Particle` section (lines 4, 13, and 22). Every species has a `Name` (e.g. line 5), under which it can be used in the definition of a reaction, a `Constitution` (e.g. line 6), describing the chemical composition, and a `State` section (e.g. line 7), to further describe the species (statistical weight, energy level). The constitution of a species is used for checking mass conservation in reactions.

Reactions are defined in `Relation` sections that contain a `Format` (line 35) and a `Rate` section (line 36), defining the rate coefficient. Rate coefficients can be defined in different forms or `Type`'s (line 37). Here we show a custom function that must be completely defined (other possibilities are a file containing a look-up table or an Arrhenius function in combination with a few parameters), and therefore also contains the expression (line 38). In addition the unit of the resulting rate coefficient is defined (line 39) as well as the constants used in the expression (lines 40 and 41). In any custom function certain predefined functions can be used to access species properties. In the rate on line 38 this is used to obtain the electron temperature. Other built-in functions are available for the energy, mass and den-

```
Type  ZDM

Species {
  Particle {
    Name O2
    Constitution O2
    State {
      Type Atom
      Weight 7
      Energy -5.16514*eV
    }
  }
  Particle {
    Name O
    Constitution O
    State {
      Type Atom
      Weight 9
      Energy 0*eV
    }
  }
  Particle {
    Name e
    State {
      Type Atom
      Weight 2
      Energy 0*eV
    }
  }
}

VolumeRelations {
  Relation {
    Name O2_dissociation
    Format "O2 + e <=> 2 O + e"
    Rate {
      Type CustomRate
      Function "C*exp(-E/(kB*Temperature(e)))"
      Unit m^3/s
      Declare C 7.1e-15
      Declare E 1.3e-18*J
    }
  }
}
```

Listing 6.1. *Part of a* plasimo *input file for a Global Model. A rudimentary system is described consisting only of an atomic and molecular oxygen species, and an electron species. In this listing only one reaction is included (dissociation).*

sity. In case of the density and temperature the values are for the current time of the model*.

As outlined in chapter 5, one of the key features of plasimo is its modular design. Some of the available functionality of plasimo is in the area of chemistry, for which plasimo provides classes to store species and reactions. Those classes are also used in this model plug-in; only the sections containing the species and reactions need to be passed to the constructors of these classes, after which all information contained in the definitions of the species and reactions is available through member functions. The species information is stored in a class called `plParticleMap` and the reactions in classes called `plParticleRelation`. Apart from simply storing the reaction specifics, the `plParticleRelation` class also provides member functions to calculate the rates. This demonstrates the benefit of plasimo's modular design; through the use of classes and Self Registering Objects†(SRO) code used in (and often designed for) certain plasimo modules can easily be used in other modules. This modular design even transcends the programming level; the sections of an input file describing the chemistry (species and reactions) can, without alteration, directly be used in other plasimo models, for instance in fluid models that also contain a certain chemistry.

For dealing with the extra source terms (see section 6.3.2) a dedicated class is defined. Similar to the `plParticleRelation` class it provides a member function to calculate the rate. We will not elaborate on the specifics in this text.

Apart from the structures for the species and relations, the constructor of the GPM plug-in prepares matrix $\boldsymbol{W}$ (see Eqn. (6.11)), and vector $\mathbf{E}$ (see Eqn. (6.15)).

Now that all functionality is in place to construct the balance equations we can continue with the solution procedure.

6.4.2 Solution procedure

We need to solve the combination of Eqns. (6.11) and (6.15), which is an initial value problem. To this end we employ the exact same solution procedure as was used for the time dependent version of the CRM (see section 5.5.2): the `plStepper` class. The solution procedure is iterative, and performed by the `do_update()` member function of the GPM model plug-in, mentioned at the start of this section. Every time the `do_update()` function is called one time step is taken by the stepper, until the desired end time of the solution is reached.

The stepper only requires access to a function $\mathbf{f}$, see Eqn. (5.1), that provides the source vector. This function is implemented as a member function of the GPM model plug-in. In the CRM the source vector was simply calculated by multiply-

* Since only the energy balance for the electrons is included in the model, only the temperature of the electron species is time dependent. Furthermore, all non-electron species are assumed to have the same constant (heavy particle) temperature.

† See section 5.3.4 for a short explanation of this mechanism.

ing the transition matrix by the density vector: $\mathbf{S} = \boldsymbol{F}\mathbf{n}$. Here we require a few more steps:

1. calculate the rate vector $\mathbf{R}$;
2. calculate the source vector for the density balances by multiplying the stoichiometry matrix by the rate vector: $\mathbf{S} = \boldsymbol{W}\mathbf{R}$;
3. calculate the source vector for the electron energy density balance by summing the input power density and inelastic energy density transfer: $Q_{\text{inelas}} = \mathbf{E}_e \cdot \mathbf{R}$;
4. add any extra source terms to the source vector.

The resulting vector is provided to the stepper so it can calculate the results for the next time step.

Just as the densities and electron energy density, the rates that are calculated every time step are stored. The rates as function of time can then be used to analyze the dominant reactions in a system.

6.5 Conclusion

The GPM module is very similar to the time dependent CRM of chapter 5, though it is aimed at accommodating more general models. Whereas the CRM only dealt with atoms and linear processes effectuated by external agents, the GPM allows for any defined species to react with each other in any way, so we are not restricted to linear processes*. In addition, the electron energy density is solved, though, in contrast to the CRM, the GPM can only handle Maxwellian EEDFs. Great flexibility is achieved by allowing for many parameters to have a predefined time dependence and by the use of freely definable extra source terms.

Since many building blocks provided by plasimo can be used in the construction of the module, its implementation is relatively straightforward. The most notable of these building blocks are tools for dealing with the chemistry, but also the stepwise solver.

In chapter 9 the GPM will be used to investigate the ionization mechanisms in a High Power Impulse Magnetron Sputtering (HiPIMS) plasma.

* Note that in a limited way the CRM does allow for non-linear processes to be included.

Part II

Applications

Chapter 7

A CRM of EUV induced plasmas

7.1 Introduction

In the future Extreme UltraViolet (EUV) radiation will be used in the manufacturing process of Integrated Circuits (ICs). A consequence of this is that reflective optics must be used in the imaging mechanism in lithography machines. In these machines the EUV radiation will interact with the background gas and form a plasma. In the vicinity of a surface, and thus a mirror, energetic ions will accelerate to the wall and potentially sputter the surface. Since this can be detrimental to the reflectivity of the mirror it is of vital importance to mitigate this process. Previous research showed that the energy of the electrons (i.e., the Electron Energy Distribution Function or EEDF) in the EUV induced plasma is responsible for the energy the ions can reach, and that this energy can be lowered by increasing the pressure of the background gas. The aim of this chapter is to corroborate the faster decay of the EEDF by increased background pressure by the use of time resolved Optical Emission Spectroscopy (OES). However, to interpret the measurements it is necessary to construct a Collisional Radiative Model (CRM).

In section 7.2 we will start with some background information about the issue at hand and the EUV induced plasma. We will then continue (section 7.3) with the CRM describing the considerations in its construction. It will turn out that the CRM requires the EEDF from an external source and that this EEDF will be far from Maxwellian. In section 7.4 a Particle In Cell Monte Carlo (PIC-MC) model is described that can provide the EEDF, but also a simple analytical model will be brought into play. The results, also comparing the PIC-MC model and analytical model, are given in section 7.5. Finally, time resolved OES measurements are

described in section 7.6, where they are also compared to the results of the CRM.

7.2 Background

7.2.1 Lithography

An important step in the manufacturing of ICs is the imaging of a *mask* onto a substrate by means of a complex set of lenses. The mask is a geometric pattern that determines whether or not a photoresist layer on the substrate is removed by the light it is exposed to. When the substrate is subsequently etched, only parts on the substrate that are still covered by the photoresist will be (partially) removed. By repeating this process a number of times structures are formed that make up electronic circuits: the IC.

To create faster and more energy efficient computer chips the structures have to be made smaller. The past five decades this has, among others, been achieved by using light with shorter wavelengths for imaging the mask. Presently, a wavelength of $\lambda = 193\,\text{nm}$ is the state of the art, but driven by the urge to go to even smaller structures one has to look for sources producing radiation with a much lower wavelength.

The company ASML, being the world's leading developer and supplier of lithography systems, aims to use EUV radiation in its next generation of machines. EUV lithography (EUVL) will use radiation in a 2% bandwidth around a central wavelength of 13.5 nm. Using radiation at this wavelength has two important consequences:

1. Due to its short attenuation length (100 nm) it is no longer possible to use lenses for the optical system.

2. To prevent background gas from absorbing the radiation, the whole system must be kept in vacuum at a pressure of 0.1 Pa to 1 Pa.

Most mirrors in a EUVL tool are (near) normal incidence multilayer mirrors. They are made up of multiple Mo-Si bi-layers, and by selecting the right thicknesses a reflectivity of up to 70% can be reached. However, the projection path inside a commercial lithography machine can require up to 10 mirrors, resulting in a total transmission of only $(0.7)^{10} \approx 2.8\%$. This means that a small decrease in reflectivity of each mirror will add up to a significant transmission loss. It is therefore essential to prevent anything from adversely affecting the reflectivity of the multilayer mirrors.

The EUV photons have enough energy (92 eV) to photoionize the background Ar gas. This will create free electrons and positive Ar ions, i.e., a plasma. The electrons are much more volatile than the ions due to their lower mass. Near a wall (or mirror) the electrons will hit the wall and remain there, whereas the ions

will hardly move (on very short timescales). However, the charge imbalance will result in a potential drop towards the wall, which in turn accelerates the positive ions in the direction of the wall. If the ions gain enough kinetic energy before they hit the wall it is possible that they can "knock" away atoms from the mirror surface, a process called *sputtering*. If enough atoms are sputtered the mirror will be damaged and its reflectivity will drop.

7.2.2 Previous research

Sputtering of the mirror surface was the subject of previous research by Van der Velden [53]. In that research a numerical model was used to simulate the creation of a plasma by EUV radiation, and the interaction of that plasma with a (mirror) surface. Also, experiments at the EUV-lab of ASML were performed to measure the sputter rate using a pulsed EUV source. Some of the conclusions of that research are:

- The simulations show that indeed a potential drop towards the mirror surface is created. This potential drop accelerates Ar ions towards the surface, so that sputtering may occur. The potential drop is determined by the mean energy of the electrons in the plasma.
- Sputtering is mainly caused by the Ar^{2+} ions that are accelerated towards the wall and to a lesser extent by the Ar^{+} ions. This is because the double charged ions gain twice the energy of single charged ions.
- When the pressure of the background gas is increased, the sputter rate also increases because more plasma is produced, which leads to more ions impacting on the mirror surface. However, because the inelastic collision frequency of the plasma electrons with the background gas also increases with pressure, the plasma electrons are cooled. The result of this is that the potential drop towards the mirror is reduced. The ions hitting the wall will have less energy so that less sputtering will occur. Between these two opposing effects there is an optimum.
- The performed experiments did not show any sputtering. This does not, however, contradict the model, since the model predicts that the sputtering would be too low to be detected ($\sim$ 1 nm per 10^9 EUV pulses) given the experimental parameters.

The key to controlling the sputtering is the mean electron energy of the electrons in the EUV induced plasma. If the mean electron energy can be kept low enough the potential drop towards the wall can be controlled, so the ions in the plasma will not cause sputtering of the mirror surface. This can be accomplished by increasing the background Ar pressure.

It is desirable to somehow monitor the effect these measures have on the EUV driven plasma, more specifically on the energy of the plasma electrons. The numerical model by Van der Velden was designed to model the sputtering of a surface near an EUV induced plasma. The aim of this chapter is to experimentally validate that model, more specifically the part of the model that describes the creation and evolution in time of the EUV produced plasma. The diagnostic technique that is used on the plasma is optical emission spectroscopy. In order to validate the model by Van der Velden with these measurements, we must first construct a collisional radiative model of the EUV driven plasma.

7.2.3 The EUV induced plasma

The plasma created by photoionization of the Ar background gas by EUV photons was extensively described by Van der Velden in his thesis [53]. Here we will only give a summary highlighting the most relevant aspects.

The EUV source (extensively studied by Kieft [26]) produces EUV radiation in 100 ns pulses at a rate of up to 1 kHz. The radiant energy* per square meter per pulse is called the radiant exposure, which is assumed to be $H_e^{\text{pulse}} = 0.6\,\text{J}\,\text{m}^{-2\dagger}$.

For an Ar background pressure of 1 Pa the density is $n_a \approx 2.4 \times 10^{20}\,\text{m}^{-3}$ at room temperature. The electron density due to photoionization can be approximated using the expression:

$$n_e = \frac{H_e^{\text{pulse}}\left[1 - \exp(-L n_a \sigma_{\text{photo}})\right]}{h\nu L} \approx \frac{H_e^{\text{pulse}} n_a \sigma_{\text{photo}}}{h\nu} \approx 10^{15}\,\text{m}^{-3}, \tag{7.1}$$

where the photoionization cross section is $\sigma_{\text{photo}} = 10^{-22}\,\text{m}^2$ [62], and the energy of a photon is $h\nu = 92\,\text{eV}$, and where we have used the fact that the plasma is optically thin ($L n_a \sigma_{\text{photo}} \ll 1$, for a typical length of $L = 10\,\text{cm}$).

Double and triple photoionization are also possible, but since the cross sections for these processes are much smaller, they are not of influence.

Every EUV pulse will create a plasma. The 100 ns pulses lie 10^{-1} s to 10^{-3} s apart, and in between pulses the plasma will decay. Electrons and ions can be removed by two-electron recombination:

$$Ar^+ + e + e \overset{K_{\text{rec}}}{\rightarrow} Ar + e, \tag{7.2}$$

which using for the rate [63]:

$$K_{\text{rec}} = 3.3 \times 10^{-21} (T_e)^{-9/2}, \tag{7.3}$$

* Within a 2% bandwidth around the center wavelength of 13.5 nm.

† This is a realistic value for the EUV source that was used for the experiments performed by Van der Velden and the experiments in section 7.6.

results in a recombination frequency of $\nu_{\text{rec}} = K_{\text{rec}} n_e^2 \approx 2 \times 10^{-7}\,\text{Hz}$ at $\hat{T}_e = 10\,\text{eV}^{*\dagger}$. Compared to the EUV pulse frequency this is clearly negligible.

The electrons and ions will spread by diffusion. The ambipolar diffusion is (see Eqn. (2.9)):

$$D_a = \frac{2}{3 n_{\text{Ar}} \sigma_{ia}^{\text{elas}}} \sqrt{\frac{k_B T_h}{\pi m_{\text{Ar}}}} \left(1 + \frac{T_e}{T_h}\right) \approx 130\,\text{m}^2\,\text{s}^{-1}, \tag{7.4}$$

where we have used $\sigma_{ia}^{\text{elas}} \approx 10^{-18}\,\text{m}^2$ taken from [64]. For a diffusion distance of $L = 5\,\text{cm}$ this results in a decay time of:

$$\tau_a = \frac{L^2}{D_a} = 2 \times 10^{-5}\,\text{s}. \tag{7.5}$$

So, realizing that the repetition frequency of the pulsed EUV source is 1 kHz, the diffusion is fast enough to let the plasma decay in the period between two subsequent pulses.

Strictly speaking, the formulae using the electron temperature T_e only apply to plasmas with a Maxwellian electron energy distribution function (EEDF) (see also footnote †). The EEDF will initially be mono-energetic with a peak at 76 eV (the energy with which the free electrons are introduced in the plasma by photoionization). Due to collisions between electrons, the EEDF will then relax towards a Maxwell distribution. The collision time for electrons with electrons can be estimated by [65]:

$$\tau_{ee} = \frac{6\sqrt{2}\pi^{\frac{3}{2}} \varepsilon_0^2 \sqrt{m_e} (k_B T_e)^{\frac{3}{2}}}{e^4 n_e \ln \Lambda}, \tag{7.6}$$

with the Coulomb parameter:

$$\Lambda = \frac{12\pi\varepsilon_0 (k_B T_e)^{\frac{3}{2}}}{\sqrt{n_e} e^3}. \tag{7.7}$$

Using $\hat{T}_e = 10\,\text{eV}$ again, we find for this time approximately $\tau_{ee} \approx 4 \times 10^{-4}\,\text{s}$. This is much longer than the diffusion time of the plasma, so before the energy of the electrons can be redistributed the plasma has already recombined at the wall.

7.2.4 Optical emission spectroscopy

The EUV driven plasma has a low electron density, is short-lived, relatively small, and only present where there is EUV radiation. This makes it very difficult to apply diagnostic methods. A study was performed by Van der Velden [53] to assess

* Temperatures with a "hat" denote that the temperature is expressed in electronvolts (1 eV $\approx$ 11 604.5 K).

† Use of the electron temperature implies that the EEDF is Maxwellian, which it certainly is not. Here it is used to represent the mean energy of the electrons.

the feasibility of several techniques, such as Langmuir probing, Thomson Scattering (TS), Microwave Interferometry (MI), Laser Induced Fluorescence (LIF), and Optical Emission Spectroscopy (OES). All but the last method were deemed unsuitable (TS, MI, and LIF) or turned out to be impossible (Langmuir probing).

In this study OES is applied. In the EUV driven plasma, excited atom levels will mostly be populated by ground state electron excitation, and depopulated by radiative decay. This production–destruction mechanism is the so called *corona balance*.

For an atomic level the density Eqn. (3.17) will then reduce to:

$$\frac{\partial n(p)}{\partial t} = \underbrace{n_e n(1) K(1,p)(t)}_{\text{electron excitation from ground state}} + \underbrace{\sum_{q>p} n(q)A(q,p)}_{\text{cascade}} - \underbrace{\sum_{p>q} n(p)A(p,q)}_{\text{spontaneous emission}} \quad , \tag{7.8}$$

with $n(p)$ and $n(q)$ the densities of (two separate) excited levels, $n(1)$ the ground level density, $K(1,p)$ the electron induced ground state excitation rate coefficient, and $A(p,q)$ the Einstein coefficient for spontaneous emission from level p to q. Note that levels are not only populated by electron collisions from the ground level, but also by radiative decay from higher levels; the *cascade contribution*.

To interpret the results of OES measurements, knowledge of the collisional excitation rates is required. The rate coefficient is the time dependent version of Eqn. (4.9):

$$K(1,p)(t) = \int_{E_{\text{th}}}^{\infty} \sigma_{1,p}^{\text{exc}}(E') f(E',t) v_e(E') dE, \tag{7.9}$$

with $f(E,t)$ the time dependent EEDF, $\sigma_{1,p}^{\text{exc}}(E)$ the cross section for the ground state excitation of level p by electron impact, and $v_e(E)$ the electron velocity. Note that the electrons need at least a threshold energy $E_{\text{th}} = E(p)$ to excite the atom to level p. Many cross sections are available from literature. These can be divided into experimental, theoretical, and semi-empirical data. Apart from cross section data we need a time dependent EEDF which must be provided by a model.

Van der Velden [53] has modeled the EUV driven plasma using a Particle In Cell Monte Carlo (PIC-MC) method, which will be treated in section 7.4.1. Among other things this model calculates the electron densities $n_e(t)$ and distribution functions $f(E,t)$ during the evolution of the plasma. To gain insight into the creation of the EEDF we will investigate them separately in section 7.4.

Using these EEDFs as input, a CRM can be constructed to model the radiation from the plasma. The model, that will be introduced in the next section, will solve a coupled set of equations in the form of (7.8). When the (time dependent)

densities and transition probabilities are known, the radiation from the plasma can be calculated and compared to experimental results.

OES measurements can be performed using standard spectrometers. Because the EUV driven plasma is highly transient in nature it is desirable to perform time resolved measurements. An added difficulty is the low photon yield per pulse. The time resolved measurements, described in section 7.6, were performed using a fast photo multiplier tube attached to a monochromator.

For spectroscopic measurements it is important to know to what extent the radiation from the plasma is absorbed before it reaches the detector.

The absorption cross section is defined by:

$$\sigma(\nu) = \frac{1}{8\pi}\lambda^2 A(u,l)\phi_\nu(\nu)\frac{g_u}{g_l}, \tag{7.10}$$

with λ the wavelength of the light being absorbed, $A(u,l)$ the Einstein coefficient for spontaneous emission from level u to level l, g_u and g_l the degeneracies of levels u and l, and $\phi_\nu(\nu)$ the line form function. This function is normalized so that:

$$\int \phi_\nu(\nu)d\nu = 1, \tag{7.11}$$

where the integration is performed over the whole transition. For $\phi_\nu(\nu)$ we will use a block form function with a width of (Doppler broadening):

$$\delta\nu = \nu_0\sqrt{\frac{k_B T}{m_{\mathrm{Ar}}c^2}}, \tag{7.12}$$

so that $\phi_\nu(\nu) = 1/\delta\nu$.

We are interested in radiation in the optical range and in Ar the most intense lines will be those of the 4p-4s transitions. As an example we will use the line at 811.8 nm, with $A = 3.3 \times 10^7\,\mathrm{s}^{-1}$ and $g(u)/g(l) = 7/5$. The Doppler broadening is $\delta\nu \approx 3.1 \times 10^8\,\mathrm{s}^{-1}$ so $\phi_\nu(\nu) \approx 3.2 \times 10^{-9}\,\mathrm{s}$. The cross section will then be $\sigma(\nu) \approx 4 \times 10^{-15}\,\mathrm{m}^2$. Since we can expect densities of the excited levels in the order of $10^{14}\,\mathrm{m}^{-3}$, the absorbed fraction over a length of $L = 1\,\mathrm{cm}$ is $nL\sigma \approx 4 \times 10^{-3}$, so we can conclude that absorption is not an issue*.

7.3 CRM

7.3.1 Non-equilibrium

In general we may state that any plasma has (several) equilibrium and non-equilibrium aspects. The degree of equilibrium departure is expressed by the ratio

* For transitions from the resonant 4s levels to the ground state absorption can be significant, due to the high transition probability and high lower level (ground) density. But that will not be of influence.

between the equilibrium and non-equilibrium parts. It can be described by dividing the plasma kinetics in forward and *corresponding* backward processes; so-called *proper* balances. Equilibrium is present on a detailed plasma aspect when the number of forward processes equals that of the corresponding backward processes. Thus when the proper balances in question equilibrate. Since electrons are, in many situations, the main agents in performing (de-)excitation kinetics we can expect that for increasing n_e more proper balances will reach a state of equilibrium. The underlying aspects of the partial equilibrium features can easily be described with a few parameters using the well-known laws of statistical mechanics. In our case, however, due to the low electron density ($n_e \approx 10^{15}\,\mathrm{m}^{-3}$) and strong transient nature with a decay time of typically 10^{-7} s, corresponding to 10^7 Hz, the plasma is far from equilibrium. We can examine some (disturbed) equilibria in more detail:

- Planck:
  The underlying proper balance is in this case given by absorption and (stimulated) emission. As is shown in the previous section, the attenuation length for radiation generated by the plasma in the visible and UV-range exceeds the dimensions of the plasma by far. The plasma is therefore optically thin, which implies that the radiation generated by the plasma can not be characterized by Planck's radiation law.

- Maxwell:
  Here the proper balance is formed by the kinetic energy exchange between electrons in e-e collisions. The photon beam creating the plasma has a very distinct energy (2% bandwidth around 13.5 nm) so initially the EEDF will be sharply peaked at 76 eV. By e-e collisions the plasma might thermalize reaching a Maxwellian EEDF. However, from Eqn. (7.6) we know that the average time between collisions is longer than the diffusion time, so the plasma will have recombined at the wall before Maxwellization can take place.

- Boltzmann:
  It is possible to describe the densities of the excited states using the Boltzmann distribution law, if at least two conditions are fulfilled. First, the energy of the agents performing the excitation and de-excitation are distributed according to a Maxwell distribution. Second, the relevant excitation and de-excitation processes are in equilibrium with each other. Since, in our case, there is no Maxwell distribution, Boltzmann can not be used to describe the excited state populations.

- Saha:
  In order for the Saha distribution law to be applicable, the balance of ionization and the corresponding reverse process of two-electron recombination

must be in equilibrium:

$$Ar + e \leftrightarrow Ar^+ + e + e$$

In section 7.2.3 the two-electron recombination frequency was estimated to be $\nu_{\text{rec}} \approx 2 \times 10^{-7}$ Hz. This is much lower than the diffusion frequency of $\nu_a = 1/\tau_a \approx 5 \times 10^4$ Hz (see Eqn. (7.5)). This means that the ions will have drifted away before an equilibrium can be established, so also Saha equilibrium is out of the question.

The foregoing considerations show that the well known equilibrium laws, as given by statistical mechanics, can not be used to describe our plasma, so we are dealing with a very unusual plasma. This means, that in order to find the light emission as a function of time, we have to describe the non-equilibrium reactions in detail. To that end we construct a collisional radiative model (CRM), consisting of relevant balance equations.

7.3.2 Classification of excitation balances

To model the plasma it is necessary to construct a set of balance equations that describe the relevant mechanisms in the plasma. In our case the plasma is ionizing, meaning that the excited levels are populated by excitation from the ground level and not by the continuum. The electron density is relatively low, so it is to be expected that lower excited levels can be described with the Corona Balance (CB) and higher levels using the Excitation Saturation Balance (ESB). The principles of both balances are shown in Fig. 7.1.

Levels with a lower principal quantum number ($p < p_{\text{cr}}$) are in the CB and states with a higher number ($p > p_{\text{cr}}$) in the ESB. The boundary level p_{cr} is calculated using Eqn. (3.34):

$$p_{\text{cr}} \approx \left(9 \times 10^{23} \frac{Z^7}{n_e}\right)^{1/9} \approx 8, \tag{7.13}$$

which is higher than any of the observable lines (spectroscopy measurements are treated in section 7.6). We can therefore conclude that the system is dominated by the CB.

The system is described by a set of coupled balance equations each having the form of Eqn. (7.8), i.e. population by electron excitation of the ground state and radiative decay from higher levels (cascade contribution), and depopulation by radiative decay.

The dominant lines observed in the spectroscopic measurements are those of the 4p-4s transitions. These transitions have a typical decay time of $\tau = 10^{-7}$ s, meaning that we can not speak of a usual CB; the CB we deal with is highly transient.

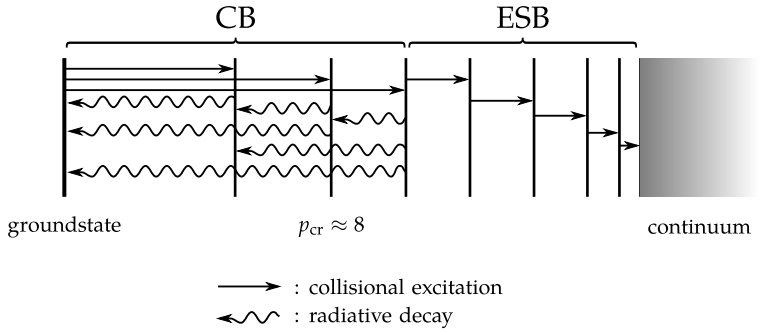


Figure 7.1. *Schematic depiction of the Corona Balance (CB) and Excitation Saturation Balance (ESB). In CB atomic levels are populated by means of electron excitation of the ground state, whereas depopulation is realized by means of spontaneous radiative decay to lower lying levels. In ESB the atoms are repetitively excited until the excited electron reaches the continuum and the atom is ionized. The critical principal quantum number for the boundary between the two balance domains, is designated by p_{cr}.*

7.3.3 CRM construction

The model is implemented using the time dependent CRM plug-in for the plasimo framework that was extensively described in chapters 4 and 5. Although it was shown by Van der Velden that Ar^{2+} ions are the main source of sputtering, only ArI and ArII lines could be identified in the measured spectra. Therefore only atomic and single ionized Ar are included in the model.

For constructing the CRM several data are required:

- electron cross sections for ground state excitation;
- transition probabilities;
- electron density as function of time;
- EEDF as function of time.

The last two items have to be taken from an external model, and will be described in detail in section 7.4. Cross sections and transition probabilities are taken from literature. Due to the highly transient nature of the plasma, with timescales in the order of 10^{-7} s, a selection of levels has been made: only levels that decay with a transition probability of more than $2 \times 10^6\,\mathrm{s}^{-1}$ are included. This implies that only for these levels radiative transition probabilities and cross sections are taken into account.

The database of the National Institute of Standards and Technology [66] was used for the radiative transition probabilities. The excited neutral Ar levels that

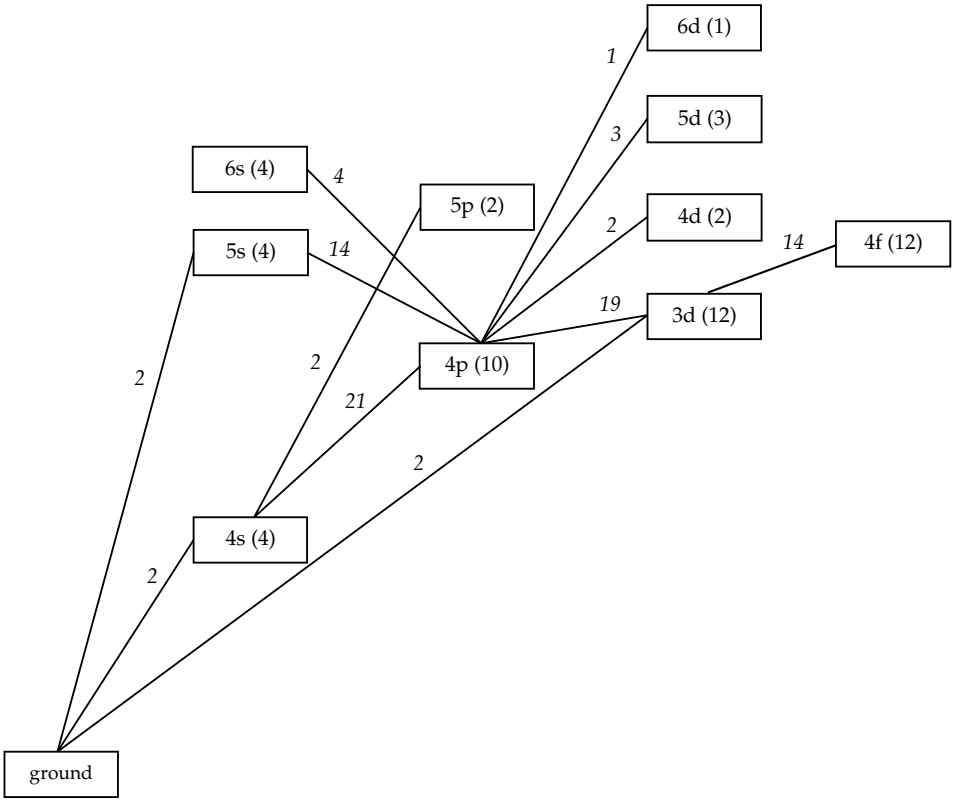


Figure 7.2. *Diagram of the levels of the atomic Ar system, ArI, that are included in the CRM. The rectangles represent blocks of levels. The number between parentheses is the number of levels the block consists of, the numbers in italics denote the number of separate radiative transitions between blocks.*

are included are listed in table 7.1*, and the diagram in Fig. 7.2 shows how the levels decay. Table 7.2 lists at which wavelengths these levels radiate and the corresponding transition probabilities. The levels to which the other levels radiate and the accompanying transition probabilities are listed in tables 7.3, 7.4, 7.5, and 7.6.

For every level that radiates, and is included in the CRM, we also need an electron excitation cross section from the ground state. For this we use a compilation of Ar cross section made by Yanguas-Gil et al. [67]. They list cross sections for the 4s, 4p, 3d, 5s, 5p, and 4d levels that are included here. The sources of these cross sections are Khakoo et al. [68] (4s), Chilton et al. [69] (4p), Hayashi [70] (3d, 5s), Weber et al. [71] (5p), and Drawin [35] (4d, 6s). Additional cross sections for 6s,

*In this text both Paschen and Racah notation will be used. To distinguish the two, Paschen notation is in italic font and Racah in normal font. See also appendix B for translation tables.

Paschen	Racah	energy [eV]	Paschen	Racah	energy [eV]
$1s_5$	$4s[3/2]_2$	11.548	$1s_3$	$4s'[1/2]_0$	11.723
$1s_4$	$4s[3/2]_1$	11.624	$1s_2$	$4s'[1/2]_1$	11.828
$2p_{10}$	$4p[1/2]_1$	12.907	$2p_5$	$4p[1/2]_0$	13.273
$2p_9$	$4p[5/2]_3$	13.076	$2p_4$	$4p'[3/2]_1$	13.283
$2p_8$	$4p[5/2]_2$	13.095	$2p_3$	$4p'[3/2]_2$	13.302
$2p_7$	$4p[3/2]_1$	13.153	$2p_2$	$4p'[1/2]_1$	13.328
$2p_6$	$4p[3/2]_2$	13.172	$2p_1$	$4p'[1/2]_0$	13.480
$3d_6$	$3d[1/2]_0$	13.845	$3d'_1$	$3d[5/2]_3$	14.099
$3d_5$	$3d[1/2]_1$	13.864	$3d''_1$	$3d[3/2]_1$	14.153
$3d'_4$	$3d[3/2]_2$	13.903	$3s''''_1$	$3d'[5/2]_2$	14.214
$3d_4$	$3d[7/2]_4$	13.979	$3s'''_1$	$3d'[3/2]_2$	14.234
$3d_3$	$3d[7/2]_3$	14.013	$3s''_1$	$3d'[5/2]_3$	14.236
$3d_2$	$3d[5/2]_2$	14.063	$3s'_1$	$3d'[3/2]_1$	14.304
$2s_5$	$5s[3/2]_2$	14.068	$2s_3$	$5s'[1/2]_0$	14.241
$2s_4$	$5s[3/2]_1$	14.090	$2s_2$	$5s'[1/2]_1$	14.255
$3p_5$	$5p[1/2]_0$	14.576	$3p_1$	$5p'[1/2]_0$	14.738
$4d_6$	$4d[1/2]_0$	14.694	$4d_5$	$4d[1/2]_1$	14.711
$3s_5$	$6s[3/2]_2$	14.839	$3s_3$	$6s'[1/2]_0$	15.014
$3s_4$	$6s[3/2]_1$	14.848	$3s_2$	$6s'[1/2]_1$	15.022
$4X$	$4f[3/2]_1$ $4f[3/2]_2$	14.901	$4U$	$4f[7/2]_3$ $4f[7/2]_4$	14.909
$4V$	$4f[9/2]_5$ $4f[9/2]_4$	14.904	$4W$	$4f'[7/2]_3$ $4f'[7/2]_4$	15.083
$4Y$	$4f[5/2]_3$ $4f[5/2]_2$	14.907	$4Z$	$4f'[5/2]_3$ $4f'[5/2]_2$	15.083
$5d_6$	$5d[1/2]_0$	15.101	$5d'_4$	$5d[7/2]_4$	15.131
$5d_5$	$5d[1/2]_1$	15.118			
$6d_6$	$6d[1/2]_0$	15.313			

Table 7.1. *The levels included in the CRM, listed in both Paschen and Racah notation, and the level energy in eV.*

level	wavelength [nm] (A [10^6 s^{-1}])		
$2p_1$	750.6 (44.5)		
$2p_2$	696.7 (6.4)	772.6 (11.7)	826.7 (15.3)
$2p_3$	706.9 (3.8)	738.6 (8.5)	841.1 (22.3)
$2p_4$	795.0 (18.6)	852.4 (13.9)	
$2p_5$	751.7 (40.2)		
$2p_6$	763.7 (24.5)	800.8 (4.9)	922.7 (5.0)
$2p_7$	772.6 (5.2)	810.6 (25.0)	867.0 (2.4)
$2p_8$	801.7 (9.3)	842.7 (21.5)	
$2p_9$	811.8 (33.1)		
$2p_{10}$	912.5 (18.9)	966.0 (5.4)	

Table 7.2. *The wavelengths at which the 4p levels radiate and in parentheses the transition probabilities.*

upper level	lower levels (A [10^6 s^{-1}])				
$2s_2$	*0* (35.0)	$2p_2$ (3.4)	$2p_3$ (8.9)	$2p_4$ (2.0)	
$2s_3$	$2p_2$ (5.1)	$2p_4$ (10.0)	$2p_7$ (2.2)	$2p_{10}$ (3.3)	
$2s_4$	*0* (77.0)	$2p_6$ (2.7)	$2p_7$ (4.6)	$2p_8$ (8.9)	$2p_{10}$ (2.4)
$2s_5$	$2p_6$ (3.3)	$2p_9$ (11.0)	$2p_{10}$ (4.9)		

Table 7.3. *The levels to which the 5s levels radiate (0 denotes the ground level) and in parentheses the transition probabilities.*

5d, and 6d are taken from Vlček [38] in the form of Drawin cross section parameters. The cross sections for the 4f levels are taken from another article by Chilton et al. [72]. The cross sections in that article are *apparent* cross sections, meaning that they include the cascade contribution from higher levels. Since the NIST database does not list any probabilities for transitions into the 4f levels, we assume that the cascade is negligible and use the apparent cross sections without correction. The relation between *direct*, *apparent*, and *optical* cross sections is explained in Fig. 7.3.

Several of the cross sections in the literature sources are for combined levels or level blocks, such as the 3d and 5s levels, the 4f levels, and all levels described by Drawin cross sections. The cross sections for the separate levels are obtained by splitting the cross section according to their degeneracy:

$$\sigma_i = \frac{g_i}{\sum_{n \in B} g_n} \sigma_B, \tag{7.14}$$

upper level	lower levels (A [10^6 s^{-1}])			
$3s'_1$	*0* (313)	$2p_1$ (5.2)	$2p_2$ (7.1)	$2p_4$ (4.5)
$3s'''_1$	$2p_2$ (6.2)	$2p_6$(3.8)		
$3s'''_1$	$2p_6$ (15.0)			
$3s''''_1$	$2p_6$ (2.2)	$2p_7$ (13.0)		
$3d'_1$	$2p_8$ (2.0)	$2p_9$ (3.1)		
$3d''_1$	$2p_7$ (7.3)	$2p_8$ (5.7)		
$3d_2$	*0* (270)	$2p_5$ (4.3)	$2p_7$ (11.0)	
$3d_3$	$2p_6$ (2.5)	$2p_{10}$ (4.9)		
$3d_4$	$2p_8$ (11.0)			
$3d_5$	$2p_{10}$ (7.4)			
$3d_6$	$2p_{10}$ (8.1)			

Table 7.4. *The levels to which the 3d levels radiate (0 denotes the ground level) and in parentheses the transition probabilities.*

upper level	lower level (A [10^6 s^{-1}])	upper level	lower levels (A [10^6 s^{-1}])	
4U	$3d'_4$ (6.5)	*4X*	$3d_5$ (4.6)	
4V	$3d'_1$ (5.4)	*4Y*	$3d_2$ (5.9)	$3d_3$ (3.5)
4W	$3s'''_1$ (9.0)	*4Z*	$3s''_1$ (5.3)	$3s'_1$ (7.7)

Table 7.5. *The levels to which the 4f levels radiate, and in parentheses the transition probabilities. Note that every level shown here actually consists of two separate levels that lie very close to each other.*

where g is the weight of a level, while the sum is taken over the weights of all levels included in block B.

The simple corona-balance model (i.e., excitation solely from the ground state) is extended to the ion system of Ar (ArII). For the excited ions, however, there are two possible population routes:

- excitation from the ion ground state, or
- simultaneous ionization and excitation from the neutral ground state.

Since the EUV-pulse creates electrons with an energy of up to 76 eV, the second process, requiring 32 eV to 38 eV, is certainly possible. Which of the two processes is most important can be determined by comparing the cross sections. Excitation

upper level	lower level (A [10^6 s^{-1}]	upper level	lower level (A [10^6 s^{-1}]
$3s_5$	$2p_3$ (6.5)	$4d_6$	$2p_{10}$ (3.1)
$3s_4$	$2p_8$ (5.4)	$4d_5$	$2p_{10}$ (2.8)
$3s_3$	$2p_4$ (9.0)	$5d_6$	$2p_{10}$ (3.2)
$3s_2$	$2p_3$ (4.6)	$5d_5$	$2p_{10}$ (2.2)
$3p_5$	$1s_4$ (5.9)	$5d'_4$	$2p_3$ (2.5)
$3p_1$	$1s_2$ (5.3)	$6d_6$	$2p_{10}$ (2.4)

Table 7.6. *The levels to which the included 6s, 5p, 4d, 5d, and 6d levels radiate, and in parentheses the transition probabilities.*

cross sections from the ion ground state for four 4p levels were measured by Imre et al. [73], where it is also mentioned that the maximum values of these cross sections are 15 to 30 times higher than those for ionization and excitation from the neutral ground level. This fact is also mentioned in an article by Boffard et al. [74] though they indicate a factor of 5 to 30. Since the ion density is estimated to be at least four orders of magnitude lower than the atom ground state, simultaneous ionization and excitation is clearly the dominant mechanism and therefore cross sections out of the atomic ground state are needed.

Direct cross sections could not be found in literature, though *optical* cross sections for several ion lines are listed by Boffard et al. [75]. Measurements of the EUV driven plasma, that will be presented in section 7.6, show that the dominant ion lines in the spectrum belong to 4p-4s transitions. Furthermore, investigation of the cascade for the ionic 4p levels shows that higher ionic levels radiate very rapidly into the 4p levels. Transition probabilities are mostly in the order of 10^7 s^{-1} and many are over 10^8 s^{-1}. Because the cascade from higher levels is so fast it is assumed that the optical cross sections can be applied to this highly transient plasma. The optical cross sections are used in the model and no separate cascade for the ion lines is included.

To convert the optical cross sections into apparent cross sections, the branching ratios of the excited levels are required (see also Fig. 7.3). These are calculated by taking all the transition probabilities of the desired levels from the NIST database. The resulting apparent cross sections are used in the model. Only the upper levels of the eight strongest lines in the measurements are included. The levels and the main wavelengths at which they radiate, including the transition probabilities are listed in table 7.7.

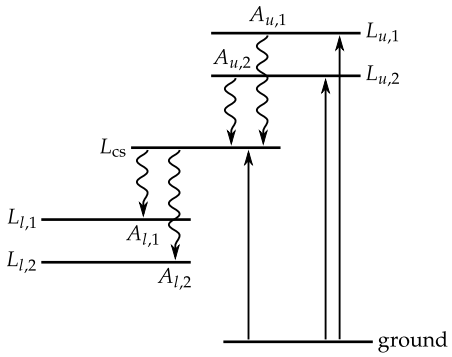


Figure 7.3. *Diagram of the radiative processes concerning a certain level L_{cs} of which the cross section is to be determined. In general a cross section is determined by exposing a gas to a mono-energetic electron beam, so all excited levels are populated from the ground state (here only shown for the upper three levels). The strength of the fluorescence signal gives the magnitude of the cross section. By performing this measurement for many different electron energies the cross section as function of energy can be determined. To determine the cross section for level L_{cs}, the fluorescence to the two lower levels $L_{l,1}$ and $L_{l,2}$ is measured. However, L_{cs} is also populated by radiative decay from the upper levels $L_{u,1}$ and $L_{u,2}$, the* cascade*. When these signals are also measured, it is possible to correct for the cascade. When the cross section is only determined by measuring a single radiative transition (e.g., to $L_{l,1}$) this is called an* optical *cross section. When the cross section is corrected for radiative transitions to other lower levels it is called an* apparent *cross section. This correction can be accomplished by either measuring all possible radiative transitions from L_{cs}, or, when all coefficients for radiative decay are known (here, $A_{l,1}$ and $A_{l,2}$), by using a correcting factor: the* branching ratio*. When only $A_{l,1}$ is measured this is for the simple case shown here: $cs_{apparent} = \frac{A_{l,1}+A_{l,2}}{A_{l,1}} cs_{optical}$ When the apparent cross section is also corrected for cascade (by simultaneously measuring $A_{u,1}$ and $A_{u,2}$) it is called a* direct *cross section.*

7.4 EEDF modeling

To complete the CRM the electron density and EEDF are needed, both as a function of time. These are found using a Particle-In-Cell Monte-Carlo (PIC-MC) model initially developed by Van der Velden [53]. In this model the creation and trajectories of electrons and ions are modeled. Since the PIC-MC model is very computationally intensive, we will also derive a much simpler analytical model for the EEDF. We will first give a short description of the PIC-MC model.

7.4.1 Particle-in-Cell Monte-Carlo model

In Particle-in-Cell modeling we tackle the overabundance of particles by introducing *super-particles*. These are particles that represent a certain amount of real

level	wavelength [nm] (A [10^6 s^{-1}])	
$4p'\,^2P^\circ_{1/2}$	413.3 (85)	447.6 (29)
$4p'\,^2P^\circ_{3/2}$	427.8 (80)	
$4p\,^2P^\circ_{3/2}$	454.6 (47)	476.6 (64)
$4p'\,^2F^\circ_{7/2}$	461.1 (79)	
$4p\,^4P^\circ_{3/2}$	473.7 (58)	
$4p\,^2D^\circ_{5/2}$	488.1 (82)	

Table 7.7. *The wavelengths at which the ArII 4p levels radiate and in parentheses the transition probabilities of the radiative transitions.*

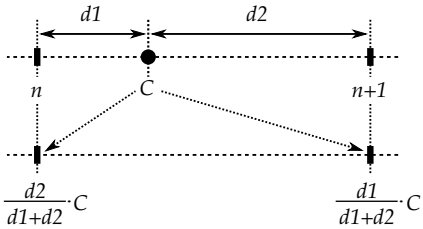


Figure 7.4. *Linear interpolation scheme as used in the PIC-MC model. Charge C is located between two nodes of the grid (n and n + 1). The charge is linearly divided between those two (closest) nodes according to the distance to the node.*

particles* (in the order of 10^9).

Even though the super-particle technique significantly reduces the number of particles involved in the calculation, there is still an enormous amount of possible interactions between particles. However, the electric (and magnetic) field at the position of every particle is required to determine its trajectory. This problem is dealt with by introducing a *grid of nodes* or particle mesh. Every node in this grid is assigned a density according to an interpolation scheme, as shown in Fig. 7.4. According to this scheme, the charge of the particles is distributed over the nodes, followed by solving the Poisson equation at the nodes. The field at the positions of the super-particles is then derived by linear interpolation.

In the Particle-In-Cell scheme the interpolation of the charges is determined with a first order scheme. Since our model is one dimensional in coordinate space, we have to divide the charge of a particle over the two closest nodes according to the distance to each node. The downside of the PIC scheme is that charge fluctuations at a scale smaller than the size of one grid cell are not resolved. This means

*The mass and charge are adjusted according to the weight.

that the grid cells must be smaller than the smallest length scale of the plasma, i.e. the Debye length. Furthermore, the time step must be smaller than the smallest time scale in the plasma, which is the inverse of the plasma frequency. An additional requirement is that the time step and size of a grid cell must be chosen such that a particle can not move more than one grid cell per time step.

The Monte-Carlo method is a generic term used for any method that uses stochastic techniques. In the PIC-MC model, these techniques are used for determining the collisions between particles:

- *when* a collision occurs;
- the *type* of collision (elastic, excitational or ionizing);
- the *outcome* of the collision.

These three aspects are determined by random numbers. The outcome of a photoionization process (by an EUV-photon) is also determined by a random number.

7.4.2 PIC-MC model of EUV driven plasma

Initially the purpose of our PIC-MC model was to model the sputtering of a surface near an EUV driven plasma. The model is one-dimensional in coordinate space and three-dimensional in velocity space. The computational domain is divided onto a grid, where initially no super-particles are present. Only ions and electrons are modeled, the Ar background gas (denoted by neutrals) is assumed to be uniformly distributed at all times.

The simulation starts when the Ar gas is exposed to an EUV-beam. The EUV photons can ionize (and double ionize) the Ar gas, so electrons and ions (in the form of test-particles) are added to the plasma while the model is running. Electrons that have enough energy can cause excitation and ionization. The latter action adds another electron-ion pair (as a couple of super-particles) to the plasma. The super-particles created by photoionization processes are placed at a random position on the grid, with a velocity in a random direction.

The model works with discrete time steps. Every time step the program performs the following procedure (also shown in Fig. 7.5):

1. At every node the charge is calculated by linear weighing;
2. The Poisson equation is solved to determine the field at the nodes;
3. The field is calculated at the position of each test-particle using linear interpolation;
4. The new position and velocity of each test-particle is determined;

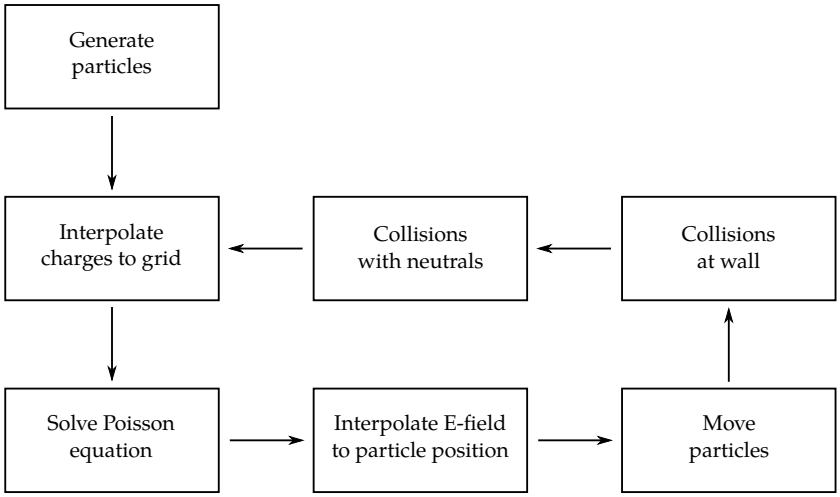


Figure 7.5. *Particle In Cell scheme. The loop is executed every time step.*

5. Test-particles that move beyond the boundaries of the grid are removed from the plasma (and the computation);
6. A Monte-Carlo routine determines whether a collision has occurred, and if so, it determines the outcome of that collision (new velocity, possibly also of the new particle).

Because it is assumed that the background density is approximately four orders of magnitude higher only collisions with Ar atoms in the ground state (neutrals) are considered:

1. elastic electron–neutral:
   $\mathrm{Ar} + \mathrm{e} \rightarrow \mathrm{Ar} + \mathrm{e}$;
2. electron excitation* of neutral:
   $\mathrm{Ar} + \mathrm{e} \rightarrow \mathrm{Ar}^* + \mathrm{e}$;
3. electron ionization of neutral:
   $\mathrm{Ar} + \mathrm{e} \rightarrow \mathrm{Ar}^+ + \mathrm{e} + \mathrm{e}$;
4. elastic ion–neutral:
   $\mathrm{Ar}^+ + \mathrm{Ar} \rightarrow \mathrm{Ar}^+ + \mathrm{Ar}$;
5. ion–neutral charge exchange:
   $\mathrm{Ar}^+ + \mathrm{Ar} \rightarrow \mathrm{Ar} + \mathrm{Ar}^+$.

*With Ar^* we refer to the excited states, that are in this model combined into a single level with an energy of 11.5 eV.

During the EUV pulse (the first 100 ns) low energy electrons are also liberated from the wall material by secondary emission. Particles that reach the wall (end of the grid) are removed from the plasma (item 5 in the procedure list). Because electrons are much faster than the heavy particles, this will result in a region with lower electron density near the wall: the sheath. Because the plasma in this region is not quasi-neutral, the ions in this region will be accelerated towards the wall. When the ions hit the wall, they can cause sputtering.

Since in this model the position and velocity of all super-particles (each representing a number of "real" particles) is known the sputtering of the wall can be studied. Whenever an ion reaches the wall, this event is recorded. Combined with empirical sputter yield data, the sputter rate of the surface can be calculated.

Modifications

Just as it is possible to follow the ions in the plasma, it is also possible to track the electrons. Not only the electron density as a function of time and position can be recorded, but also the EEDF. The measurements on the EUV driven plasma presented in section 7.6 are however not performed near a wall, but in the intermediate focus.

Analysis of the results of the PIC-MC model show that the sheath is smaller than one fifth of the plasma dimensions. It is therefore assumed that the central third of the plasma is a good representation of the plasma if no wall would be present. The data for the electron density and EEDF are taken from this region.

To speed up the simulation somewhat, processes that are not relevant for the center of the plasma can be excluded from the model. Additionally, the original model included apart from Ar also H, which was also removed.

Thus the following modifications were made to the original PIC-MC model:

- H was removed;
- secondary emission was removed;
- impact calculation of ions on the wall was removed;
- the EEDF is calculated from the electrons in the central third of the computational domain.

7.4.3 EEDF analysis

As stated before, the PIC-MC model can be used to provide the EEDF for the CRM. How the mechanisms in the PIC-MC model will result in an EEDF, and what the EEDF will look like is not evident. To understand the dynamics of the PIC-MC model a simplified analytical *ladder*-model for the EEDF is constructed.

The initial electrons in the plasma are the result of photoionization of the background Ar gas by the EUV pulse. This will result in electrons of 76.2 eV, so the EEDF will be non-Maxwellian from the start. The electrons will then lose energy by collisions with the background gas: elastic, exciting, and ionizing collisions. The electrons will lose their energy in steps, hence the name ladder-model.

The evolution of the EEDF caused by the collisions will be modeled analytically for every separate mechanism. For comparison, each of these mechanisms is also modeled separately using the PIC-MC model. Finally the EEDFs of the PIC-MC model incorporating all the mechanisms will be presented. These EEDFs (at different pressure) will be used for the CRM.

7.4.4 Elastic collisions

To justify that electrons will mainly lose their energy in (more or less) discrete steps, we will first prove that the mechanism of gradual energy loss due to elastic processes can be neglected. We therefore solve a simplified form of the electron energy equation. This equation reads:

$$n_e \frac{3}{2} k_B \frac{\partial T_e}{\partial t} = -n_e n_a k^{\mathrm{m}} \cdot \frac{3 m_e}{M} \left[k_B \left(T_e - T_a \right) \right], \tag{7.15}$$

with m_e the electron mass, M the mass of the Ar atom, and k^{m} the rate coefficient for momentum transfer. This equation can be written as:

$$\frac{1}{T_e} \frac{\partial T_e}{\partial t} = -n_a k^{\mathrm{m}} \frac{2 m_e}{M} = -\nu_{\mathrm{elas}}, \tag{7.16}$$

where the approximation $T_e - T_a \approx T_e$ has been used. This gives for the electron temperature as a function of time:

$$T_e(t) = T_e(0) \cdot \exp\left(-\nu_{\mathrm{elas}} t\right) \tag{7.17}$$

The cross sections that are used in the PIC-MC model are shown in Fig. 7.6. At 76.2 eV we find:

$$k^{\mathrm{m}} = \sigma_{\mathrm{elas}} v \approx 2.5 \times 10^{-20} \cdot 6.3 \times 10^{6} = 1.6 \times 10^{-13}\,\mathrm{m^3\,s^{-1}}. \tag{7.18}$$

Taking $n_a = 2.4 \times 10^{20}\,\mathrm{m^{-3}}$, this gives for the frequency of elastic decay the value $\nu_{\mathrm{elas}} \approx 10^3\,\mathrm{s^{-1}}$. Compared to the time of an EUV-pulse (100 ns) it is clear that elastic processes can not be held responsible for the cooling of electrons.

Note, that Eqn. 7.15 is based on the assumption that a certain temperature can be assigned to the electrons. This is of course not the case in view of the deviation from a Maxwellian energy distribution function (see discussion in section 7.3.1). However, the conclusion that the loss of energy during the EUV pulse due to elastic collisions can be neglected, will remain unaltered.

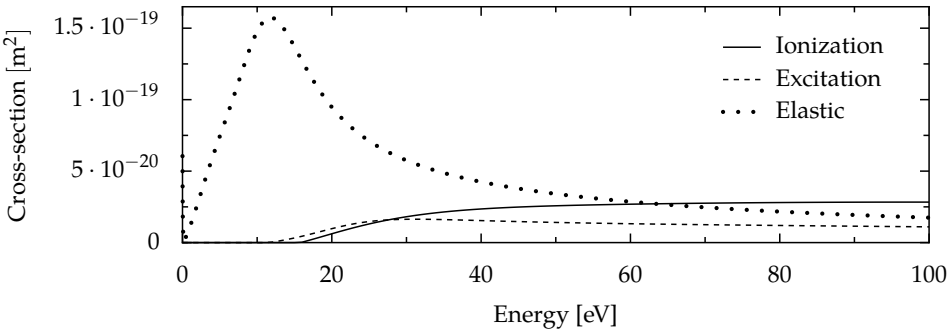


Figure 7.6. *Cross sections used in the PIC-MC model as function of the electron energy.*

7.4.5 Excitation

The second mechanism by which electrons can lose energy is the inelastic process of collisional excitation. It will be demonstrated that in contrast to elastic collisions, excitation is indeed an effective energy loss process. Moreover, it leads to a more or less stepwise decrease in energy. To understand the impact of these excitation processes on the temporal behavior of the EEDF we construct a *ladder*-model. It is based on the same assumption as the PIC-MC model in the sense that all excited atomic Ar levels are combined into a single level at 11.5 eV above the Ar ground state. This means that after being created by a photon (of 92 eV) the electrons (initially all with an energy of $92 - 15.8 = 76.2$ eV) will decay in fixed steps of 11.5 eV, as shown in Fig. 7.7.

The electrons are "cooled" in a chain-reaction until they reach the energy of 7.2 eV, where they are no longer able to excite Ar atoms. The result is that we have seven *quasi discrete* energy levels of free electrons. So the EEDF can be seen as composed by the population of these quasi discrete levels. When the density of every quasi level is identified by a number as in Fig. 7.7, the seven steps can each be described by the following equation:

$$\frac{\partial n_i}{\partial t} = P_i - D_i n_i, \tag{7.19}$$

with n_i the electron density of a certain level i, P_i the population term for that level, and D_i the depopulation frequency of level i to level $i-1$. The next level $(i-1)$ is populated by the depopulation of the adjacent upper level (i), so the population term of $i-1$ equals the depopulation term of i: $P_{i-1} = n_i D_i$.

The depopulation frequency can be written as:

$$D_i = n_a v_i \sigma_i, \tag{7.20}$$

with v_i the speed of those electrons, and σ_i the excitation cross section of Ar by those electrons (see Fig. 7.6).

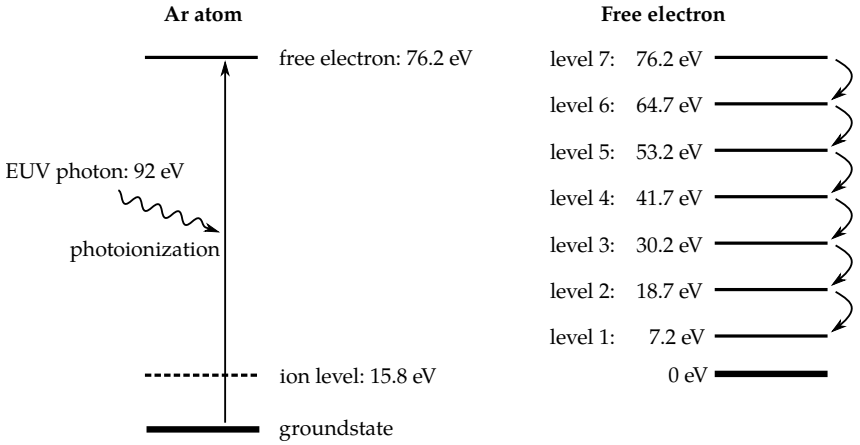


Figure 7.7. *Schematic depiction of the initial creation of free electrons by the EUV pulse (left) and stepwise cooling of the free electrons (right). The free electron loses 11.5 eV in every step of the ladder, which is every time it excites an Ar atom in the ground level.*

In the first step the electrons are populated by photoionization, so:

$$P_7 \equiv P = \frac{E_e^{\text{pulse}} n_a \sigma_{\text{ph}}}{h\nu}, \tag{7.21}$$

with σ_{ph} the photoionization cross section, and E_e^{pulse} the irradiance (power per square meter) of one EUV pulse. We assume that E_e^{pulse} is constant during an EUV pulse. P is used for this term since it is the initial source of electrons. The radiant energy per square meter of a single 100 ns EUV pulse is $H_e^{\text{pulse}} = 0.6\,\text{J}\,\text{m}^{-2}$ so:

$$E_e^{\text{pulse}} = \frac{H_e^{\text{pulse}}}{10^{-7}} = 6 \times 10^6\,\text{W}\,\text{m}^{-2}. \tag{7.22}$$

The photoionization cross section is $\sigma_{\text{ph}} = 1.37 \times 10^{-22}\,\text{m}^2$, so the population is $P = 6.7 \times 10^{22}\,\text{m}^{-3}\,\text{s}^{-1}$ at 5 Pa and $P = 2.7 \times 10^{21}\,\text{m}^{-3}\,\text{s}^{-1}$ at 0.2 Pa. For the first step (level 7), Eqn. (7.19) can now be written as:

$$\frac{\partial n_7(t)}{\partial t} = P - D_7 n_7(t) \tag{7.23}$$

so, with $n_7(0) = 0$, the solution reads:

$$n_7(t) = \frac{P}{D_7}\left(1 - e^{-D_7 t}\right). \tag{7.24}$$

level (i)	7	6	5	4	3	2
energy [eV]	76.2	64.7	53.2	41.7	30.2	18.7
D_i [10^6 s^{-1}] (5 Pa)	76	74	72	70	64	25
D_i [10^6 s^{-1}] (0.2 Pa)	3.0	2.9	2.9	2.8	2.6	0.99

Table 7.8. *Pressure dependent destruction frequencies of all excitation steps in Eqn.* (7.20). *Note that no value is included for level 1, since electrons at that level are not energetic enough to excite atoms.*

The occupations of the subsequent lower steps (levels) are ruled by the differential equations:

$$\frac{\partial n_i(t)}{\partial t} = D_{i+1} n_{i+1}(t) - D_i n_i(t), \tag{7.25}$$

for i is 6 through 2, with initial condition $n_i(0) = 0$. As an example, the solution for the next highest level ($i = 6$) is:

$$n_6(t) = \frac{P}{D_6}\left(1 + \frac{1}{D_7 - D_6}\left(D_6 e^{-D_7 t} - D_7 e^{-D_6 t}\right)\right).$$

The last level, at 7.2 eV, does not have enough energy to excite atoms so it does not have a depopulation term:

$$\frac{\partial n_1(t)}{\partial t} = D_2 n_2(t), \tag{7.26}$$

with, again, $n_1(0) = 0$.

The EUV pulse ends after 100 ns, so from that moment on there is no more initial population term ($P_7 \equiv P = 0$). The same equations need to be solved, except that now the boundary conditions are the values at $t = 10^{-7}$ s.

Table 7.8 shows the quasi levels and their depopulation frequencies D_i.

Solving the equations for all seven levels, and employing the frequencies in table 7.8, leads to the results shown in Fig. 7.8. Comparing the results obtained for 5 Pa to those for 0.2 Pa, it is clear that at high pressure the energetic levels are able to reach a quasi steady state due to the high depopulation frequency. Electrons with high energy are swiftly channeled to the lowest level. Level 2 has the lowest frequency, so the density builds up much higher than the higher levels. Because the electrons can not go below level 1, all electrons are eventually accumulated in that level. At low pressure the electrons lose their energy more slowly (less collisions), so the separate steps in the cooling of the high energy electrons are clearly visible. At low pressure it takes much longer for all higher levels to be depleted and all electrons to end up at level 1.

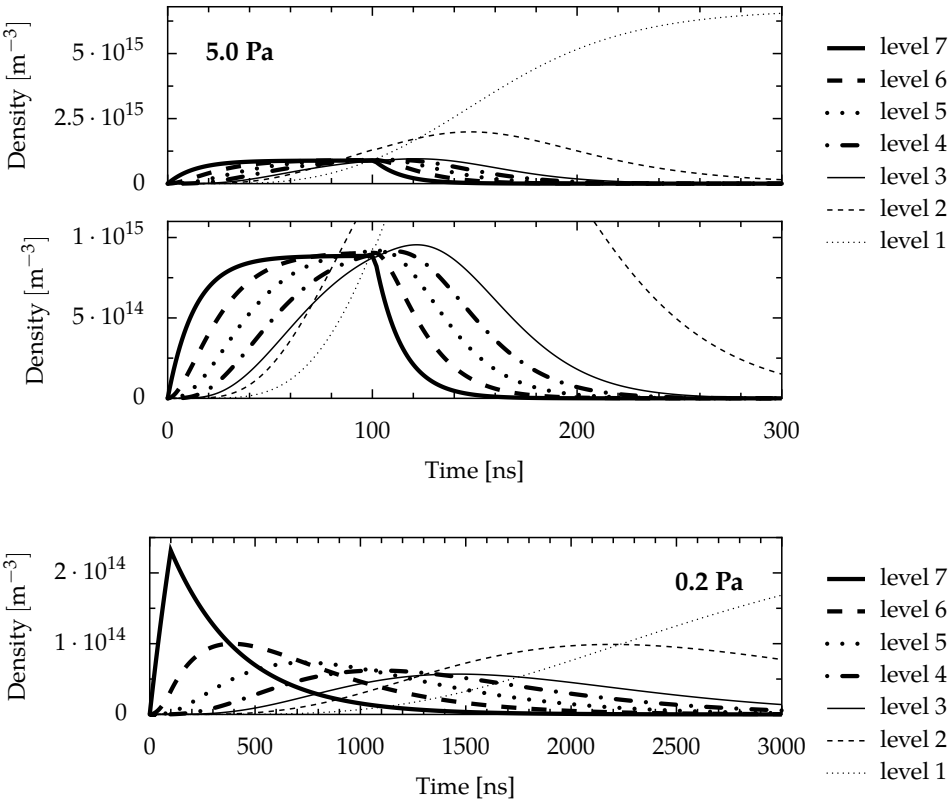


Figure 7.8. *Analytically determined evolution for two pressures of the quasi levels as a function of time. The two top graphs are the same except for the scale of the y-axis, this makes the evolution of levels 7 through 3 more clear. Note that the scale of the x-axis is 10 times as long for the low pressure case.*

In Fig. 7.9 the evolution of the same energy levels as Fig. 7.8 is shown, but here the values are obtained from the PIC-MC model in which only the excitation collision is enabled, just as in the ladder model.

When the electrons are created they all have the same energy (76.2 eV), but the EEDF from the PIC-MC model shows broadening. The EEDF at the end of the EUV pulse ($t = 100$ ns) is shown in Fig. 7.10.

The lines representing the quasi levels in Fig. 7.9 were obtained by integrating the EEDF over a range of 11.5 eV around each level. The broadening of the peaks shown in Fig. 7.10 is unexpected, since only exciting collisions are included that decrease the energy of the electron by exactly 11.5 eV. Further investigation into this issue showed that there are small fluctuations in the electric field caused by

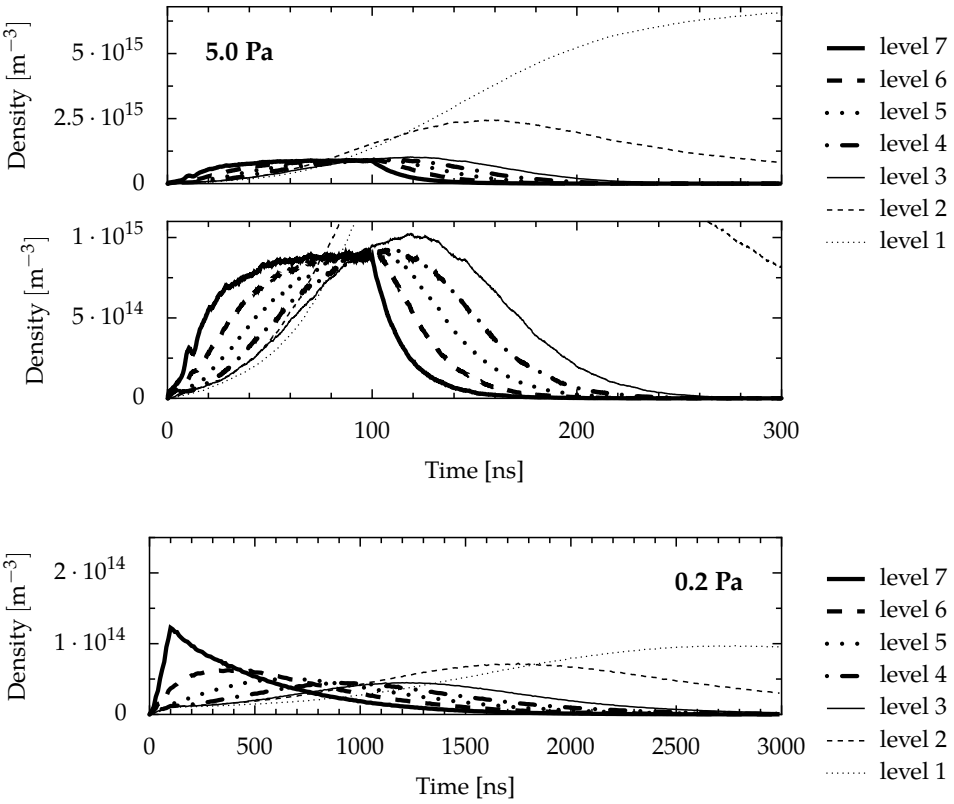


Figure 7.9. *Evolution of the EEDF at the quasi levels as a function of time, calculated by the PIC-MC model. The top graph shows the results for a background pressure of 5 Pa, the bottom graph for 0.2 Pa. Note the difference in scales as in Fig. 7.8.*

the moving charges. This leads to small fluctuations in electron energy and thus to a broadening of the peaks.

At high pressure (5 Pa) the analytical model shows good agreement with the PIC-MC model, both in the densities that are reached as in the behavior in time. At lower pressure the densities that are reached with the PIC-MC model are lower and the levels evolve faster, though the shapes show good agreement. Because the peaks are broadened the electrons lose their energy faster. This increases the effective depopulation frequencies, which results in lower level densities, and a faster evolution.

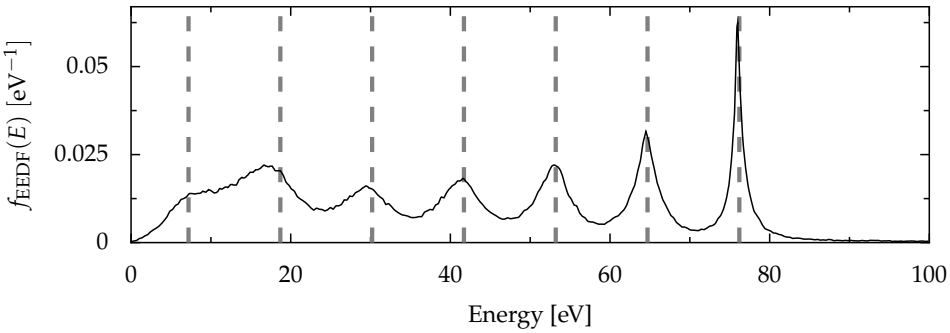


Figure 7.10. *The EEDF at $t = 100$ ns (the end of the EUV pulse) as calculated by the PIC-MC model using only excitational collisions at 5 Pa. The dashed lines represent the levels 1 through 7 that were used in the analytical model, as defined in Fig. 7.7. Note the broadening of the peaks.*

7.4.6 Ionization

Another possibility for energetic electrons to lose their energy is by ionizing the background gas. This process is similar to excitation with the important difference that there is not a well defined drop in energy. An essential feature of ionization is that a new free electron is created and that the excess energy of the incident electron (electron energy minus the ionization energy) is divided over both electrons. This division is however not equal, but subject to a certain distribution function. A semi-empirical function by Opal et al. [76] is used in the PIC-MC model, and is illustrated in Fig. 7.11. Because of this random distribution, the electrons will not lose their energy in well defined steps, but the electrons will be spread over a range of energies.

Fig. 7.11 shows that the excess energy after ionization is mostly divided unevenly. For an incident electron energy of 76.2 eV (60.4 eV remaining), in approximately 50% of the cases less than 10 eV is transferred to the new electron, so that the incident electron keeps more than 50 eV. From Fig. 7.11 it is clear that the step down in energy of the incident electron is in most cases between 15.8 and 35 eV. Most electrons will therefore cool down to under 10 eV after just two ionizing collisions.

To circumvent the fact that in the PIC-MC model the remaining kinetic energy after ionization is randomly divided over the two resulting electrons, it is assumed that the division of the remaining energy is fixed. For convenience we will assume that in all cases 9.6 eV will be transferred to the liberated electron. This will result in the scheme shown in Fig. 7.12.

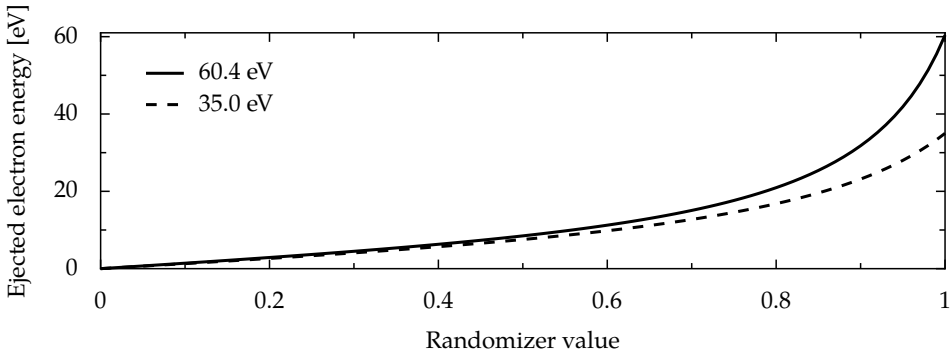


Figure 7.11. *Opal function [76] giving the distribution, according to a random number, of the energy of the ejected electron after ionization by an electron with 76.2 eV, and 50.8 eV (remaining kinetic energy 60.4 eV and 35 eV respectively), as used in the PIC-MC model. An incident electron of 76.2 eV ionizes an Ar atom and loses 15.8 eV in the process. The remaining 60.4 eV is then divided between the incident electron and the electron liberated in the ionization process. The division is determined as follows: a random number between 0 and 1 is chosen and the Opal function (in the 60.4 eV case the solid line) gives the amount of energy that is transferred to the new electron. Whatever energy is left will be the new energy of the incident electron (see also Fig. 7.12).*

The equations describing the densities of the four levels are now:

$$\frac{\partial n_4(t)}{\partial t} = P - D_4 n_4(t) \tag{7.27a}$$

$$\frac{\partial n_3(t)}{\partial t} = D_4 n_4(t) - D_3 n_3(t) \tag{7.27b}$$

$$\frac{\partial n_2(t)}{\partial t} = D_3 n_3(t) - D_2 n_2(t) \tag{7.27c}$$

$$\frac{\partial n_1(t)}{\partial t} = D_4 n_4(t) + D_3 n_3(t) + D_2 n_2(t) \tag{7.27d}$$

Note that the electrons liberated by the ionization step from level 2 will end up with no energy. There is no level with energy zero included in the model, so these electrons are not added to the density of level 1. This will not be of much influence. The production term is identical to the excitation case (7.21). For the depopulation terms D_i Eqn. (7.20) is used, only now with the ionization cross section (shown in Fig. 7.6). The values of the depopulation frequencies are listed in table 7.9. Note that these frequencies are roughly twice as high as those in table 7.8, since the cross section for ionization is roughly twice as high as the excitation cross section.

The results of this analytical model are shown in Fig. 7.13. Similar as in the previous subsection, where the excitational cooling mechanism was studied, the PIC-MC model was modified to include only the ionization process. The results

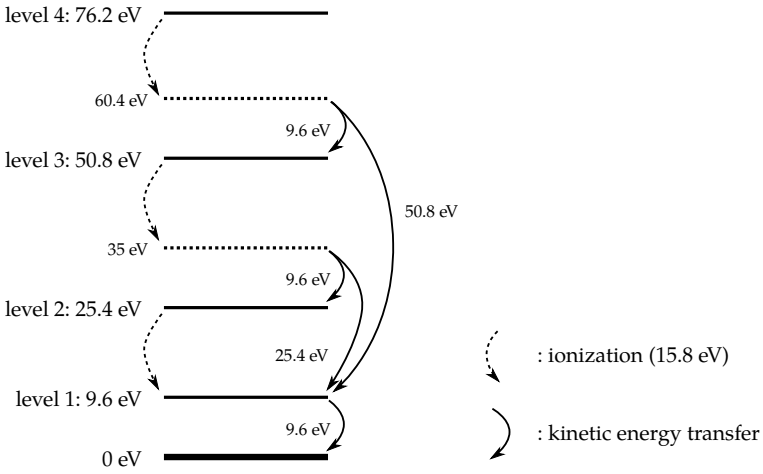


Figure 7.12. *Schematic depiction of the levels used in the analytical model for the quasi electron levels relevant for the ionization process. All electrons start at level 4. When they ionize an Ar atom they lose 15.8 eV (dashed arrow). Of the remaining 60.4 eV, 9.6 eV is transferred (solid arrow) to the liberated electron (level 1), leaving the incident electron with 50.8 eV (level 3). For electrons at level 3 and 2 the process repeats itself, populating level 1 and 2.*

level (i)	4	3	2
energy [eV]	76.2	50.8	25.4
D_i [$\times 10^6 s^{-1}$] (5 Pa)	173	131	48.3
D_i [$\times 10^6 s^{-1}$] (0.2 Pa)	6.9	5.3	1.9

Table 7.9. *Pressure dependent depopulation frequencies in Eqn.* (7.27)*. Note that no constant is included for level 1, since electrons at that level are not energetic enough to ionize atoms.*

of the PIC-MC model are shown in Fig. 7.14.

The graphs in 7.13 and 7.14 are quite similar. Only level 2 and 3 have a higher density in the analytical model compared to the PIC-MC model.

The EEDF after 50 ns (halfway through the EUV pulse) is show in Fig. 7.15. Level 4 at 76.2 eV (Fig. 7.12) is clearly recognizable, but in the PIC-MC model the analytical level at 50.8 eV (level 3) is absent. Instead we see a broad local EEDF peak around 40 eV. Because of the spreading of the excess energy over the electrons, lower levels are not recognizable. Apparently the excess energy is divided more evenly over the electrons in the PIC-MC model, so the step to level 3 is larger. Naturally this means that the density at the energy used in the analytical model is lower. This effect is repeated in the step to level 2.

When the results of excitation and ionization cooling are compared, it is clear

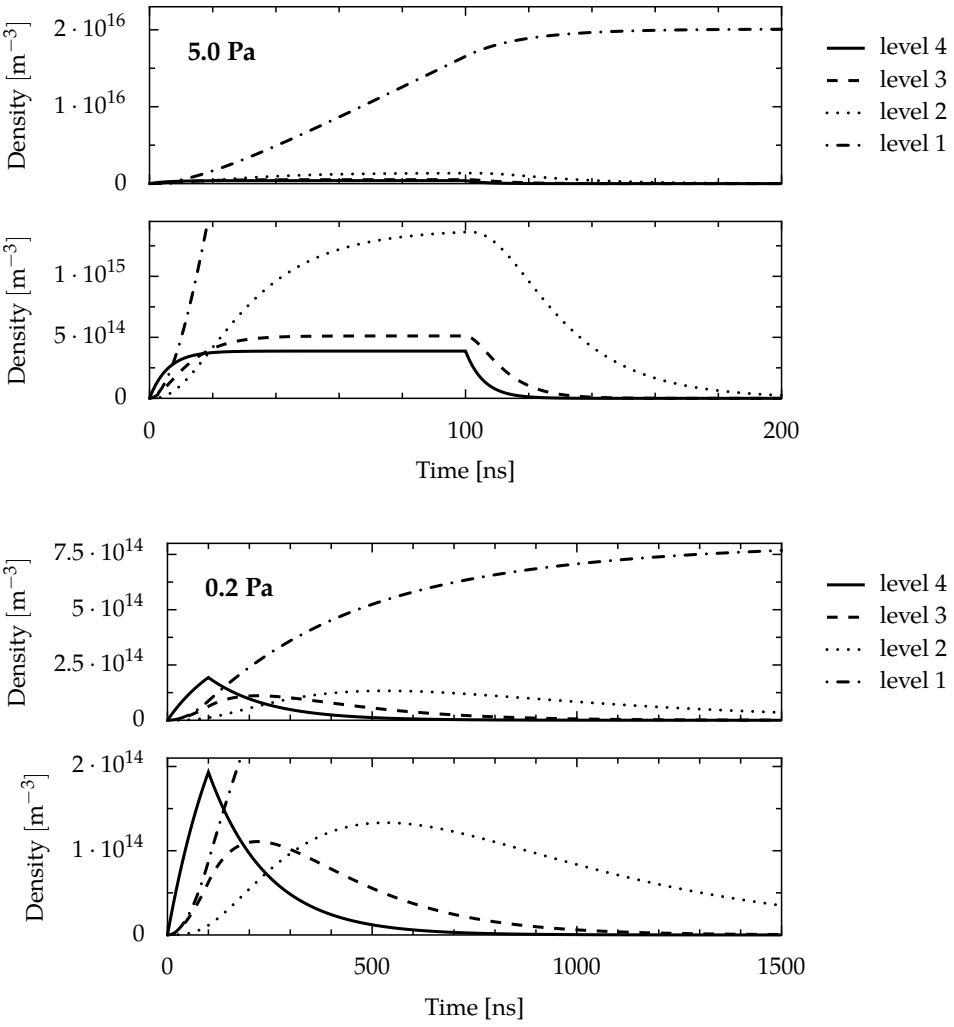


Figure 7.13. *Analytically determined evolution of the EEDF for two pressures of the quasi levels as shown in Fig. 7.12, as a function of time. The top pair and bottom pair graphs are the same except for the scale of the y-axis. Note also the difference in time scale (x-axis).*

that ionization will be the dominant cooling mechanism. It is faster because of the higher frequency (due to the higher cross section), and because it involves larger and therefore fewer steps.

Comparison of the various models shows that it is quite possible to use simple analytical models to generate usable EEDFs instead of the computationally expensive PIC-MC model.

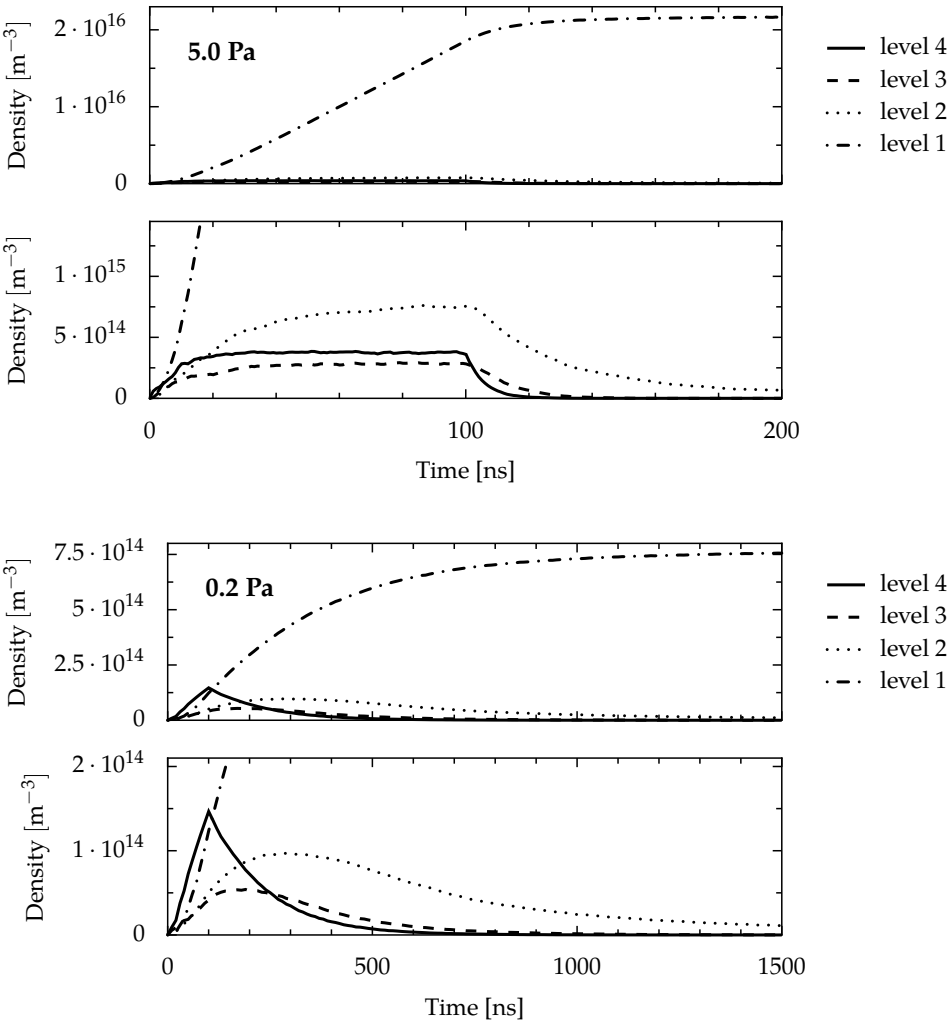


Figure 7.14. *Evolution of the EEDF of the quasi levels as a function of time, calculated with the PIC-MC model with only ionizing collisions enabled. Note the difference in scale of both axes.*

7.4.7 Complete PIC-MC model

The EEDFs computed by the PIC-MC model with all processes included, as listed in subsection 7.4.2, are shown in Fig. 7.16 for 5 Pa, and 7.17 for 0.2 Pa. The EEDFs at three different time steps are shown in Fig. 7.18.

Several striking features are found:

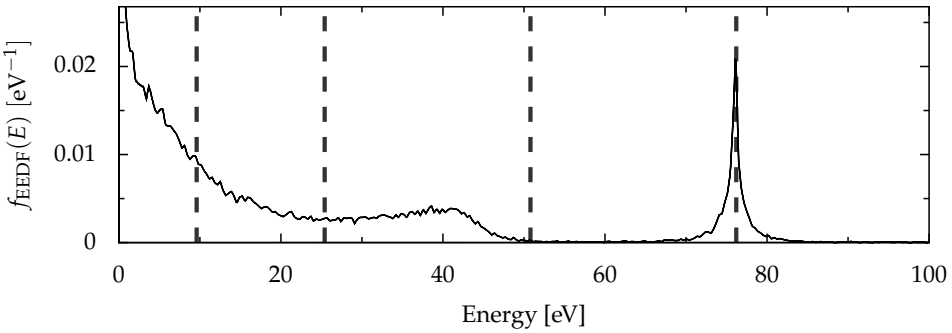


Figure 7.15. *The EEDF at $t = 50$ ns as calculated by the PIC-MC model using only ionizing collisions at 5 Pa. The dashed lines represent the locations of the levels as defined in Fig. 7.12.*

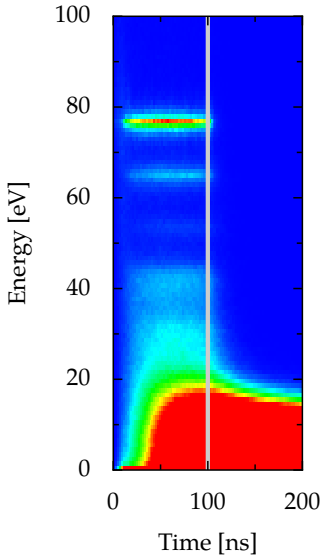


Figure 7.16. *Color map of the EEDF as computed by the complete PIC-MC model at 5 Pa. Blue represents zero density, and red maximum density of the peak at 76 eV. The gray line at 100 ns marks the end of the EUV pulse.*

- At high pressure (5 Pa), the electrons lose their energy very rapidly. Almost instantly after the EUV pulse has ended (at 100 ns) there are no high energy electrons left.
- Fast decreasing density of high energy electrons means fast increasing density of low energy electrons. Already during the EUV pulse a high density of low energy electrons is reached in the 5 Pa case.

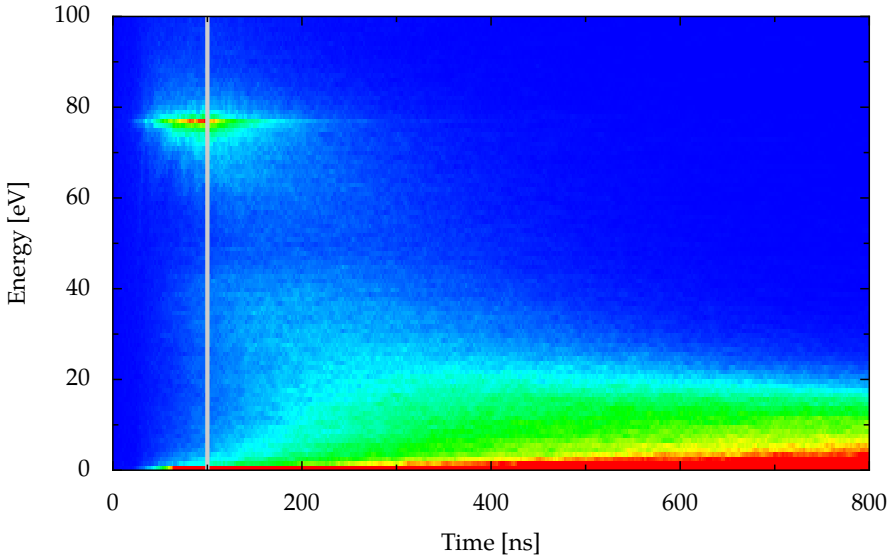


Figure 7.17. *Color map of the EEDF as computed by the complete PIC-MC model at 0.2 Pa. Blue represents zero density, and red maximum density of the peak at 76 eV. The gray line at 100 ns marks the end of the EUV pulse.*

- At high pressure, there are four peaks visible:
  - At 76 eV: the peak of the electrons initially introduced by photoionization.
  - At 65 eV and 53 eV: after exciting one Ar atom and two Ar atoms respectively (see also Fig. 7.7).
  - At 40 eV: the energy left after ionization (see also Fig. 7.15;
- At low pressure the peak representing the initial electrons is still intact at the end of the EUV pulse. About 150 ns later the peak has collapsed, but a large fraction of the electrons still has a high energy;
- The density of low energy electrons increases slowly. Even 400 ns after the EUV pulse that density is still increasing.
- At low pressure only the peak at 76 eV can easily be recognized. Broad peaks of higher densities can be seen at 65 eV and 40 eV.

The effect of both excitation and ionization processes can be seen in the EEDFs. Especially at high pressure different steps are recognizable. At high pressure the electrons lose their energy very rapidly. There will only be electrons with a high

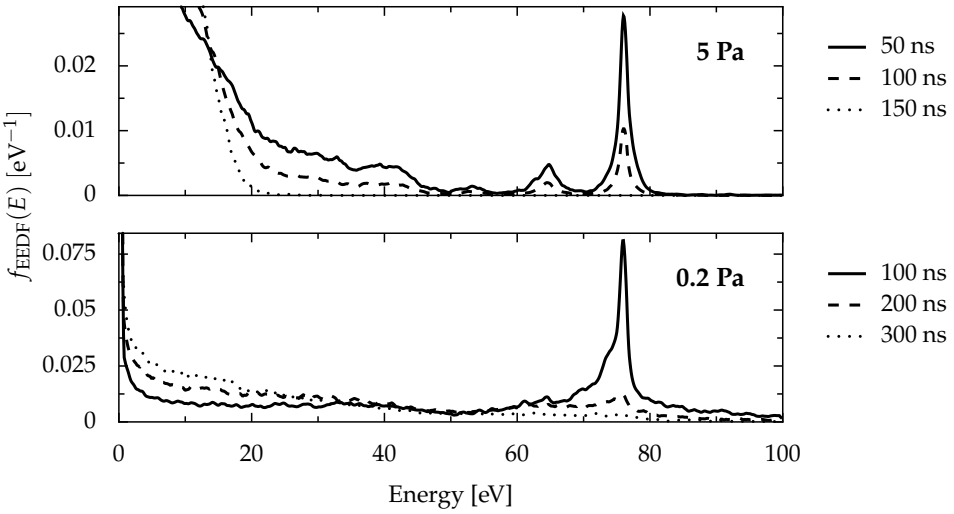


Figure 7.18. *EEDFs as calculated by the PIC-MC model at 5 Pa and 0.2 Pa at different moments in time.*

energy during the EUV pulse. At low pressure, even 200 ns after the EUV pulse has ended, there is still a considerable fraction of high energy electrons in the plasma. Low energy electrons appear very fast at high pressure, but only after the EUV pulse at low pressure.

From these EEDFs we can expect the following results for the CRM:

- At high pressure, ArI lines will rise quickly and start their decay immediately after the EUV pulse;
- At low pressure, ArI lines will rise more slowly, reaching their maximum well after the end of the EUV pulse;
- At high pressure, ArII lines will only be visible during the EUV pulse because they decay very rapidly ($A \approx 5 \times 10^7\,\mathrm{s}^{-1}$);
- At low pressure, ArII lines will only appear at the end of the EUV pulse, and decay more slowly.

7.5 CRM results

In the previous sections we have shown that the EUV induced plasma can not be described using equilibrium laws and that in order to interpret the results of OES measurements we require a CRM, which in turn requires an EEDF and electron

level	wavelength [nm] (A [10^6 s^{-1}])	
2p1	750.6 (22)	
2p2	696.7 (156)	
2p3	706.9 (263)	738.6 (118)
2p5	751.6 (25)	
2p6	763.7 (41)	
2p7	810.6 (40)	
2p9	811.8 (30)	

Table 7.10. *The eight strongest lines in the ArI spectrum, the levels from which they radiate, and between parentheses the transition probabilities of the radiative transitions in 10^6 s^{-1}.*

density from a model. In section 7.4.5 and 7.4.6 a comparison was made between an analytical (ladder model) and numerical (PIC-MC) EEDF. We will now use both forms of EEDF in the CRM. The *numerical* EEDFs that are used (as look-up tables) are those discussed in section 7.4.7. For the *analytical* EEDFs we will use the EEDFs resulting from the ladder model in which only ionization was enabled as a basis. To account for the additional energy losses of the electrons due to excitation, we use the sum of the ionization and excitation curves in Fig. 7.6. The EEDF is formed by four rectangular functions of 2 eV width, each centered around one of the energy levels as shown in Fig. 7.12. The height of the four rectangular functions is determined by Eqn. (7.27), and thus time dependent. For both EEDFs the EUV pulse is considered to be a rectangular function with a width of 100 ns.

The results for 5 Pa are shown in Fig. 7.19 for atomic spectral lines, and Fig. 7.20 for ionic spectral lines. The results for 0.2 Pa are shown in Figs. 7.21 and 7.22. Only the eight strongest atomic lines in the measured spectra (see section 7.6) are plotted. They are listed in table 7.10. The figures showing plots for the ionic lines include all eight ionic 4p $\rightarrow$ 4s lines that are included in the CRM, see table 7.7.

At 0.2 Pa both the atomic and ionic lines produced by the analytical EEDF are in reasonable agreement with those obtained by the numerical EEDF. At 5 Pa there are some differences. The shapes are quite similar, though the ionic lines for the analytical EEDF reach an equilibrium state after approximately 50 ns, whereas for the numerical EEDF the lines keep increasing until the end of the EUV pulse. Both the atomic and ionic lines show lower intensity for the analytic EEDF; for the atomic lines the difference is quite substantial. This can be attributed to the fact that in the analytical model the electrons are cooled by overestimated ionization; i.e., it includes a contribution representing excitation. By excitation the electrons would cool much slower, so due to the overestimation there are less electrons left to excite atoms. At low pressure this effect is compensated by the fact that the

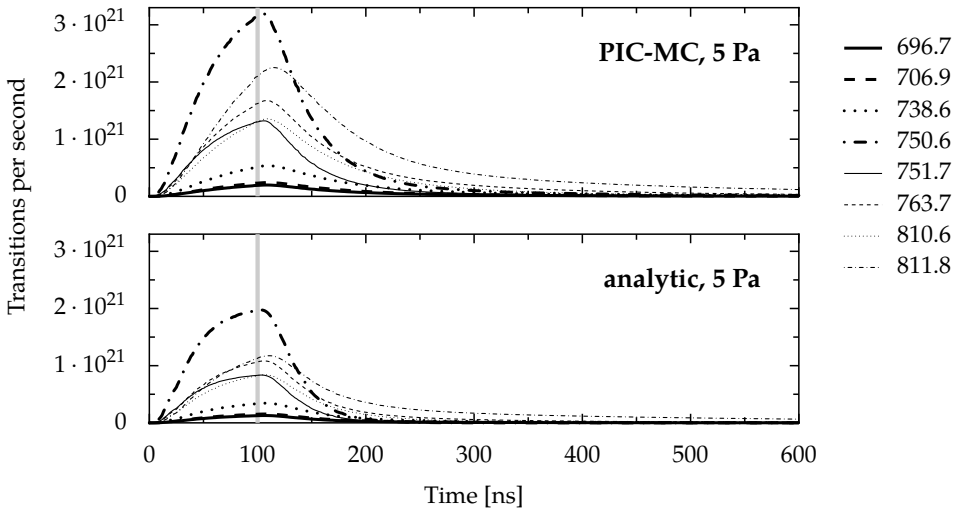


Figure 7.19. *Atomic spectral lines as calculated by the CRM using a numerical (top) and analytical (bottom) EEDF. The pressure is 5 Pa, the end of the EUV pulse is at 100 ns (vertical, gray line).*

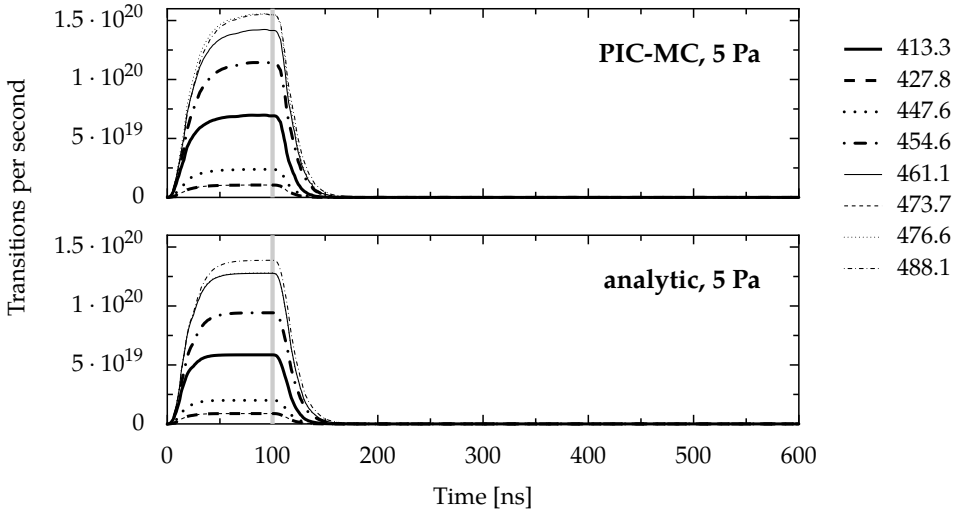


Figure 7.20. *Ionic spectral lines as calculated by the CRM using a numerical (top) and analytical (bottom) EEDF. The pressure is 5 Pa, the end of the EUV pulse is at 100 ns (vertical, gray line).*

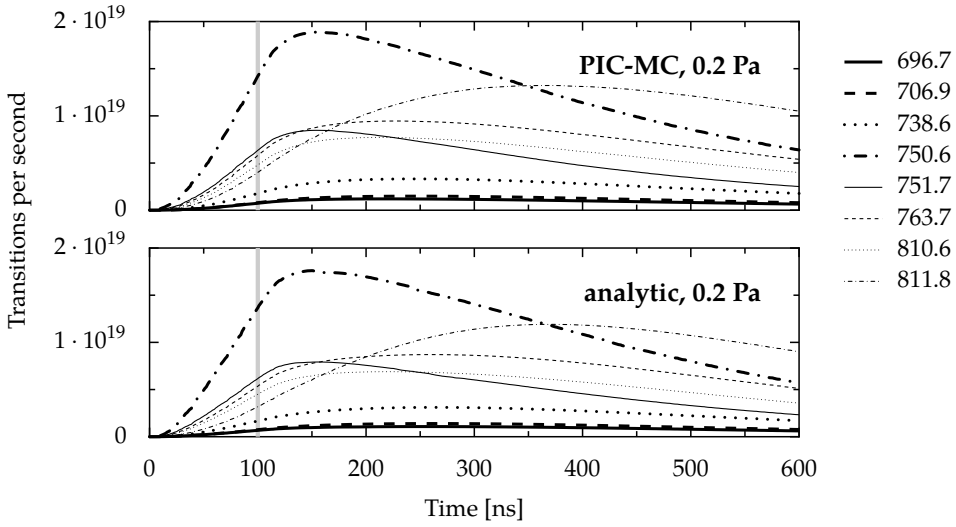


Figure 7.21. *Atomic spectral lines as calculated by the CRM using a numerical (top) and analytical (bottom) EEDF. The pressure is 0.2 Pa, the end of the EUV pulse is at 100 ns (vertical, gray line).*

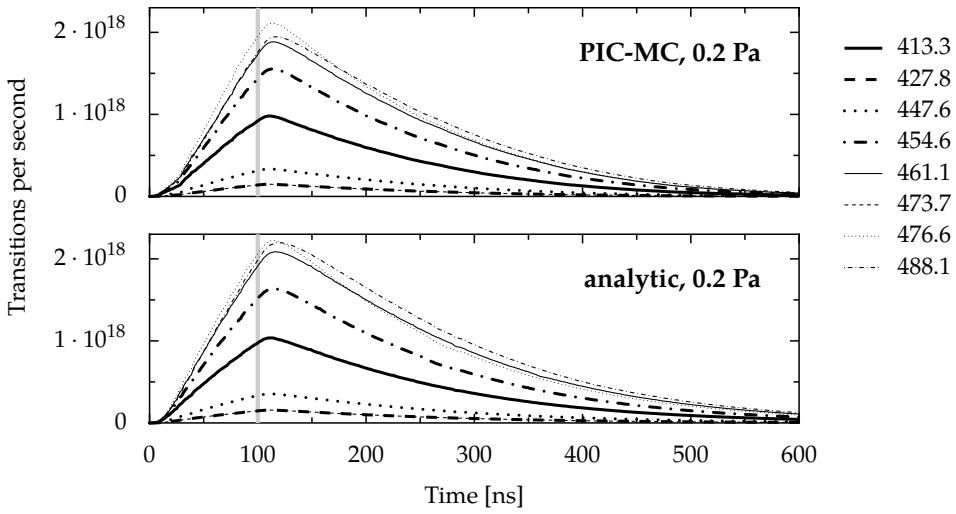


Figure 7.22. *Ionic spectral lines as calculated by the CRM using a numerical (top) and analytical (bottom) EEDF. The pressure is 0.2 Pa, the end of the EUV pulse is at 100 ns (vertical, gray line).*

energy of the electrons is spread over a larger energy range, due to the slower cooling. In the analytical EEDF the electrons at the lowest rung of the cooling ladder can not excite an atom, but in the numerical EEDF, the lowest rung is spread over a larger energy range, so part of these low energy electrons can still excite atoms.

The plots of the spectral lines show that with a rudimentary analytic ladder model the EEDF can be modeled rather accurately. Both forms of the EEDF show that the spectral lines give insight into the evolution in time of the EEDF. At 5 Pa the atomic lines reach their maximum at the end of the EUV pulse, and decay rapidly afterwards, in approximately 200 ns. At 0.2 Pa the atomic lines rise more slowly, only reaching their maximum after the EUV pulse. Most notably, the 811 nm line reaches its maximum 300 ns after the end of the EUV pulse. The reason for this is the cascade contribution.

In Fig. 7.23 the population mechanisms for two 4p levels are shown: *2p1*, which radiates at 750.6 nm, and *2p9*, which radiates at 811.8 nm. Both levels are populated by collisional excitation from the ground state and radiative decay from higher lying levels; the cascade contribution. For the *2p1* level the cascade contribution is negligible at both pressures. However, the population of the *2p9* level does have a significant cascade contribution, at 0.2 Pa this is even comparable to the collisional contribution. This means that this level is to a large extent indirectly populated via higher lying levels, and thus, that the level density will reach its maximum with a certain delay. The direct result is that the radiative decay will have a similar delay in its maximum.

The delay in the maximum of level *2p9* is also caused by a second effect. In Fig. 7.24 the cross sections of level *2p1* and *2p9* are plotted. While the cross sections have a comparable maximum value near threshold, the cross section of *2p9* quickly drops after the maximum. This means that this cross section acts as a "probe" for the EEDF at the energy at which the cross section has a maximum, which is at approximately 25 eV. When we examine Fig. 7.17 we observe that the EEDF occupation at 25 eV rises until around 300 ns, after which it slowly decreases. This behavior is exactly what we see in Fig. 7.23d.

For level *2p1* the behavior is different because due to the broad cross section the collisional population is also determined by a broad energy range. In the broad energy range (above the threshold of 13 eV) the EEDF rises until the end of the EUV pulse, and then slowly and steadily drops. This can also be seen in the population rate, see Fig. 7.23c.

Overall, the results of the CRM show that the spectral lines directly provide information about the EEDF. The ionic lines reflect the time evolution of the EEDF between 40 eV and 80 eV, while the atomic lines reflect the EEDF between 20 eV and 40 eV. Furthermore, the ion lines closely follow the shape of the EUV pulse, with a delay of up to 50 ns, while the atomic spectral line at 750.6 nm is a good "probe" for the EEDF since it is not influenced by cascade contribution and has a

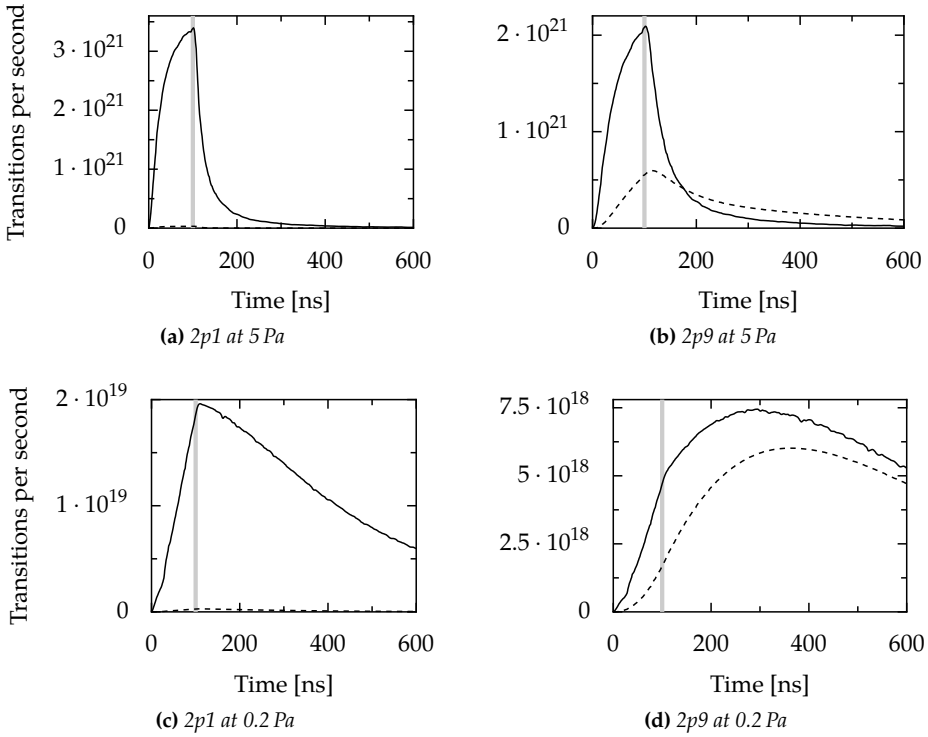


Figure 7.23. *Collisional and cascade population rates: (a) for level 2p1 at 5 Pa, radiating at 750.6 nm; (b) for level 2p9 at 5 Pa, radiating at 811.8 nm; (c) for level 2p1 at 0.2 Pa; (d) for level 2p9 at 0.2 Pa. The solid lines show the collisional rate (from the ground level), the dashed line the cascade contribution from higher levels. The vertical, gray line at 100 ns shows the end of the EUV pulse.*

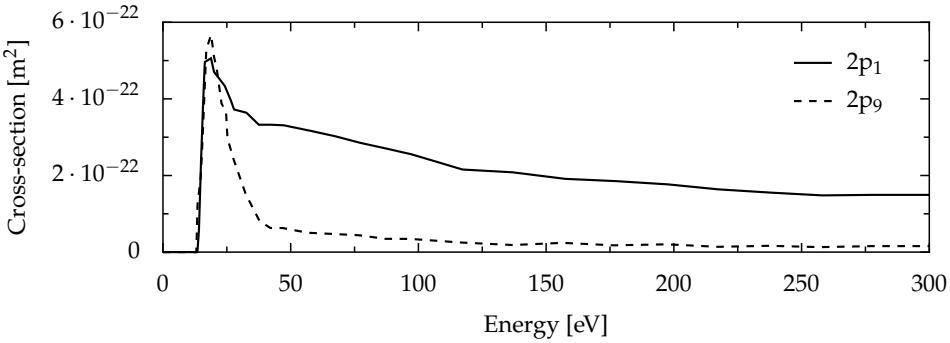


Figure 7.24. *The cross sections for electron excitation from the ground state for Ar levels 2p1 and 2p9.*

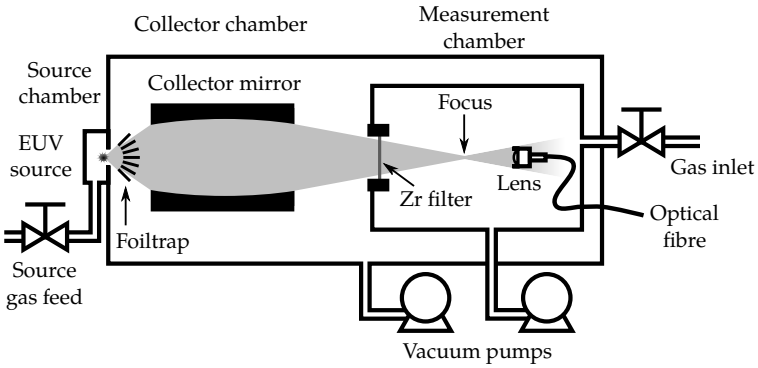


Figure 7.25. *Schematic drawing of the experimental setup.*

large cross section. The difference between the time dependent spectra at the two different pressures is clearly shown by the spectral lines.

7.6 Experimental results and discussion

Apart from the model, time resolved spectral lines have been determined experimentally. We will first give a description of the experimental setup, followed by a discussion of the results in comparison to the results of the model.

7.6.1 Experimental setup

The experiments were performed in the EUV laboratory at ASML, where a Hollow Cathode Triggered (HCT) Xe Discharge Produced Plasma source (DPP) was used (see Kieft [26]). A schematic depiction of the setup is shown in Fig. 7.25.

The setup consists of three parts: the source chamber, the collector chamber, and the measurement chamber. In the source chamber a small ($\sim$ 1 mm) pulsed (up to $\sim$ 1 kHz), plasma is created, which needs a mixture of He, Ar, and Xe with a total filling pressure of 10 Pa to 30 Pa. This plasma produces radiation in a wide wavelength range, among which the desired EUV radiation.

The source chamber is directly connected to the collector chamber. It contains an ellipsoidal mirror (the collector) with the function to collect the light originating from the source. The inside of the collector is coated with a gold layer to act as a grazing incidence mirror. To prevent debris originating from the plasma from entering the collector chamber and possibly damaging the optics, a foiltrap [77] is placed between the source and collector chamber. The collector mirror images the collected radiation at a focus point in the measurement chamber, the so-called intermediate focus.

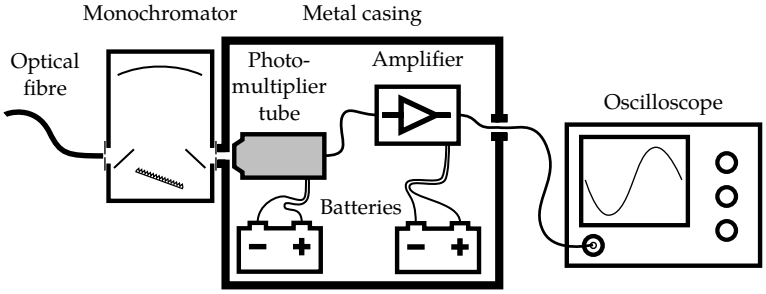


Figure 7.26. *Schematic drawing of the measurement setup. Batteries were used as power supply in order to suppress EM-interference from the EUV source.*

The collector chamber is separated from the measurement chamber by a filter to prevent all but EUV radiation from entering the measurement chamber. The filter consists of a 150 nm thick Zr foil reinforced by a wire mesh so that it can withstand larger pressure differences between the chambers. The filter therefore enables control of the gas mixture and pressure in the measurement chamber independent of the conditions in the source and collector chamber. Normally, this setup is not equipped with a separate measurement chamber. It was specifically added to the setup to accommodate the spectroscopical experiment, since it requires control of the composition and pressure of the background gas. A separate measurement chamber is required for this task, because altering the conditions in the collector chamber can disrupt the EUV source. While the pressure in the collector chamber was kept at a constant value of 0.1 Pa to ensure proper operation of the EUV source, the pressure in the measurement chamber could be varied from roughly 0.1 Pa to 10 Pa.

To record the evolution of the line intensities as a function of time, the setup schematically depicted in Fig. 7.26 was used. The fluorescent light from the EUV created plasma picked up by the collimating lens was led into a monochromator through an optical fiber. The monochromator (an Oriel 7240 with off-axis Ebert configuration) used a 1200 l/mm grating with a blaze wavelength of 500 nm. The width of the entrance and exit slit was 280 µm, which equals an approximate bandpass of 2 nm. With this configuration the usable wavelength region of the monochromator is 300 nm to 1000 nm. Narrower slits were available but were not used to maximize the signal yield. Because of the low resolution of the setup, some lines were measured simultaneously. For instance, of the ArI system the lines at 750.6 nm and 751.6 nm are measured simultaneously, as are the lines at 810.6 nm and 811.8 nm. The measurements of all ionic lines will have some added radiation from neighboring lines.

The exit slit of the monochromator is connected to a photomultiplier tube (PMT). The PMT (Hamamatsu 6780-04) has a working range of 185 nm to 850 nm

and a rise time of 0.78 ns. The signal of the PMT is then amplified by a 1 GHz preamplifier (Ortec 9306) before the signal is recorded by a digital oscilloscope (LeCroy WaveSurfer 454).

When the EUV source is running it also generates a large amount of EM-interference. Initially, the interference picked up by the preamplifier was several orders of magnitude larger than the signal from the PMT. It was therefore necessary to place all the electronics in a metal enclosure. The power supplies for the PMT and the preamplifier were realized using batteries, which were also placed inside the enclosure. Although using this setup the interference was sufficiently reduced to receive a clear signal on the scope, some electronic filtering still had to be applied.

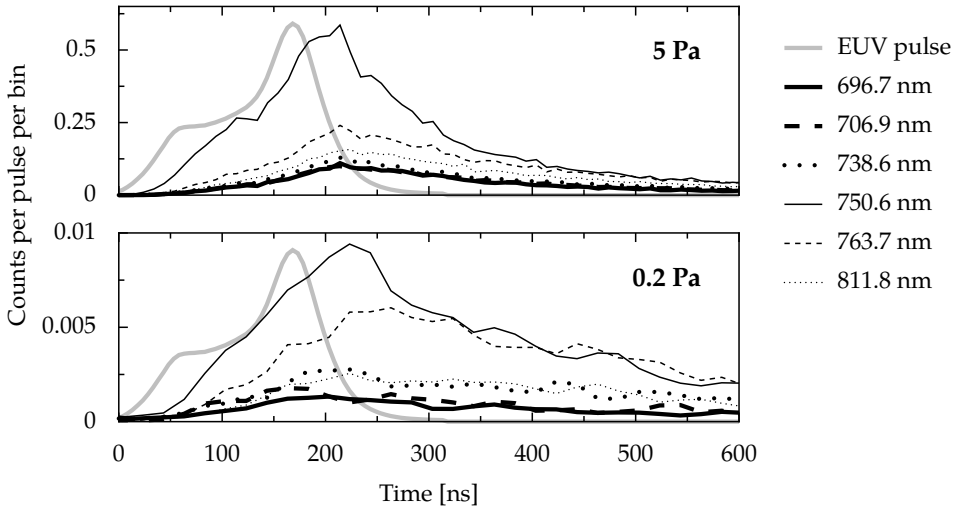
The radiation of the plasma is very weak. A measurement with a spectrometer (OceanOptics HR2000+) resulted in approximately 8000 counts for the 750.6 nm line, when the spectrum was measured for 60 s and the source was running at 1 kHz, with a background pressure of 5 Pa. According to the manufacturer, the sensitivity is 41 photons per count, meaning that per pulse only 5 photons are registered. The measurements presented in this section are obtained by combining at least 10 000 measurements per spectral line by *binning*. Each separate measurement (a single triggered measurement of the oscilloscope) is stored on a computer. For every measurement the total length is divided into a number of intervals, called *bins*. The combination is realized by counting the number of pulses in every bin for all measurements.

7.6.2 Results

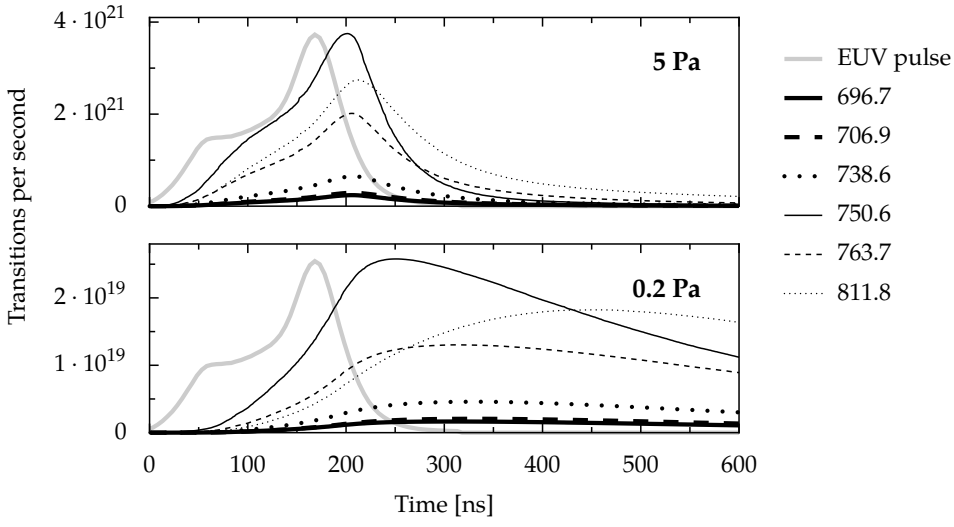
The results of the measurements are shown in Fig. 7.27 for the atomic lines and in Fig. 7.28 for the ionic lines. For comparison, the results of the CRM are also plotted in these figures. The EEDF required for the CRM was computed by the PIC-MC model for an EUV pulse with the same shape as in the experiments, and with an increased energy of 50% to account for the first small pulse.

Comparing the experimental lines between pressures, it is clear that for 0.2 Pa both the atomic and ionic lines decay slower. The maxima of the atomic lines lag behind the maximum of the EUV pulse, more so at 0.2 Pa than at 5 Pa. The ratios between the atomic lines are comparable, with the exception of the 763.7 nm line.

When we compare the experimental results to the CRM results we see some similarities and some clear differences. For the atomic lines the slower decay for 0.2 Pa is present in both results, as is the delayed maximum for 0.2 Pa. Furthermore, the relative intensity of the lines is comparable, except for the 811.8 nm line. While this line is the second strongest line in the CRM results, it is one of least intense lines in the measurements. Furthermore, the prominent delayed maximum of the 811.8 nm line at 0.2 Pa displayed in the CRM results is not seen in the experimental results.

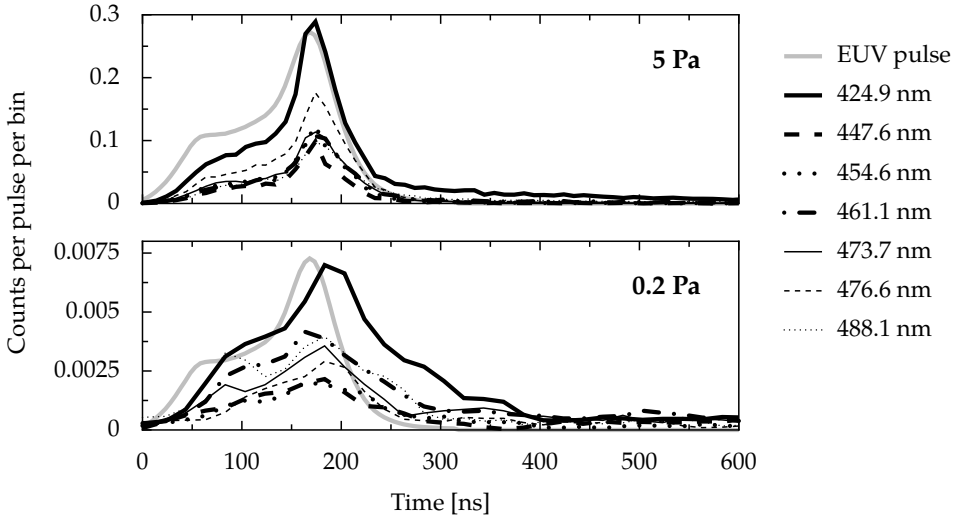


(a) *Experimental results.*

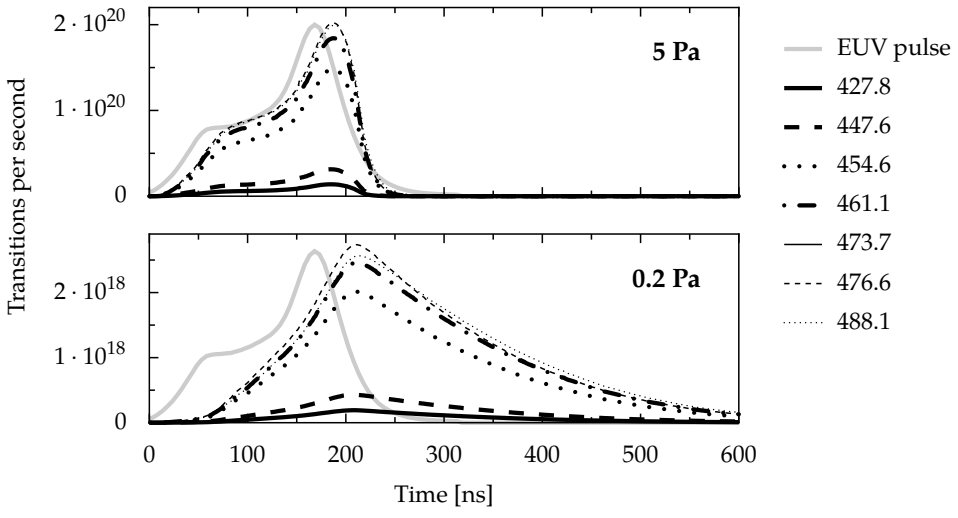


(b) *CRM results*

Figure 7.27. *The most intense ArI lines as function of time resulting from: (a) the measurements, (b) the CRM. For both, the top graph shows the results for 5 Pa and the bottom for 0.2 Pa. The gray line shows the outline of the EUV-pulse. For the experimental results the bin size is 10 ns for 5 Pa, and 20 ns for 0.2 Pa. The 0.2 Pa lines of the experimental results are smoothed by weighted average of three points.*



(a) *Experimental results.*



(b) *CRM results.*

Figure 7.28. *The most intense ArII lines as function of time resulting from: (a) measurements, (b) the CRM. For both, the top graph shows the results for 5 Pa and the bottom for 0.2 Pa. The gray line shows the outline of the EUV-pulse. For the experimental results the bin size is 10 ns for 5 Pa, and 20 ns for 0.2 Pa. The 0.2 Pa lines of the experimental results are smoothed by weighted average of three points.*

Of the ionic lines, the shapes between model and experiment are in agreement, though the decay at 0.2 Pa is significantly slower in the CRM. The relative intensity between lines is completely different; while the 424.9 nm line is dominant in the measurements, it is the least intense line in the CRM.

The difference in relative intensity of the ionic lines can only be explained by incorrect cross sections and/or transition probabilities. The faster decay of the lines in the experimental results can be explained by geometrical effects. The PIC-MC model is one-dimensional, so it is assumed that the plasma does not diffuse. If the plasma expands or diffuses after it is created this would explain the faster decay seen in the measurements. The same effect explains some of the discrepancies seen in the atomic lines. While it gives a plausible reason for the faster delay seen at 0.2 Pa, it can also account in part for the fact that no delayed maximum of the 811.8 nm line is visible; the delayed population along the cascade would be much lower. Apart from this, the lower sensitivity of the detector for 811.8 nm compared to lower wavelengths can explain the lower intensity of the measured line.

Overall we can state that although there are some clear differences between the experimental and CRM results, the experiments support the findings of the CRM that an increase in pressures of the background gas significantly decreases the density of high energy electrons.

7.7 Conclusion

The sputter rate of a surface by Ar ions created by the interaction of EUV radiation with a low pressure Ar gas is mainly determined by the energy of the electrons (the EEDF). The energy, and thus the sputter rate, can be lowered by increasing the background pressure. To verify the effect of increased pressure on the EEDF is not straightforward. Time resolved optical emission spectroscopy can be employed to investigate the EEDF. The EUV induced plasma is not governed by any form of equilibrium, therefore a CRM is required to interpret results. A simple CRM, modeling a plasma in the Corona Balance regime, has been constructed. The EEDF required by this CRM can be supplied by a PIC-MC model, or from a simple analytical ladder model. Compared to the complex and computationally expensive PIC-MC model, the analytical model gives satisfactory results for the modeling of the time dependent behavior of emission lines.

The CRM shows that spectral lines give a good representation of the EEDF; ionic lines for high energy (40 eV to 80 eV), atomic lines for lower energy (20 eV to 40 eV). At high pressure (5 Pa) the lines decay significantly faster than at low pressure (0.2 Pa). Though at low pressure the lines decay faster in the experiments, the overall evolution in time of the CRM results is in good agreement with the experimental results, confirming the effect of increased pressure on the EEDF. Apart

from the faster decay in the measured lines, some significant differences are seen in the relative intensities of the lines.

Discrepancies can partly be attributed to the simplicity of the assumptions used for creating the EEDF, which do not take into account spatial decay of the plasma. Inaccurate cross sections and detector sensitivity can account for the differences in relative intensities of the lines.

Chapter 8

A CRM of time dependent LIF experiments

8.1 Introduction

In chapters 3, 4, and 5 a description of a general CRM code and its implementation using the plasimo framework was described. In chapter 7 the resulting CRM module was applied to construct a time dependent CRM of EUV induced Ar plasmas. In this chapter we will employ the CRM module to make a model of Laser Induced Fluorescence (LIF) experiments in Ar plasmas.

The CRM must give an accurate description of a real laser–plasma experiment, in this case a Surfatron Induced Plasma (SIP) in Ar subjected to a laser pulse, tuned to Ar transitions. In section 8.2 the CRM of a complete Ar system will be constructed from various sources, in the form of an input file for the plasimo CRM module. This Ar CRM will subsequently be used to model time resolved LIF experiments.

In LIF experiments two excited levels are brought into short-lived equilibrium due to absorption of, and stimulated emission by a laser pulse. The result is that the densities of the states in the upper and lower level will attain the same value. However, during and after the pulse the system as a whole is in imbalance, causing the systems to strive to a new equilibrium. After the pulse the system will return to the steady state it was in before the pulse. In section 8.3 the time dependent CRM will be employed to study this process.

The laser pulse causes a sudden change in density of the upper and lower level. Since all levels are directly or indirectly connected to each other, this disturbance

will spread through the system. This process can be modeled with the CRM and some of the results will be compared to experiments (section 8.3.3).

The goal of this chapter is to provide the tools for a profound analysis, and show some of the results that can be obtained.

8.2 The Ar CRM

The CRM of Ar consists of several parts:

- the atom level scheme;
- radiative transitions;
- electron excitation cross sections;
- heavy particle excitation rates.

We will describe each of these elements in some detail.

8.2.1 Levels

In Fig. 8.1 a schematic depiction is shown of the Ar system as included in the CRM. Table 8.1 gives an overview of the 78 included levels*. A full list of the levels, their energies, and statistical weights can be found in the appendix in table C.1. The energies of the levels have been taken from Minnhagen et al. [78]. The energies of the level blocks are determined by using the statistical weight g to get the weighted average:

$$E_{\text{block}} = \frac{\sum_l g_l E_l}{\sum_l g_l}, \tag{8.1}$$

with E_l and g_l the energy and statistical weight of level l included in the block.

8.2.2 Radiative transitions

The online NIST database [66] was used as the source for Einstein coefficients for spontaneous emission. Since most of the levels in the CRM are in level blocks, while the NIST database provides the transitions between individual levels, some processing is required to convert the level-to-level transition probabilities in those for the traffic between level blocks. When a transition (from level u to level l) *originates* from a level block (b_u) the weighted average by weight of the level degeneracies (as in Eqn. 8.1) of the transition probability is used:

$$A(b_u, l) = \frac{g_u A(u, l)}{\sum_i g_i}, \tag{8.2}$$

* We will predominantly use Racah notation for levels and blocks. In some occurrences Paschen is also given. See appendix B for conversion tables, or table 7.1 for an overview of the lower levels.

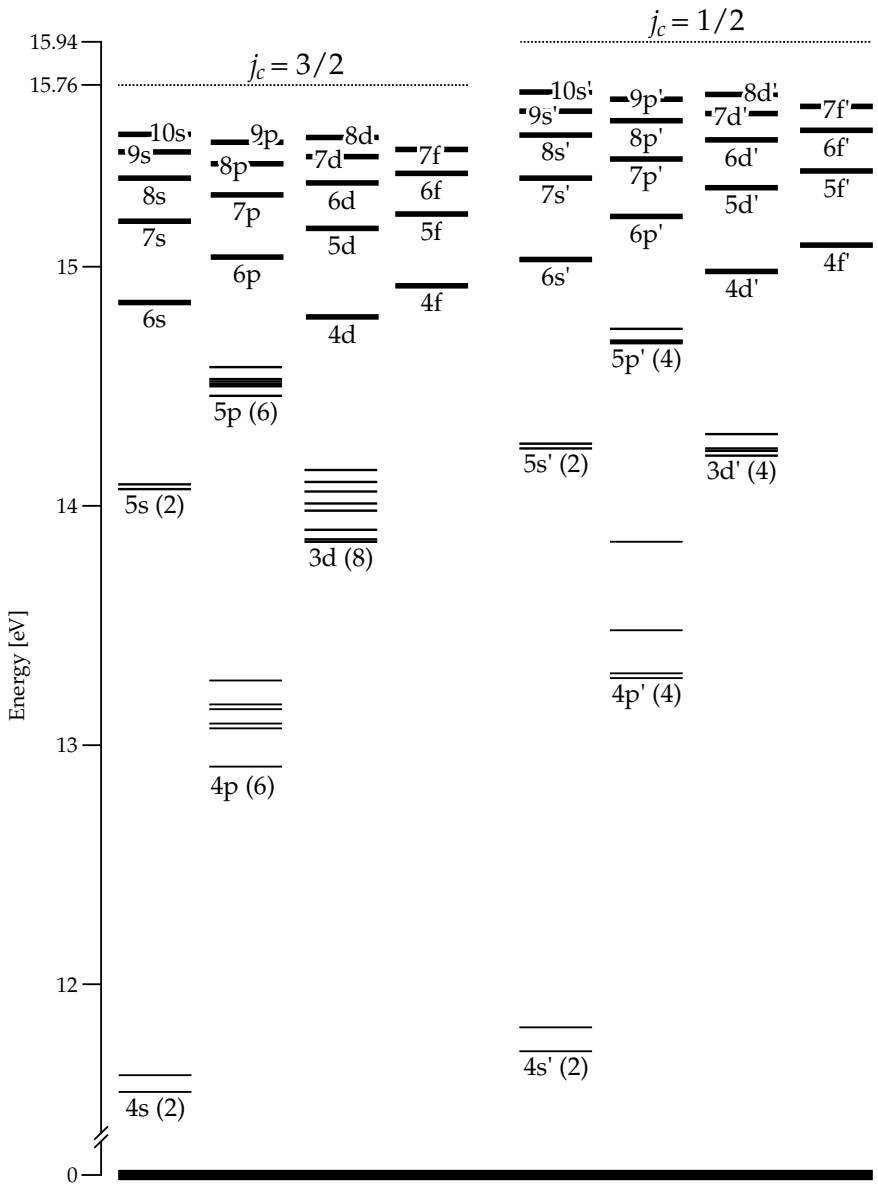


Figure 8.1. *Graphical overview of the Ar system as used in the CRM. The system is split into two parts according to the configuration of the core: $j_c = 3/2$ (un-primed), $j_c = 1/2$ (primed). The following blocks are split into individual levels (denoted by thin lines): 4s, 5s, 4p, 5p, and 3d (all primed as well as un-primed), see also table 8.1. All other levels are combined in blocks, with a block consisting of all levels with the same principal and orbital quantum number and core configuration. Note that the ion level energy in the un-primed configuration is 15.76 eV and in the primed configuration 15.94 eV.*

Individual levels (number of levels)	Grouped levels Primed and un-primed
4s (2), 4s′ (2), 5s (2), 5s′ (2)	6s, 7s, 8s, 9s, 10s
4p (6), 4p′ (4), 5p (6), 5p′ (4)	6p, 7p, 8p, 9p
3d (8), 3d′ (4)	4d, 5d, 6d, 7d, 8d
	4f, 5f, 6f, 7f

Table 8.1. *Overview of the levels included in the model (apart from the atom and ion ground state), denoted by their principal and orbital quantum number. The prime symbol (′) denotes the core configuration: primed ($j_c = 1/2$) and un-primed ($j_c = 3/2$). The left column lists the levels that are included individually, the number between parentheses is the number of levels with the same principal and orbital quantum number. The right column lists blocks of levels. Each block in the list is included twice in the model, one for each core configuration.*

with g_u the degeneracy of the upper level u, and the sum running over all levels i in the originating block b_u. The weighted averages of all transitions originating from the block ending at level l can simply be added:

$$A(b_u, l) = \frac{\sum_j g_j A(u_j, l)}{\sum_i g_i}, \tag{8.3}$$

where the sum in the numerator runs over all levels u_j in block b_u with a radiative transition to level l.

If the *destination* level is also part of a block we can sum the separate $A(b_u, l)$ of Eqn. (8.3):

$$A(b_u, b_l) = \sum_k \frac{\sum_j g_j A(u_j, l_k)}{\sum_i g_i}, \tag{8.4}$$

where the first sum runs over all levels l_k in the lower group b_l.

Table C.2 in the appendix lists all radiative transitions, their wavelengths and transition probabilities.

8.2.3 Cross sections

Cross sections for electron excitation have been taken from various literature sources. For excitation from the ground state a collection by Yanguas-Gil et al. [67] was used. This is the same set of cross sections that was used in chapter 7 for the CRM of an EUV driven plasma. This collection contains look-up tables for experimentally determined cross sections for the ground state excitation of 4s, 4p, 3d, 5s, and 5p levels. Some of those cross sections are for small blocks of levels (for instance some of the 3d and 5s levels). Since in this CRM the 3d and 5s levels

are all modeled individually, the cross sections are split according to the weight of the upper level, see Eqn. 7.14.

For excitation from the ground state to higher levels, semi-empirical cross sections by Vlček et al. [38] were used in the form of fit parameters for Drawin cross sections (Eqns. (4.23), (4.24), and (4.25)). Some of those cross sections were also split by ratio of statistical weight using Eqn. (7.14). The Drawin fit parameters by Vlček et al. provide cross sections to levels 4d, 5d, 6d (only un-primed), 6s, 7s, and 8s (only un-primed).

Excitation cross sections from 4s levels were taken from Zatsarinny et al. [79] in the form of look-up tables, obtained from R-matrix calculations. They provide cross sections from the four separate 4s levels towards the other 4s levels and all separate 4p, 3d, and 5s levels.

The cross sections for excitation were completed using calculated fit parameters for Drawin cross section by Kimura et al. [37]. They list fit parameters between all blocks listed in table 8.1 (Kimura also combines the levels that are treated individually in this CRM in blocks), with separate parameters for the primed and un-primed subsystem. Since the parameters by Kimura et al. are for blocks, they have been unraveled using Eqn. (7.14). For transitions between levels in the primed and un-primed subsystem the average of the primed and un-primed fit parameters was used. Depending on the orbital quantum number of the upper and lower block, a Drawin cross section for optically allowed transition (Eqn. (4.23) for $\Delta l = \pm 1$) or parity forbidden transition (Eqn. (4.24) for $\Delta l \neq \pm 1$) was used. Cross sections for transitions between levels within the same block (principal and orbital quantum number) were not included (except for 4s).

In addition to excitation cross sections, ionization cross sections were included. For direct ionization from the ground state the cross section from the collection by Yanguas-Gil et al. was used, originating from experiments by Rapp et al. [80]. The ionization cross section from all other levels was taken from Vlček in the form of Drawin fit parameters.

In total, the CRM contains cross sections for 2587 endothermic electron induced transitions. For all collisional electron excitation transitions detailed balancing is enabled (section 4.3.2), as are radiative recombination and two electron recombination (section 4.3.3).

8.2.4 Heavy particle induced processes

In addition to excitation by electrons, excitation by ground state atoms is included:

$$Ar_p + Ar_1 + (E_{pq}) \overset{K_{Ar}(p,q)}{\rightarrow} Ar_q + Ar_1, \tag{8.5}$$

Lower level p	Upper level q	$b_{pq}[\mathrm{m^2\,J^{-1}}]$
$4s[3/2]_2$	$4s[3/2]_1$	$1.12 \times 10^{-5} \hat{E}_{pq}^{-2.26}$
$4s[3/2]_2$	$4s'[3/2]_0$	$3.0 \times 10^{-7} \hat{E}_{pq}^{-2.26}$
$4s[3/2]_2$	$4s'[3/2]_1$	$3.0 \times 10^{-7} \hat{E}_{pq}^{-2.26}$
$4s[3/2]_1$	$4s'[3/2]_0$	$3.0 \times 10^{-7} \hat{E}_{pq}^{-2.26}$
$4s[3/2]_1$	$4s'[3/2]_0$	$3.0 \times 10^{-7} \hat{E}_{pq}^{-2.26}$
$4s'[1/2]_0$	$4s'[3/2]_1$	$1.12 \times 10^{-5} \hat{E}_{pq}^{-2.26}$

Table 8.2. *Fit parameters for heavy particle excitation rates (Eqn.* (8.6)*) between 4s levels.*

where an Ar atom is excited from level p to level q by collision with an Ar atom in the ground state (1). From Vlček et al. [38] the following rate coefficient is used:

$$K_{Ar}(p,q) = 4\sqrt{\frac{k_B T_h}{\pi M_{Ar}}}\, b_{pq} \left(E_{pq} + 2k_B T_h\right) \exp\left(\frac{-E_{pq}}{k_B T_h}\right), \tag{8.6}$$

with b_{pq} (unit $\mathrm{m^2\,J^{-1}}$) a fit parameter depending on the upper and lower level. Except for the transitions listed in table 8.2 dealing with transitions between 4s levels, its value is $b_{pq} = 5.42 \times 10^{-3} \hat{E}_{pq}^{-2.26} \mathrm{m^2\,J^{-1}}$, with the energy in electronvolt.

The reverse process of Eqn. (8.5), heavy particle de-excitation, is also included. Its rate is derived from the excitation rate by use of detailed balancing, where we assume that the heavy particle energy distribution function is Maxwellian (see also Eqn. (4.29)):

$$K_{Ar}(q,p) = K_{Ar}(p,q) \frac{g_p}{g_q} \exp\left(\frac{E_{pq}}{k_B T_h}\right). \tag{8.7}$$

Following Vlček et al., Eqn (8.6) is also used for heavy particle ionization:

$$Ar_p + Ar_1 + (E_{p+}) \overset{K_{Ar}(p,+)}{\rightarrow} Ar^+ + e + Ar_1. \tag{8.8}$$

For all levels p the fit parameter is $b_{p+} = 5.42 \times 10^{-3} \hat{E}_{p+}^{-2.26} \mathrm{m^2\,J^{-1}}$.

Additionally, heavy particle recombination, the reverse process of Eqn. (8.8), is included. As demonstrated by Collins [81] the Saha balance can be employed to determine the rate coefficient for recombination, which results in:

$$K_{Ar}(+,p) = K_{Ar}(p,+) \frac{g_p}{2g_+} \left(\frac{h^2}{2\pi \mathrm{m}_e k_B T_e}\right)^{3/2} \exp\left(\frac{E_{p+}}{k_B T_h}\right). \tag{8.9}$$

8.2.5 Laser induced processes

The last, but most essential processes included in the CRM are those induced by the laser. The plasma is targeted by a laser that is tuned to a specific transition in

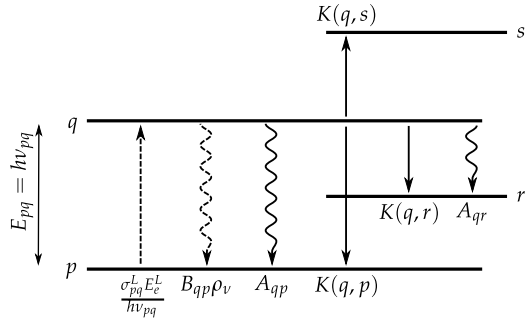


Figure 8.2. *Schematic depiction of the processes involved in a LIF process. Level p is excited to level q due to absorption of a photon from the laser I_L. Simultaneously the laser causes stimulated emission of level q at a frequency $B^\nu_{qp}\rho_\nu$. Furthermore, level q is depopulated by spontaneous emission to p and other lower levels r, and by electron (de-)excitation to levels p, r and s.*

the system. In Fig. 8.2 a schematic depiction of an atomic system is shown, that is exposed to a laser tuned to the *p*-*q* transition ($E_{pq} = h\nu_{pq}$).

Atoms in level p absorb a photon from the laser, exciting the atoms to level q. At the same time the laser photons cause stimulated emission of atoms in level q, forcing a decay to level p. Meanwhile, level q is also depopulated by all the usual processes: spontaneous radiative decay to lower levels (q and r), excitation to higher levels (s), and de-excitation to lower levels (p and r). The same applies, of course, to level p.

The rate coefficient for population of q due to the laser is described by:

$$\mathcal{F}^L(p,q) = \frac{\sigma^L_{pq}E^L_e}{h\nu_{pq}}, \tag{8.10}$$

with $h\nu$ the photon energy, E^L_e the laser irradiance in $\mathrm{W\,m^{-2}}$, and σ^L_{pq} the laser absorption cross section. The absorption cross section is proportional to the Einstein coefficient for absorption B_{pq}, which is related to the Einstein coefficient for stimulated emission B_{qp} by:

$$B_{qp} = \frac{g_p}{g_q}B_{pq}, \tag{8.11}$$

with g the degeneracy of a level. This means that the population of level q by a laser with irradiance E^L_e can be described by:

$$\frac{dn_q}{dt} = \frac{\sigma^L_{pq}E^L_e}{h\nu_{pq}}n_p - \frac{\sigma^L_{qp}E^L_e}{h\nu_{pq}}n_q = \mathcal{F}^L_{pq}\left(n_p - \frac{g_p}{g_q}n_q\right) \tag{8.12}$$

Similarly, the depopulation of level p can be described by:

$$\frac{dn_p}{dt} = \mathcal{F}^L_{pq}\left(\frac{g_p}{g_q}n_q - n_p\right). \tag{8.13}$$

The LIF process is added to the CRM transition matrix $\boldsymbol{F}$ (see Eqn. (4.57)) in the form of four elements:

$$\begin{aligned} F_{pp} &= F_{pp} - \mathcal{F}^L_{pq} & F_{pq} &= F_{pq} + \mathcal{F}^L_{pq}\frac{g_p}{g_q} \\ F_{qp} &= F_{qp} + \mathcal{F}^L_{pq} & F_{qq} &= F_{qq} - \mathcal{F}^L_{pq}\frac{g_p}{g_q} \end{aligned} \tag{8.14}$$

The CRM allows for a time dependent pulse with any temporal shape. Here, we will assume that the laser irradiance is in the form of an arbitrarily high rectangular pulse in a block shape. The result will be that during the pulse levels p and q are *saturated*. In Eqns. (8.12) and (8.13) the influence of only the laser pulse on the level population is given. As shown in Fig. 8.2, there will also be other processes influencing the densities of these levels. If the laser intensity I_L is high enough, these other processes can be neglected, resulting in a quasi steady state of levels p and q:

$$\frac{n_q}{g_q} = \frac{n_p}{g_p}. \tag{8.15}$$

When levels p and q are saturated, their frequencies in the transition matrix, Eqn. (8.14), cancel each other out in the source vector $\mathbf{S} = \boldsymbol{F}\mathbf{n}$ (see Eqn. (4.2)). The result is that the CRM becomes a system of balance equations with the added constraint of Eqn. (8.15). During the pulse the system will evolve towards a new steady state. The mechanisms driving this evolution are the radiative and collisional (electron and heavy particle) processes in the plasma.

When the pulse has ended the system will return to the initial state, that is the state of the system before the laser pulse entered the plasma. However, before this condition can be reached the system has to get rid of the fast density change created by the laser.. This disruption will spread through the system, which can also be experimentally observed by monitoring the transient spectra. As the disruption travels through the different layers of the atomic system, spectral lines will respond in some manner to the changes in level densities, thereby providing a wealth of information on the various transitions in the atomic system.

8.3 CRM results

The Ar CRM described in the previous section has been implemented using the PLASIMO CRM module. The densities of the ground level, ion level, and electrons,

and the electron and heavy particle temperature* can freely be adjusted. Furthermore, the temporal behavior of the laser pulse can be given any desired shape. For simplicity (see previous section) the laser intensity is assumed to be in the form of a rectangular function with an arbitrarily high value with near infinite slopes. The result is that when the laser is switched on, the lower and upper levels of the transition stimulated by the laser are instantly in saturation. Saturation will then remain until the pulse ends.

The initial densities of the Ar system are calculated using the QSS solution (see section 4.2.1). The subsequent evolution of the system influenced by the LIF process is modeled using the time dependent solver (see section 4.2.2).

We will first examine what happens to the saturation level during the pulse, and we will do this for different electron densities.

8.3.1 LIF saturation

During the pulse, saturation is reached (Eqn. (8.15)). However, this does not mean that during the laser pulse the level densities will not change. The system will evolve towards a new steady state, due to the (de)population mechanisms present in the system. Since mainly the electrons facilitate transitions between levels, the electron density will be an important factor in the relaxation of the system toward a steady state.

Figure 8.3 shows the evolution of the weighted density ($\eta = n/g$) of the upper and lower level†. The laser is tuned to a wavelength of $\lambda_{pq} = 696.7\,\text{nm}$, which corresponds to the transition from $4p'[1/2]_1$ to $4s[3/2]_2$ (in Paschen notation $2p_2$ to $1s_5$). The η values are scaled for easier comparison of the curves.

A 10 ns laser pulse starts at 1 ns, causing instant saturation. When the saturation density is reached instantly, the sum of the two level densities does not change, meaning that:

$$n_p^0 + n_q^0 = n_p^s(t) + n_q^s(t), \tag{8.16}$$

with n^0 the density in steady state before the pulse, and $n^s(t)$ the saturation density. Combined with Eqn. (8.15) this results in the weighted saturation densities:

$$\eta_p^s = \eta_q^s = \frac{1}{g_p + g_q}(g_p\eta_p^0 + g_q\eta_q^0). \tag{8.17}$$

When we define:

$$\Delta_\eta^0 = \eta_p^0 - \eta_q^0, \tag{8.18}$$

this can be written as:

$$\eta_p^s = \eta_q^s = \frac{1}{g_p + g_q}\left(g_p(\eta_q^0 + \Delta_\eta^0) + g_q\eta_q^0\right) = \eta_q^0 + \frac{g_p}{g_p + g_q}\Delta_\eta^0. \tag{8.19}$$

*In this chapter the electron energy distribution function is assumed to be Maxwellian.

† Note that the lower level has a higher density. In a density plot the lower (in energy) level will lie above the upper (in energy) level.

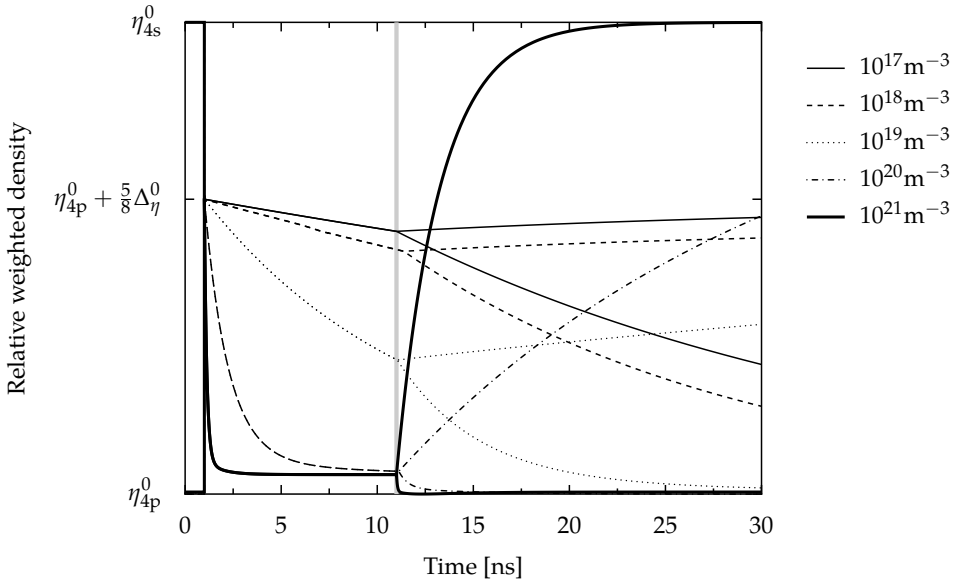


Figure 8.3. *Influence of the electron density on the saturation density. Shown are the weighted densities ($\eta = n/g$) of the levels* $4s[3/2]_2$ *and* $4p'[1/2]_1$ *($1s_5$ and $2p_2$ in Paschen notation) as a function of time for different electron densities. The weighted densities have been scaled so that at time $t = 0$ the overlapping η values for the 4p level are at the bottom of the graph, and for the 4s level at the top. The laser pulse (starting at 1 ns and ending at 11 ns, gray line) depopulates the 4s level in favor of the 4p level, until saturation is reached: $\eta_p = \eta_q$ or $n_q/g_q = n_p/g_p$. During saturation the 4s and 4p curves overlap. After the pulse we see a bifurcation; the lower level (4s) density goes up while the density for the upper level (4p) goes back down again, both returning to their initial value (though not visible for low electron densities).*

For the 696.7 nm line we have $p = 4s[3/2]_2$ and $q = 4p'[1/2]_1$ so $g_p = 5$ and $g_q = 3$. The saturation level, also shown in Fig. 8.3, is therefore at $\eta^0_{4p} + \frac{5}{8}\Delta^0_\eta$.

The density of the 4p levels is much lower than for the 4s levels. The calculation of the saturation density shows that the upward jump of the 4p level is far greater than the downward drop of the 4s level. Since $\eta_{4p} \ll \eta_{4s}$, the density difference, Eqn. (8.18), will be $\Delta_\eta \approx \eta_{4s}$. The density change factor of the 4s level will be approximately 5/8, but for the 4p level the density can increase with a far higher factor, even up to 100.

From the curves in Fig. 8.3 it is clear that the electrons drive the system to a new equilibrium state, which for $n_e = 10^{21}\ \mathrm{m}^{-3}$ is already reached after 1 ns. After the pulse the system returns to the initial state, which again happens faster for higher electron density.

There is a clear difference in the time that the 4s and 4p levels require to re-

gain their initial value. The basic balance equation (3.23) (neglecting transport) describes the level densities:

$$\frac{\partial}{\partial t} n(p) = P(p) - n(p)\, D(p), \tag{8.20}$$

with $P(p)$ the total production and $D(p)$ the total destruction. When the steady state density is $n^0(p)$ and the density due to a disturbance is $n^*(p)$ the density returns to steady state according to:

$$n(p)(t) = n^0(p) + \Big(n^*(p) - n^0(p)\Big)\, \exp\Big(-D(p)\, t\Big). \tag{8.21}$$

Since for the 4p level $D(p)$ is higher than for the 4s level, the 4p level will restore faster to its initial density.

The results show that, only for low electron density and short pulse length the theoretical saturation density of Eqn. (8.17) can be used to approximate the disruption in system densities due to the laser pulse. An accurate determination of the saturation density requires a CRM incorporating the (de)population processes influencing the levels during the pulse.

8.3.2 System response

We will now use the CRM to model the response to the laser pulse of other levels in the system. Instead of focusing on the direct fluorescence of the upper level, the *first order* response, we will examine the response of other levels, that are directly or indirectly connected to the upper (and lower) level. This is the *second order* response or Laser Collisional Induced Fluorescence (LCIF) [82, 83, 84].

As in the previous section, the LIF process in the Ar CRM will populate the $4p'[1/2]_1$ ($2p_2$) level in favor of the $4s[3/2]_2$ ($1s_5$) level, corresponding to the wavelength $\lambda = 696.7\,\text{nm}$. Instead of the radiative transitions from $4p'[1/2]_1$, we will examine the radiative transitions from other 4p levels. The other 4p levels can be affected by the laser pulse in several ways, as depicted in Fig. 8.4.

The laser pulse has two direct consequences: the lower level density ($4s[3/2]_2$) is decreased and the upper level density ($4p'[1/2]_1$) is increased. As in the previous section we will assume that the laser pulse is energetic enough to instantly achieve saturation. The indirect result is that population of other 4p levels (either direct, via higher levels, or via lower levels) due to the increase of $4p'[1/2]_1$ is increased. At the same time, population of 4p levels is decreased since population from $4s[3/2]_2$ (also involving other intermediate levels) is decreased.

The direct reaction to the laser can be seen in Fig. 8.3: the 4s and 4p level reach saturation and after the pulse the 4p level recovers quickly, and the 4s level slowly. Combined this means that other 4p levels, and therefore the spectral lines from other 4p levels, will first increase, then decrease below the steady state value,

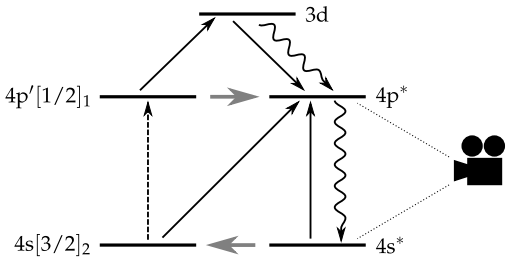


Figure 8.4. *Schematic depiction of an LCIF scheme. A laser excites level* $4s[3/2]_2$ *to* $4p'[1/2]_1$ *resulting in a decreased density of the 4s level and increased density of the 4p level. Other 4s and 4p levels (denoted by * so they can represent any other 4s and 4p level) are linked to these two levels through heavy particle collisions (gray arrows), electron collisions (black solid arrows), and radiative transitions (squiggly arrow), possibly involving other (higher) levels (here only level 3d is shown). The response to the laser pulse of some other radiative transition (4p* to 4s*) is recorded.*

to which they will eventually recover. The exact nature of this double response depends on the rates that form the paths between the levels.

To simulate the responses of spectral lines, the following settings were used in the CRM:

- ground state density: $9.7 \times 10^{22}\,\mathrm{m}^{-3}$ (4 mbar);
- electron density: $6.8 \times 10^{18}\,\mathrm{m}^{-3}$;
- electron temperature: 1.3 eV;
- cut-off level: 6d;
- laser pulse length 8 ns.

The responses of four lines are shown in Fig. 8.5.

All lines modeled by the CRM show a similar response, i.e. a sharp increase, a fast decay to below the equilibrium value, followed by a slow rise. There is little difference in the decay time following the sharp peak. The lines mostly differ in the depth and rise time of the negative response, which is caused by the different rates by which the depleted $4s[3/2]_2$ is linked to the radiating 4p levels.

8.3.3 Comparison to experiments

The spectral lines shown in Fig. 8.5 have also been measured in our group [54], with the same parameters. The results are shown in Fig. 8.6. Apart from the lines shown, many more lines were measured, but we will restrict ourselves to these four lines.

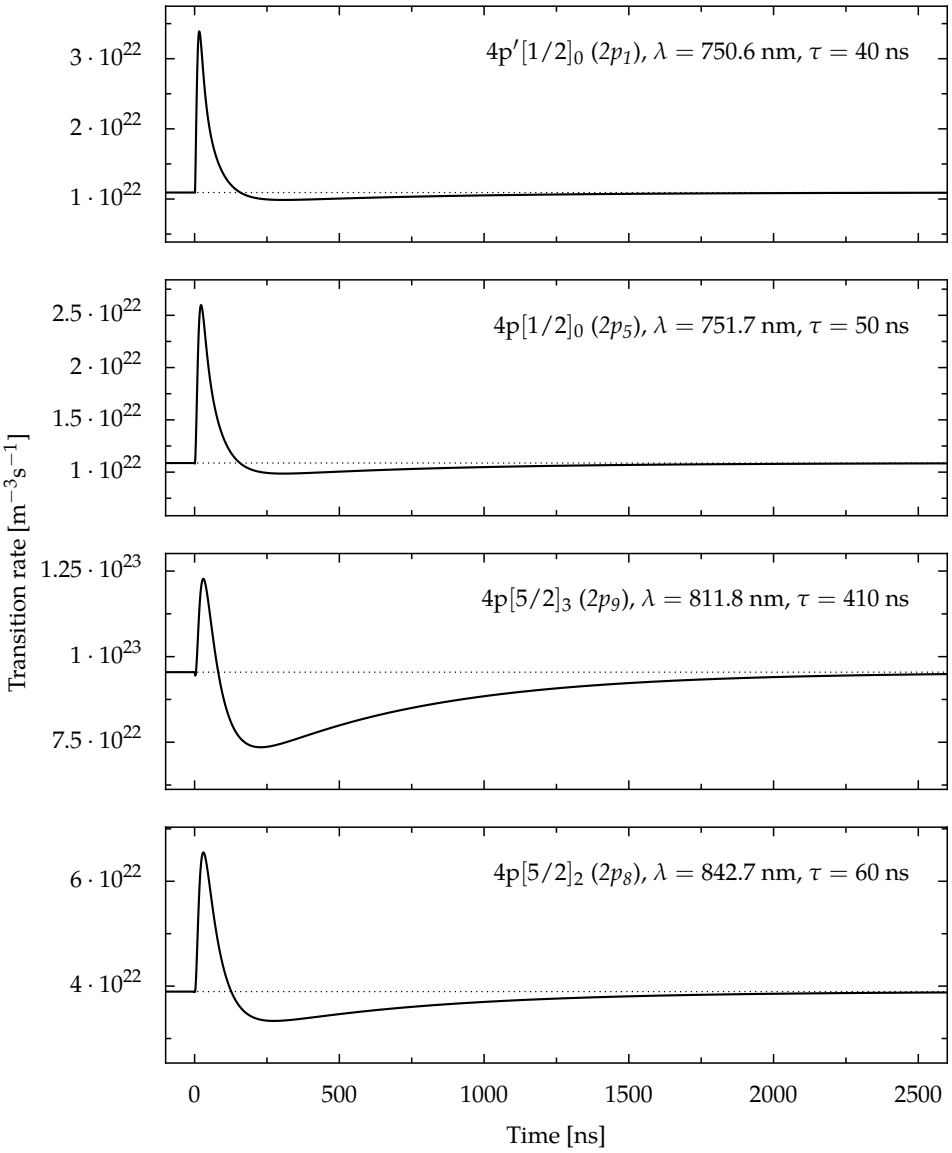


Figure 8.5. *The temporal behavior of the response of four spectral lines as calculated by the CRM. Listed are the radiating level in Racah notation, Paschen notation between parentheses, wavelength, and decay time. In the plot for the 811.8 nm line the time is the rise time. None of these lines originate from the pumped 4p level ($4p'[1/2]_1$).*

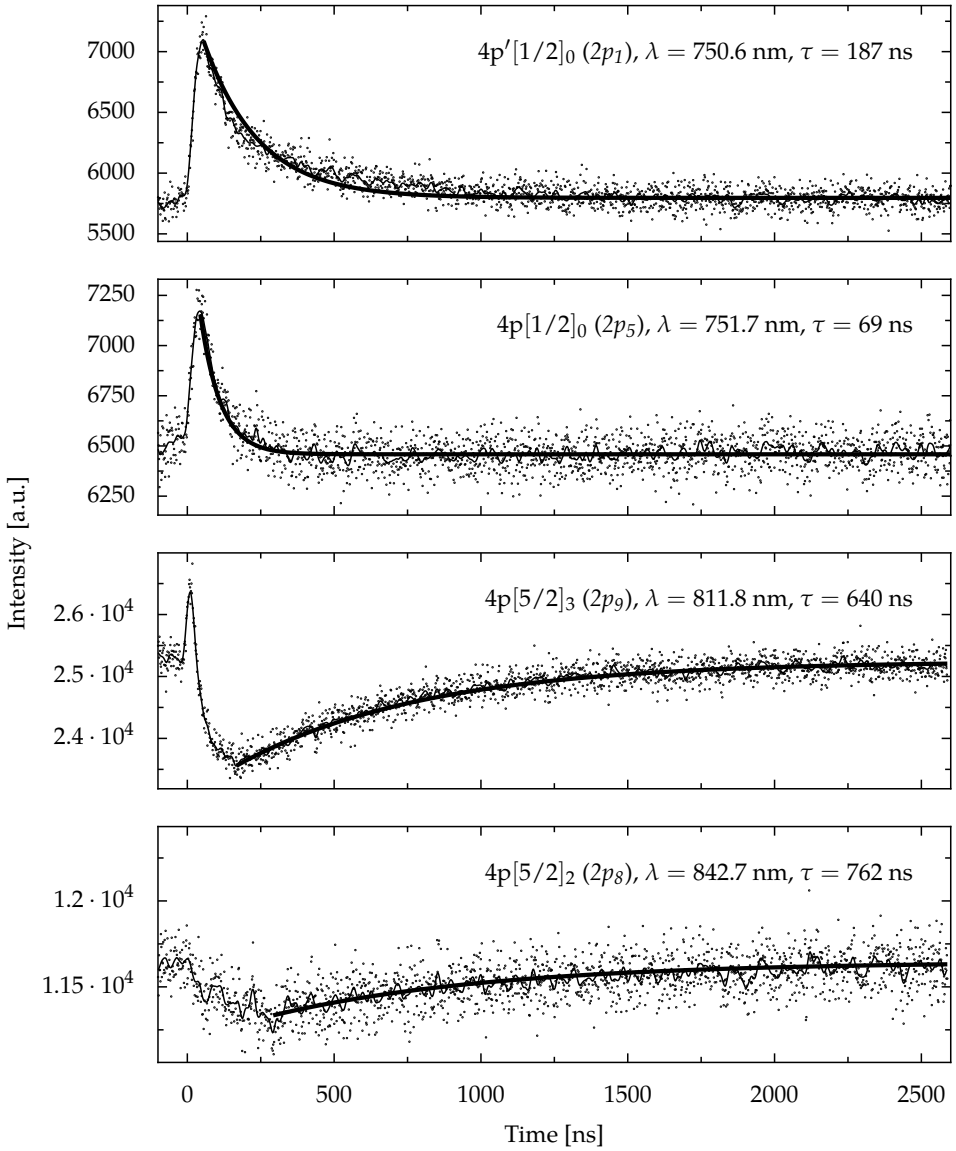


Figure 8.6. *The temporal behavior of the response of four spectral lines (see also Fig. 8.5). Listed are the radiating level in Racah notation, Paschen notation between parentheses, wavelength, and decay time. In the bottom two plots (811.8 nm and 842.7 nm) the time is the rise time. None of these lines originate from the pumped 4p level ($4p'[1/2]_1$). An exponential fit using τ is plotted as a thick black line.*

The measurements were performed on a Surfatron Induced Plasma (SIP) [85] for which the electron density and temperature, obtained by Thomson scattering, are well known [86, 87]. In this plasma the electron density can easily be controlled and adjusted. The laser pulses are generated with a high rep-rate YAG-Dye laser system. The spectral line responses are recorded by a Multi Channel Scaler connected to a monochromator.

The measured response of the 751.1 nm and 811.8 nm lines show reasonable agreement with the behavior predicted by the model. However, all the 4p lines modeled by the CRM decay with a similar time (30 ns to 60 ns), whereas in the experiments a wide range of 69 ns to 412 ns has been found. Furthermore, all modeled lines show a peaked, positive response followed by a (mostly shallow) negative response. In the experiments the shallow, negative response is often not observed, possibly due to noise. What is even more striking is that some lines do not show the initial positive response, as the line at 842.7 nm. Also the magnitude of the response is in most cases much larger in the experimental results than in the measurements.

Analysis of the CRM reveals that the 3d (and higher) levels form an important conduit for the population of the 4p levels. However, experimental data for these collisional transitions is not available. In the CRM, Drawin cross sections with calculated fit parameters are used. Since those fit parameters are only available for the 3d levels as a block, the initial, positive responses of all 4p levels in the model are very similar. From the experimental results we see that in reality things are more refined, i.e., different 4p-3d transitions have very different cross sections.

8.4 Conclusion

The plasimo CRM module has been used to model a LIF experiment in an Ar plasma. Since (de)population mechanisms above the 4p levels are involved in this process, a model of a complete Ar system has been constructed, using data from various sources.

The time dependent CRM enables us to model the laser pulse in a LIF experiment in great detail. It is shown that due to electron collisions the saturation density that is reached during the laser pulse does not stay constant, but decays to an equilibrium value. A CRM is required to determine the saturation density at the end of the pulse.

LIF is a very good tool to investigate how excitation channels build up the atomic system. The CRM is a useful tool in analyzing these channels. Rates between levels can be accessed and analyzed individually and time resolved. However, the experimental results lay bare the shortcomings of the CRM: the experimental data it is based on. In LCIF, rates above the 4p levels play an important role, but no experimental data is available for these levels. Combination with ex-

periments can improve the accuracy of the CRM, and enhance scientific insight into the system.

Chapter 9

A global model of HiPIMS discharges

9.1 Introduction

In chapter 2 an overview was given of types of zero-dimensional models for plasma chemistry. Chapters 3, 4, and 5 contain a detailed description of a Collisional Radiative Model (CRM) and its implementation in the plasimo framework. Applications of this CRM code were treated in chapter 7 and 8 where a CRM of an EUV driven plasma and a CRM of a LIF experiment were described. Chapter 6 deals with the implementation of a Global Plasma Model (GPM) code in the plasimo framework.

In this chapter we will use the GPM to construct a model of a High Power Impulse Magnetron Sputtering (HiPIMS) plasma. It is the implementation of a model by Gudmundsson [88], and serves as a proof of concept for the GPM module in plasimo. The parameters (species, reactions, and input parameters) are taken from the model by Gudmundsson, but we will elaborate on them.

HiPIMS, also denoted by High Power Pulsed Magnetron Sputtering (HPPMS), was introduced by Koeznetsov et al [89] in 1999. The method is based on the fact that by applying high power in the form of short and very intense pulses to a magnetron plasma, a very high plasma density can be reached. Whereas in Conventional DC Magnetron Sputtering (CDCMS) densities of 10^{15} m^{-3} can be reached, this number is increased to more than 10^{18} m^{-3} at 5 mTorr in HiPIMS [90]. The discharge is created by applying high voltage (500 V to 2000 V), short (50 µs to 500 µs), unipolar pulses with a low duty cycle (1 Hz to 1000 Hz) to the cathode target, resulting in a peak power that can reach values up to kilowatts and even megawatts.

The advantages of the HiPIMS technique are the uniform deposition of structures with high aspect ratios, interface modification through ion irradiation and increased film density. In general these advantages are attributed to the large degree of ionization of the sputtering flux. However, the reported ionization degree in literature is highly inconsistent, ranging from 4.5% from a C target [91], to more than 90% for a Ti target [92].

The goal of the GPM in this chapter is to explore the ionization mechanism and the temporal behavior of the plasma parameters. The ionization degree can easily be deduced from these results.

This chapter will continue with a description of the model: the species, the reactions between them, and their interaction with the wall. In section 9.3 some details of the implementation of this model for the GPM PLASIMO module are given. The results are shown in section 9.4.

9.2 Model

The plasma under investigation is an Ar plasma that sputters an Al surface. The surface is bombarded by Ar ions, thereby introducing Al atoms into the plasma. Subsequently, selfsputtering will also occur after Al atoms have been ionized. The model consists of balance equations for the densities of the following species:

- Ar atoms in the ground state: Ar, energy level: 0 eV;
- excited Ar atoms: Ar^*, energy level: 11.56 eV;
- Ar ions: Ar^+, energy level: 15.76 eV;
- Al atoms in the ground state: Al, energy level: 0 eV;
- Al ions: Al^+, energy level: 5.99 eV.

There is no separate density balance for the electrons. Since quasi-neutrality is assumed, the density of the electrons equals the sum of the Ar and Al ion densities: $n_e = n_{Ar^+} + n_{Al^+}$. The density of Ar ground state atoms is kept constant; i.e., the balance equation source term is kept to zero.

Additionally (see also chapter 6), the model contains a balance equation for the electron energy density, see Eqn. (6.12). The source terms for the balance equations are formed by reactions between these species and by processes involving the wall.

9.2.1 Volume relations

The reactions that occur in the bulk of the plasma are listed in table 9.1. All reactions and their respective rate coefficients can be entered as is into the GPM model,

Process *reaction*	Rate coefficient	Source
Ar ionization: $Ar + e \rightarrow Ar^+ + 2e$	$k_{\text{iz}} = 2.9 \times 10^{-14}\, \hat{T}_e^{0.50} \exp(-17.8/\hat{T}_e)$	[93]
Ar excitation: $Ar + e \rightarrow Ar^* + e$	$k_{\text{exc}} = 6.37 \times 10^{-15} \exp(-12.53/\hat{T}_e)$	[94]
Ar excited ionization: $Ar^* + e \rightarrow Ar^+ + 2e$	$k_{\text{exc, iz}} = 6.8 \times 10^{-15}\, \hat{T}_e^{0.67} \exp(-4.20/\hat{T}_e)$	[95]
Ar de-excitation: $Ar^* + e \rightarrow Ar + e$	$k_{\text{deexc}} = 4.3 \times 10^{-16}\, \hat{T}_e^{0.74}$	[16]
Ar elastic collisions: $Ar + e \rightarrow Ar + e$	$\ln(k_{\text{elas}}) = -31.3897 + 1.6090 \ln(\hat{T}_e) + 0.0618 \ln(\hat{T}_e)^2 - 0.1171 \ln(\hat{T}_e)^3$	[93]
Ar radiative decay: $Ar^* \rightarrow Ar + h\nu$	$k_{\text{rad}} = 3.15 \times 10^8$	[66]
Al ionization: $Al + e \rightarrow Al^+ + 2e$	$k_{\text{iz},Al} = 1.23 \times 10^{-13} \exp(-7.23/\hat{T}_e)$	[96]
Al Penning ionization: $Ar^* + Al \rightarrow Al^+ + Ar$	$k_{\text{P}} = 5.9 \times 10^{-16}$	[97]
Charge exchange: $Ar^+ + Al \rightarrow Ar + Al^+$	$k_{\text{chex}} = 1 \times 10^{-15}$	[97]

Table 9.1. *The volume reactions included in the model and their rate coefficients. The electron temperatures are in electronvolt.*

except for the elastic collisional process. In this process electrons and ground state Ar atoms are involved, but the source terms of neither of these species will get an extra term due to this process, since the net stoichiometry is zero. However, the electron energy density balance should receive a source term from this process, therefore it will be implemented as an extra source term, (see also section 6.3.2). The extra source term for elastic collisions is defined as:

$$\mathcal{S}_{\text{elas}}^{\text{extra}} = -k_{\text{elas}} n_{Ar} n_e \frac{3m_e}{m_{Ar}} k_B T_e. \tag{9.1}$$

The processes in table 9.1 describe the *chemistry* in the bulk of the plasma. Apart from this chemistry, species will be transported to the wall and species will be introduced into the bulk from the wall. This means that *transport* and *configuration* aspects come into play, but they are combined and treated as frequencies.

9.2.2 Wall losses

Loss of neutral species is described by simple diffusion:

$$X \rightarrow X_{\text{wall}}, \tag{9.2}$$

where X represents either excited Ar (Ar^*) or metal atoms (Al). These two diffusion reactions make it necessary to introduce two more species into the model: Ar_{wall} and Al_{wall}. The rate coefficient for the diffusion reactions is:

$$k_{\text{diff},X} = \frac{D_X}{\Lambda^2}, \tag{9.3}$$

with D_X the diffusion coefficient, and Λ the effective diffusion length. The diffusion coefficient is given by:

$$D_X = \frac{k_B T_h \lambda_X}{m_X v_{\text{th}}}, \tag{9.4}$$

where we use the thermal velocity $v_{\text{th}} = \sqrt{8 k_B T_h / \pi m_X}$, and the mean free path $\lambda_X = (n_{Ar}\sigma_{X,Ar})^{-1}$, with $\sigma_{X,Ar}$ the X-Ar scattering cross section for which a value of $10^{-18}\,\text{m}^2$ is assumed.

The configuration aspects of the plasma are represented in the effective diffusion length Λ. The shape of the plasma is cylindrical with radius R and length L, resulting in:

$$\Lambda = \left[\left(\frac{\pi}{L}\right)^2 + \left(\frac{2.405}{R}\right)^2 \right]^{-1/2}. \tag{9.5}$$

As is evident from table 9.1, recombination in the bulk is not included, but charged particles are lost at the wall. Furthermore it is assumed that ion and electron fluxes toward the wall balance at all times.

For the treatment of the transport of charged particles we follow Godyak et al [21] who gave an analytical solution for the diffusion equation, resulting in the plasma densities at the sheath edges (see also section 2.4):

$$\frac{n_{\text{sheath},L}}{n_{\text{bulk}}} = h_L \approx 0.86 \left(3.0 + \frac{L}{2\lambda_i}\right)^{-1/2}, \tag{9.6}$$

at the axial sheath edge and:

$$\frac{n_{\text{sheath},R}}{n_{\text{bulk}}} = h_R \approx 0.80 \left(4.0 + \frac{R}{2\lambda_i}\right)^{-1/2}, \tag{9.7}$$

at the radial sheath edge. A value of $10^{-18}\,\text{m}^2$ is used for all ion-neutral cross sections to obtain λ_i.

At the sheath edges the fluxes of ions leaving the plasma will be:

$$\Gamma_i = n_i\, u_{B,i}, \tag{9.8}$$

with $u_{B,i} = \sqrt{k_B T_e / M_i}$ the Bohm velocity. For the ionic species i (Ar^+ and Al^+) the balance of generation in the bulk and losses at the wall is then:

$$V \frac{\partial n_i}{\partial t} = V \sum_j \mathcal{S}_j - (h_L A_L + h_R A_R) n_i u_{B,i}, \tag{9.9}$$

where the sum runs over all volume reactions in table 9.1 that form a source or sink for the ionic species, V is the bulk volume, and A_L and A_R are the areas of the axial and radial sheath. It is assumed that the sheath thickness is negligible, so that $A_L = 2\pi R^2$ and $A_R = 2\pi RL$. Dividing by the volume $V = \pi R^2 L$ the rate coefficients for wall losses of the ions are:

$$k_{\text{wall},i} = \frac{h_L R^2 + h_R RL}{R^2 L} 2u_{B,i}, \tag{9.10}$$

with i being either Ar^+ or Al^+.

Metal atoms are however not only lost at the wall, they are also introduced into the bulk of the plasma by sputtering.

9.2.3 Sputtering

To determine the flux of particles entering the bulk from sputtering, the flux of ions on the surface is required and the sputter yield from those ions. The rate coefficient for losses to the target, a disc with radius R_T, is in analogy with Eqn. (9.10):

$$k_{\text{target},i} = \frac{h_L R_T^2}{R^2 L} u_{B,i}, \tag{9.11}$$

which multiplied by the sputter yield gives the rate coefficient of the source from sputtering. The target surface is sputtered by both ion species: Ar^+ and M^+. Using the sputter yield γ_{sputter} by Ar^+ and $\gamma_{\text{selfsputter}}$ by M^+ the total source term from sputtering for species M is:

$$\mathcal{S}_{\text{sputter},Al} = \frac{h_L R_T^2}{R^2 L} \left(\gamma_{\text{sputter}} u_{B,Ar^+} n_{Ar^+} + \gamma_{\text{selfsputter}} u_{B,Al^+} n_{Al^+} \right). \tag{9.12}$$

The sputter yield is the average number of atoms ejected from the target per incident ion, which depends on the energy of the incident ion. However, in this model there is no self-consistent way to determine the energy of the ions upon impact of the target. It is assumed that the sputter yields depend on the target voltage which was measured during an experiment. The voltage as function of time for a background pressure of 10 mTorr is shown in Fig. 9.1 together with the sputter yields derived from this voltage when combined with sputter yield data collected by Ruzic [98] and Hayward et al. [99].

The source terms by sputtering will be added as extra source terms.

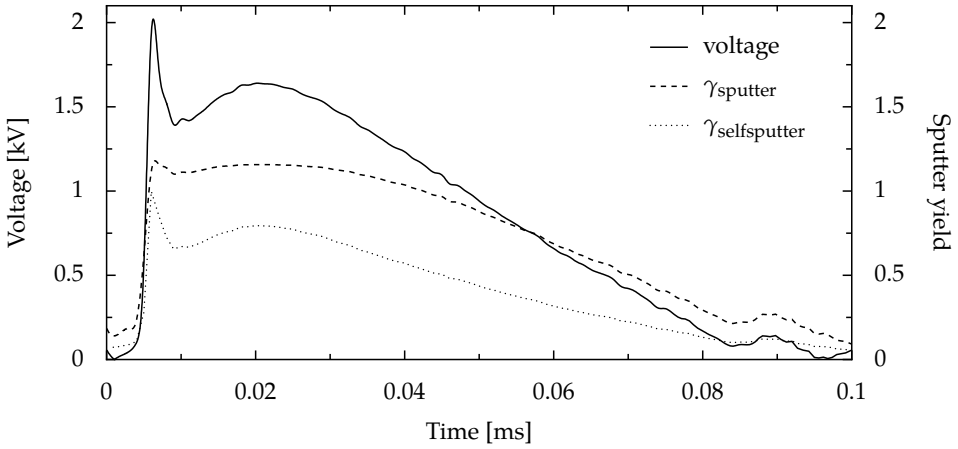


Figure 9.1. *Experimentally determined target voltage at 10 mTorr as a function of time, and the sputter yields derived from the voltage.*

9.2.4 Electron energy density balance

The reactions in table 9.1 involving electrons will form source terms for the electron energy density balance. These source terms are automatically calculated by the GPM, apart from the term for elastic collisions which is implemented as an extra source (Eqn. (9.1)).

Charged particles flowing to the wall also contribute in the form of a negative term to the energy balance. The source term is:

$$\mathcal{S}_{\text{wall},E} = -k_{\text{wall},Ar^+} n_{Ar^+} \frac{3}{2} k_B (\varepsilon_e + \varepsilon_i), \tag{9.13}$$

where ε_e and ε_i are the mean kinetic energy per lost electron and ion respectively.

The last term that is added to the energy balance is the input power. This is a time dependent external control parameter shown in Fig. 9.2.

9.3 Implementation in GPM

The set of balance equations (density of every species and electron energy density) will be solved with the PLASIMO GPM module, described in chapter 6. All the reactions, extra source terms, and external control parameters were described in the previous section, making the actual implementation quite straightforward.

Figures 9.3 through 9.6 show screenshots of the PLASIMO Graphical User Interface (GUI). They show how the species (Fig. 9.3), volume reactions (Fig. 9.4), rate coefficients (Fig. 9.5), and extra source terms (Fig. 9.6) can be entered.

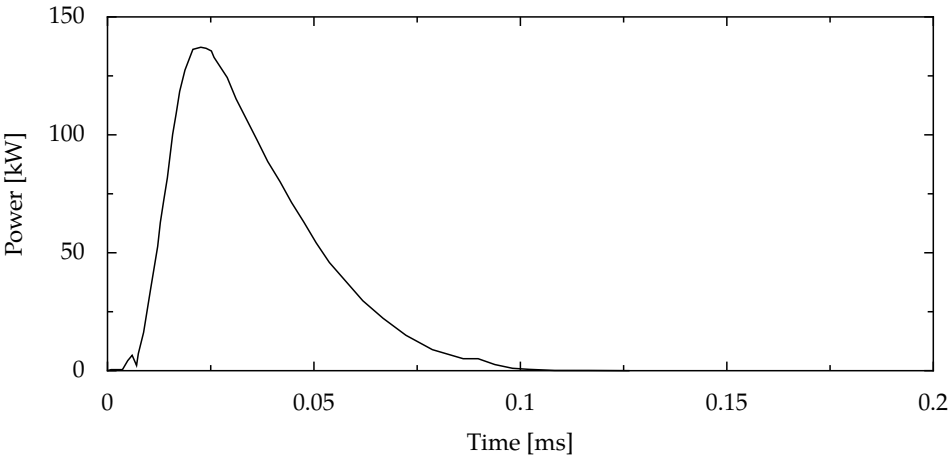


Figure 9.2. *Input power as function of time, implemented via a look-up table.*

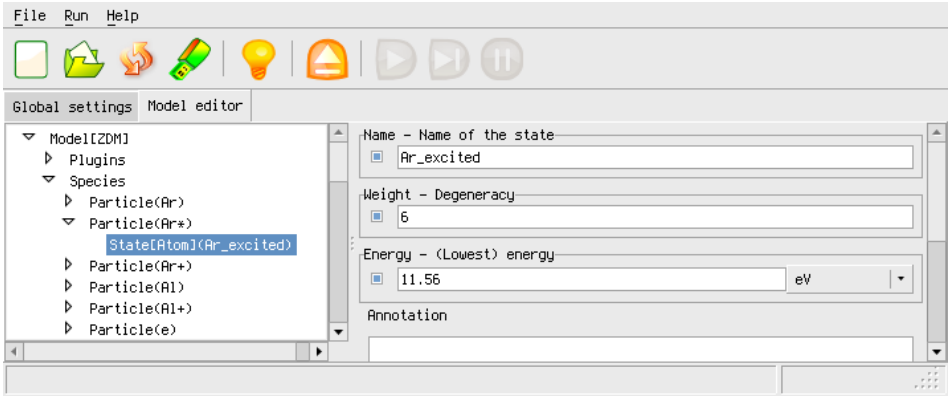


Figure 9.3. *Screenshot of the graphical* plasimo *application, showing how the species are entered in the "Model editor". Species can easily be added and edited in the GUI.*

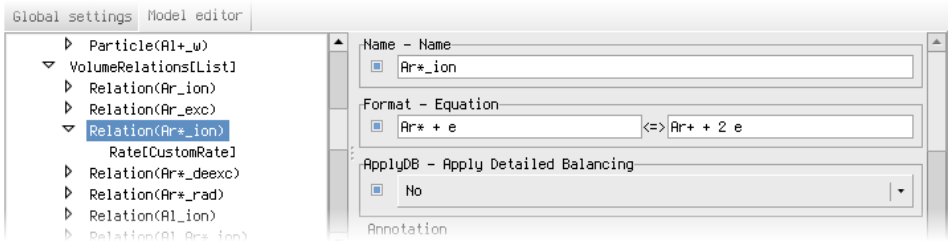


Figure 9.4. *Detail of a screenshot of* plasimo *showing how reactions can be entered; in this case the ionization of excited Ar atoms by electrons.*

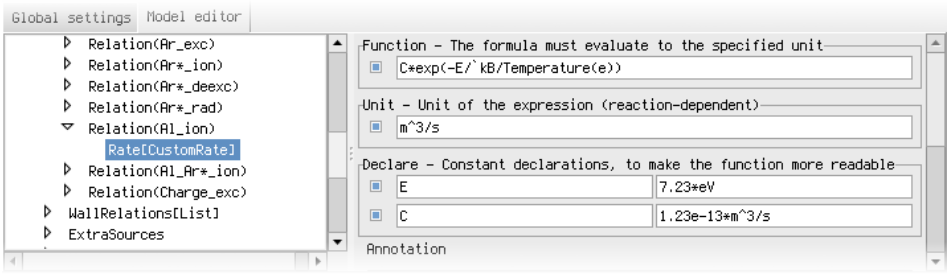


Figure 9.5. *Detail of a screenshot of* PLASIMO *showing how rate coefficients can be defined; in this case for the ionization of Al atoms by electrons. A general expression can be used with an arbitrary amount of parameters that can be defined separately.*

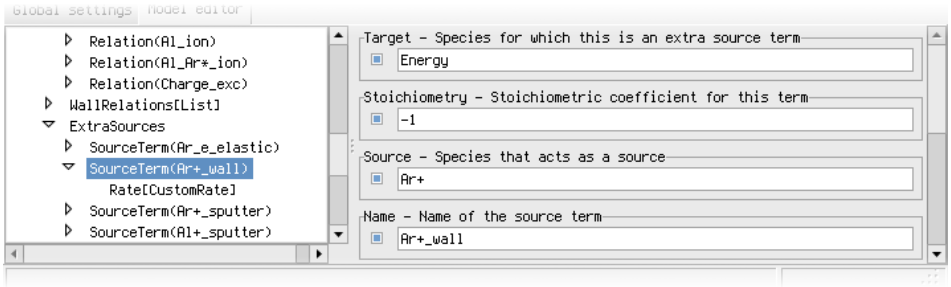


Figure 9.6. *Detail of a screenshot of* PLASIMO *showing how extra source terms are defined; in this case energy losses to the wall by diffusion of electron ion pairs. In addition to the entries shown, a rate coefficient must be defined, similar to the screenshot in Fig. 9.5.*

In addition to the data shown in the screenshots, the input power must be defined (in this case a look-up table in a file), and several other control parameters, the most important of which are the length $L = 15\,\text{cm}$ and radius $R = 15\,\text{cm}$ of the discharge chamber and the radius of the Al target $R_T = 7.5\,\text{cm}$. For all species an initial density must be defined. The background Ar density is set to $3.2 \times 10^{20}\,\text{m}^{-3}$ (10 mTorr at 300 K). The initial density of other species is set to zero, except for the Ar ions, for which the initial density is $10^{10}\,\text{m}^{-3}$. Other control parameters are for instance the run-time and solver. Once all the data is entered, the model can be run, and the calculation results are shown real-time via the GUI.

9.4 Results and discussion

The densities of the Ar ions, and Al atoms and ions are shown in Fig. 9.7. The peak densities and the times at which these are reached are listed in table 9.2. When comparing the results to those reported by Gudmundsson (also in table 9.2), we

	plasimo		Gudmundsson	
Species	Density [m^{-3}]	Peak time [µs]	Density [m^{-3}]	Peak time [µs]
Ar^+	7.2×10^{19}	60	3.4×10^{19}	53
Al^+	1.5×10^{18}	79	9.2×10^{17}	70
Al	3.7×10^{16}	91	1×10^{17}	41

Table 9.2. *Resulting densities as calculated by the GPM* plasimo *module and results as listed by Gudmundsson.*

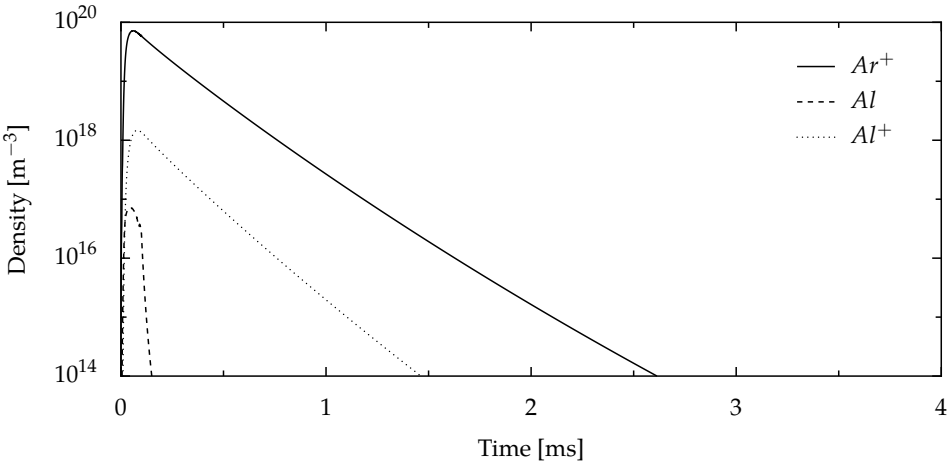


Figure 9.7. *Species densities as function of time of the Ar ions and Al atoms and ions.*

see that our ion densities are roughly twice as high, and peak slightly later. In contrast the Al atom density is lower and peaks after the ion densities, whereas in the results by Gudmundsson the peak lies before the ion density peaks. Furthermore, in Gudmundsson's results the densities decay much slower (by about a factor of three).

These differences can be most likely attributed to differences in the rate parameters that were used. In many cases we had to make assumptions for the rate coefficients or cross sections. The paper of Gudmundsson is not at all times quite clear about this.

The ionized Al fraction is shown in Fig. 9.8. Already evident from Fig. 9.7 the ionized fraction during the pulse is very high, approximately 0.98. Also shown in Fig. 9.8 is the fraction of ionized flux towards the wall. This is determined from the flux of Al ions $\Gamma_{Al^+} \approx 0.61 n_{Al^+} u_{B,Al}$, and Al atoms $\Gamma_{Al} = \frac{1}{4} n_{Al} v_{Al}$. Because the discharge is not in thermal equilibrium, i.e., the electron temperature T_e is

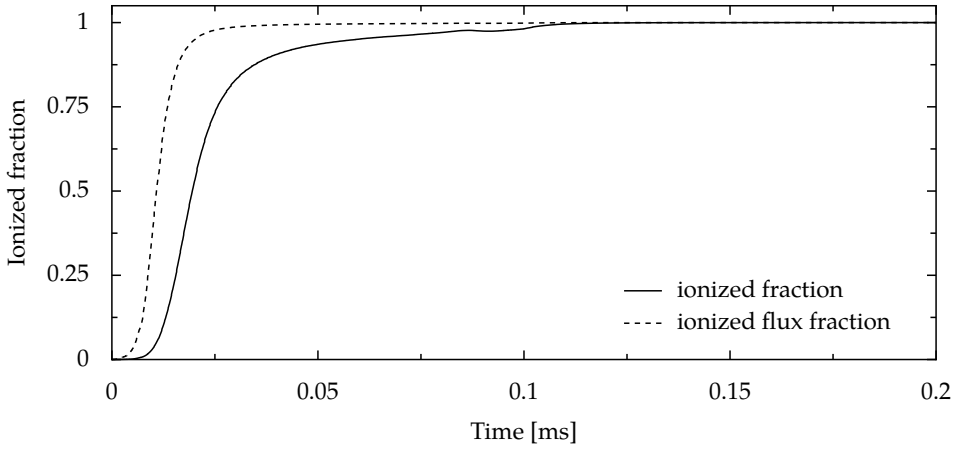


Figure 9.8. *Ionized volume Al fraction* $(n_{Al^+} / (n_{Al} + n_{Al^+}))$ *and ionized flux fraction* $(\Gamma_{Al^+} / (\Gamma_{Al} + \Gamma_{Al^+}))$ *as function of time.*

significantly larger than the neutral gas temperature T_h, the fraction of ionized metal flux is larger than the fraction of ionized metal in the plasma. These results compare very well to the results of Gudmundsson.

From Fig. 9.9 we can see that the dominant process for ionization of Al atoms is different before and after the pulse. Whereas during the pulse Al is mainly ionized by electron impact ionization, after the pulse the electron density has dropped so much that charge exchange with Ar ions becomes dominant. Furthermore, at no time Penning ionization is an important process.

In the results by Gudmundsson the time at which the rate for charge exchange dominates the rate for electron impact ionization is roughly 30 µs earlier, after which electron ionization quickly becomes negligible. This indicates that in Gudmundsson's results the electron temperature decays more quickly (the electron temperature is not reported).

9.5 Conclusion

The PLASIMO GPM module has been used to implement a zero dimensional model of a HiPIMS plasma that sputters an Al target. The required density balance equations, including the density balance equation for electron energy, are automatically constructed from the definition of a set of species and volume reactions. In addition some reactions have to be defined to deal with processes involving the wall. Diffusion to the wall is included by introducing a separate wall species. The more complex process of sputtering is dealt with by defining the process as an extra source term.

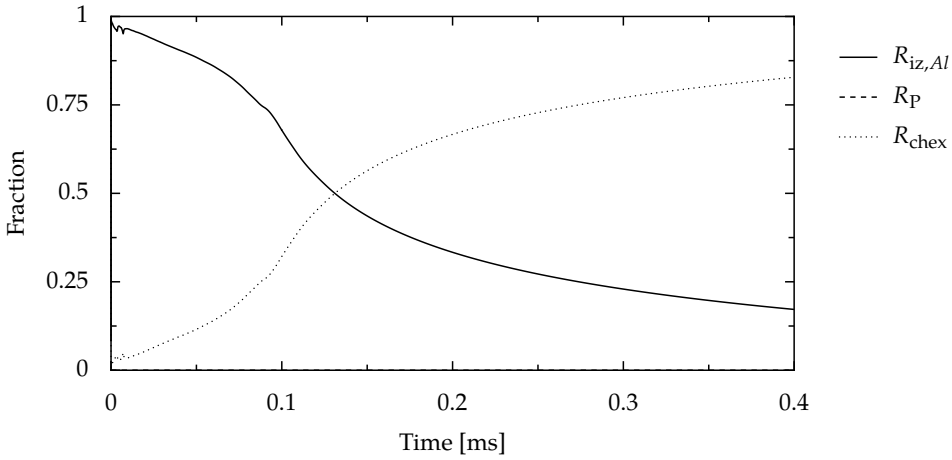


Figure 9.9. *Relative Al ionization mechanism as function of time. At all times Penning ionization (R_P) is negligible.*

The model showed that Al in the plasma and in the flux towards the target is almost fully ionized. Furthermore, during the pulse Al ions are mostly created by electron impact ionization of the sputtered Al atoms, and after the pulse mostly by charge exchange with Ar ions. Though these results were also found by Gudmundsson, there are also differences: the species densities have different peak values and transient behavior (time of peak and speed of decay). The most likely cause for this is a difference in the parameters that were used, since the model is very sensitive to changes in cross sections (and therefore rates).

The GPM module is a convenient tool to probe the effect of rates on the plasma behavior, giving valuable insights.

The plasimo interface offers a user friendly way of defining the model and manipulating its parameters. Parameters such as rate coefficients can easily be changed and the results of such changes can then be examined by viewing the resulting model data in the same plasimo interface. When rate coefficients and sputter rates for other species (e.g., Cu or W) are known, the model can effortlessly be adapted to accommodate for these other species, and after a run-time of mere minutes the results are presented.

This case study shows that the plasimo ZDM is a convenient, flexible tool for constructing general GPMs.

Chapter 10

General conclusions

The subject of this thesis is the documentation of the design and application of models that focus on chemistry aspects of plasmas. Other aspects, that can be classified as configuration and transport, are reduced to frequencies.

In chapter 2 a classification of zero-dimensional models was made. The focus lies on Collisional Radiative Model (CRM) and Global Plasma Models (GPM) employing the Reaction Exploration Method (REM). CRMs describe (mainly) atomic systems in which the processes are facilitated by external agents (photons, electrons), whereas GPMs are more general models that can describe any form of chemistry, while simultaneously solving the energy balance. Chapters 3, 4, and 5 were dedicated to a theoretical treatment and implementation of the CRM, and chapter 6 to the GPM.

In chapter 3 the three tasks of the CRM (to determine the Atomic State Distribution Function, effective conversion rates, and source terms for the energy balance) were formulated using simple matrix-vector representations. This was made possible by making a distinction between levels (species) for which transport is important and for which it can be neglected; the former being the Transport Sensitive (TS) levels, and the latter Local Chemistry (LC) levels. The matrix representation offers great flexibility, by allowing for a system with more than two TS levels (beyond the typical atom and ion ground state). This is demonstrated for a plasma in which radiation transport is important, which is dealt with by promoting an LC level to a TS level.

In chapter 4 we have described the processes that are included in the transition matrix that is required for the matrix representation of the CRM tasks, and how this matrix is constructed. The CRM has been implemented in the PLASIMO

framework, which is written in the C++ programming language. We have shown how the use of classes in C++ offers the possibility for a strong modular design. Modularity is a concept that is used throughout plasimo on all levels. We have shown how at the most basic level a plasimo model can be built, which we have applied to the CRM. The result is a flexible, general CRM code, that can be used for a wide range of plasmas. The CRM module supports steady state and transient CRMs, defined using rate coefficients or cross sections in combination with any form of EEDF.

In chapter 6 the plasimo module for GPMs was introduced. The GPM is defined by a set of species and general reactions between those species. The benefits of modularity have been demonstrated in this chapter, as many tools provided by plasimo are utilized to construct the GPM module.

The CRM and GPM plasimo modules are designed to be user friendly, general purpose tools. Chapters 7 and 8 describe applications of the CRM module, and chapter 9 of the GPM module.

Extreme UltraViolet (EUV) Ar plasmas are the subject of chapter 7. These plasmas, present in lithography machines, can cause sputtering of mirror surfaces within those machines, which is detrimental to the mirror's reflectivity. The sputter rate is largely determined by the electron energy, which can be monitored using Optical Emission Spectroscopy (OES). A time dependent CRM is required to interpret the OES measurements.

One of the ingredients the CRM module requires is the Electron Energy Distribution Function (EEDF). This EEDF can be obtained from a complex and computationally expensive Monte-Carlo model, but we have shown that a simple analytical model gives satisfactory results. The results of the CRM have been compared to experimental results, showing that, as expected, an increased background pressure effectively reduces the electron energy. Although the overall evolution in time of the CRM results show good agreement with the experiments, some discrepancies are found. These can be attributed to the simplicity of the assumptions used for creating the EEDF (no spatial effects), inaccuracies in the cross sections that were used, and experimental concerns.

The second application of the CRM module, is for Laser Induced Fluorescence (LIF) experiments in Ar plasmas in chapter 8. Whereas a relatively simple model was sufficient for the EUV induced Ar plasma, for this plasma a complete Ar system has been constructed, using data from various sources. With the time dependent CRM the effect of a laser pulse on the densities in a plasma can be studied in great detail. It was shown that the saturation densities during the laser pulse can quickly reach an equilibrium, depending on the electron density. This equilibrium is far from the theoretical saturation densities, and requires a CRM to be determined.

The CRM was also used to model the response of levels not directly involved in the LIF process. Comparison with experimental results showed that lack of

accurate cross section data can result in strong discrepancies between experiments and model. However, the CRM is a useful tool to investigate excitation channels in the system.

Chapter 9 describes an application of the GPM module in the form of a well-known case study, which is the implementation of a model by Gudmundsson [88]. It is used to model a High Power Impulse Magnetron Sputtering (HiPIMS) Ar plasma that sputters an Al surface. We demonstrate how chemistry can easily be entered into the model. In this model transport and configuration play an important role, and care must be taken in translating these aspects into frequencies. Once this has been done, a flexible tool is available in which several parameters can easily be adjusted.

The model shows that the Al in the plasma has a very high ionization degree, and that initially electron impact ionization is the main mechanism for creating Al ions, whereas later this becomes charge exchange with Ar ions. These main results are identical to the original model, but some differences are also found, which can be attributed to differences in the rate coefficients that were used.

Design versus research

Although this thesis lying before you is the documentation of a design, it has several research aspects:

- the EUV induced plasma described in chapter 7 is from a research point of view highly unusual in its lack of equilibria;
- the time resolved measurements performed on this plasma are unique;
- the CRM for LIF experiments described in chapter 8 combined with experimental results provides exceptional insights into the dynamics of the Ar system.

In addition, we can state that, although the GPM presented in chapter 9 does not provide important new insights, its flexibility does offer the possibility to obtain results for other sputtered species with relatively little effort. The GPM model also paves the way to many other new research studies that will lead to new technological applications.

Appendices

Appendix A

General C/C++ constructs

In this appendix some important aspects of the C and C++ programming language will be explained. The goal is to give more details about the aspects used in chapter 5. We will do this by discussing listings of small, simple programs. The treatment in this appendix is far from complete and not intended as a language tutorial. Many textbooks are available to learn C/C++ or to be used as reference. Here we will only mention the books by Dennis Ritchie [100] (co-authored by Brian Kernighan), the developer of C, and the book by Bjarne Stroustrup [101], developer of C++. Additionally many tutorials are available online, for instance `cprogramming.com` and `cplusplus.com`.

Since C++ is an enhancement of C, and therefore most constructs in C are also valid in C++*, we will start with some syntaxes in C.

A.1 C: data types and functions

The C programming language is a very widely used general purpose procedural programming language. The language consists of instructions that are executed in sequence, but allows for subroutines to be defined that also consist of a series of computational steps. These subroutines (also called procedures or functions) might be called at any given time at any point during a program's execution. It is also possible for subroutines to be called by other subroutines.

The C programming language has several *built-in* data types, also called *primitive* data types. These are the basic types of variables that can be used to perform

* There are many C syntaxes that are invalid or behave differently in C++. This is however beyond the scope of this text, and only C code that is valid in C++ will be treated.

```
1  double square(double x) {
2    double result = x * x;
3    return result;
4  }
5
6  int main() {
7    double input1 = 2.718;
8    double input2(3.142);
9    double out;
10   out = input1 + square(input2);
11   return 0;
12 }
```

Listing A.1. *Listing of a simple C program equipped with a function to calculate the square.*

calculations. Here, we only mention `int`, the mathematical integer, and `double`, a floating point number of double precision, meaning that the number is represented using 64 bits. Integers can also be defined as `unsigned`, so that negative values are not allowed.

The listing of a simple program is shown in listing A.1. Every C program must implement a "primary" subroutine: the `main()`-function. When the program starts, it starts with the first instruction inside this function. Apart from the `main()`-function, other functions can be defined, and all functions must have the following structure:

- output data type;
- name of the function;
- between parentheses a number of pairs of data type and variable name, separated by commas, representing the input data. When no input is required the list can be empty.

The data types can be any that are known by the program. The output data type can also be `void`, meaning that there is no return value. In listing A.1 two functions are defined: `main()` in line 6, and `square()` in line 1. The `main()` function must always be defined, and this is where the program will start. The `square()` function calculates the square of the input value (a `double`) and then returns this value, which is again a `double`. Note the definition of the return values of both these functions. Functions that have a non-`void` return data type, must end with a `return`-statement (line 3 and 11), where a return value of the appropriate data type is returned by the function*.

* A function can have multiple "exit points", i.e. `return` statements. When one is encountered the execution of the program exits the function and continues where it left of.

In addition to the functions, some variables are declared (lines 2, 7, 8, and 9). A variable is always declared by a data type followed by a variable name*. Note that a variable can only be used after it has been declared, and once it is declared, its type can not change. In line 9 a variable is only defined; it is not given a value, so it is *uninitialized*. In contrast, the variables `input1` and `input2` are declared and initialized, each in a different way. Variable `input1` is initialized using *c-like initialization* and variable `input2` using *constructor initialization*, which will also be used for classes.

The variables within each function can only be used within the function it is defined in. So variable `x` can not be used in the `main()` function, and `input1` can not be used in `square()`. This is called the *scope* of a variable.

What is important to realize, is that in all cases the definition of functions and declaration of variables follows the same syntax. Sometimes the syntax is obscured by complex data types or complex variable names, but the structure is always the same. Whenever parentheses are encountered in code it means that a function is involved, either as a function definition or a function call.

A.2 C++: a simple class

The main feature of C++ is that it allows for the definition of *custom* data types called classes. A class is a construct that combines data storage and functionality. Data is stored in *members*, and functionality is implemented in *methods*, also called *member functions*. Members can be of any known data type, including other classes.

Listing A.2 shows a the declaration and implementation of a simple class. It contains the definition of a class for a rectangle. A rectangle has a width and a height so naturally the class has two members to store these two values (lines 14 and 15). In addition the class has a member function to calculate the surface. Line 10 contains the definition and the name of this member function and its in- and output values including their data type. The class has a second, special member function in line 5; the *constructor*. The constructor is special since it has no return value, and since it is called automatically when a variable of the class type is declared. A variable of a class type is an *instance* of that class or an *object*. The constructor can be recognized from the fact that it has the same name as the class. Since the constructor is automatically called when an object is declared, it is in general used to initialize the members. Members can be initialized in the body of the constructor function (line 8) or in the so called *initializer list* (line 6), which uses the constructor initialization syntax.

*There are certain rules the name of a variable should follow. For example, it can not start with a digit, and it can not contain certain characters like "-", "+", and "{".

```
1  #include <iostream>
2
3  class Rectangle{
4    public:
5      Rectangle(double width, double height)
6       : m_width(width)
7      {
8        m_height = height;
9      }
10     double surface() {
11       return m_width * m_height;
12     }
13   private:
14     double m_width;
15     double m_height;
16 };
17
18 int main() {
19   double width = 2.0;
20   double height(4.0);
21   Rectangle my_rec(width, height);
22
23   double surface = my_rec.surface();
24   std::cout << surface;
25   return 0;
26 }
```

Listing A.2. *Listing of a simple C++ program equipped with a class.*

The `Rectangle` class is divided into two sections: a `public` section and a `private` section, denoted by their respective *access specifiers**. Access specifiers define the accessibility of class members; `public` members are accessible from outside the class, `private` only from members within the class.

In line 21 an *instance* of the class, called an *object*, is created. This uses the same syntax as line 20: constructor initialization. The constructor is called passing the two required initialization values. Whereas a `double` only requires a single initialization value, the `Rectangle` class requires two values, as defined on line 5.

In line 23 the `surface()` member of `my_rec` is called, which can be recognized by the dot ".". This is a function call just like any other, only now the function resides within a class so we need to specify which object contains the desired function.

* There is no limit to the number of sections; the access of a member is defined by the last access specifier that was listed. If no access specifier is listed, members are `private`.

On line 24 the contents of the `surface` variable are sent to output (printed on screen). This line requires some more explanation. The line starts with the object `std::cout`. The actual object is `cout` which is an object representing console output; everything sent to this object is printed to screen. It is defined externally in the file `iostream.h`, a so called *header* file, containing only definitions (no implementation) of classes and functions. We can use this externally defined object since it is included in line 1. The prefix `std::` means that the `cout` object is available in a so called *namespace*. A namespace is an environment within which identifiers (variables, functions) can be defined and subsequently used. In this way multiple identifiers with the same name can be defined as long as they reside in different namespaces. They can only be used within a namespace or in combination with the appropriate namespace, requiring a double colon "`::`" (as demonstrated by `std::cout`).

The second element of line 24 is the "`<<`"-operator*, which is here used to send the variable `surface` to the `cout` object. This demonstrates an important feature of C++ called *operator overloading*.

A.3 Operator overloading

As previously stated, classes are types just like the built-in types. This means that they can be used in combination with operators, but the result of an operator acting on a class† must be defined explicitly. An example is shown in listing A.3, where the result of adding two `Rectangle` objects is that the width of the resulting `Rectangle` is the sum of the widths of the original `Rectangle`s‡.

The `Rectangle` class is extended with the operator on line 13. Note that only the fact that this operator is available for this class is defined. The actual implementation is located outside the class on lines 19 and 20. This separation of *interface* (the definition of a class by its members and member functions by their names, input types, and output types) and *implementation* can be done for all member functions. It is a feature that is widely used in C++, and in practice the two parts are usually placed in separate files: a header file (for the interface) and a source file (for the implementation).

The interface (line 13) shows that the `+=` operator requires one `Rectangle` object as input (the right-hand side of the operator) and returns a `Rectangle` object (the left-hand side). This format is also present on line 19, but the namespace `Rectangle` is added since it concerns a member of that class. The body of the im-

* The standard use of the "`<<`"-operator is *shift left*, i.e., a bitwise shifting to the left of an integer variable. A shift by one bit equals a multiplication by 2.

† Nearly all operators can be overloaded, for instance "`+`", "`--`", "`&&`", and "`()`".

‡ Not the "`+`" operator but the "`+=`" operator is implemented. So instead of a new `Rectangle` object being created, the width of the right-hand side `Rectangle` object is added to the width of the left-hand side `Rectangle` object.

```
#include <iostream>

class Rectangle{
  public:
    Rectangle(double width, double height)
     : m_width(width)
    {
      m_height = height;
    }
    double surface() {
      return m_width * m_height;
    }
    Rectangle operator+=(Rectangle& other);
  private:
    double m_width;
    double m_height;
};

Rectangle Rectangle::operator+=(Rectangle& other) {
  m_width += other.width;
}

int main() {
  double width = 2.0;
  double width2 = 3.0;
  double height(4.0);
  Rectangle my_rec(width, height);
  Rectangle my_rec2(width2, height);

  my_rec += my_rec2;
  std::cout << my_rec.surface();
  return 0;
}
```

Listing A.3. *Listing of a simple C++ program demonstrating operator overloading. The code is an extension of the code in listing A.2.*

```
1  #include <iostream>
2
3  int main() {
4    int i = 2.0;
5    int* p_i;
6    p_i = &i;
7    std::cout << *p_i;
8    int& r_i = i;
9    std::cout << r_i;
10   return 0;
11 }
```

Listing A.4. *Listing of a simple C++ program demonstrating the syntax of pointers and references.*

plementation on line 20 shows that the width of the `Rectangle` is increased with the width of the `Rectangle` given as argument.

On line 30 the `+=` operator is used so that the new width of `my_rec` will be $2.0 + 3.0 = 5.0$.

In the definition of the argument list of the operator on lines 13 and 19 there is an extra "`&`" character denoting that the argument is passed as a *reference*.

A.4 Pointers and references

A *pointer* is a variable that holds an address (usually the address of another variable) and a reference is a datatype that only refers to another variable, i.e., it is an alternative name for a variable. The difference is that a pointer itself represents data (the address) and a reference does not, meaning that a pointer can be declared without initialization (it points to an undefined address in memory) and a reference can only refer to an existing variable.

In listing A.4 the syntax and use of pointers and references is shown. On line 5 a pointer is defined (note the "`*`"), but it is not initialized, meaning that its contents (the address) is not defined and it can point to any value in memory. In the next line (line 6) the pointer is given the address of variable `i` (note the "`&`"). Line 7 shows how the pointer can be used to access variable `i`. By prefixing the pointer variable with "`*`" the pointer is *dereferenced*, meaning that the value at the address the pointer points to is read from memory.

On line 8 a reference to variable `i` is defined (note the "`&`"), and on line 9 the reference is used to print the value of `i` to screen. It shows that reference `r_i` can be used just like a normal variable, it is only a different name.

Pointers and references are often used in function calls. When a variable is passed to a function as argument, the whole object is copied. This is called *pass by*

```
1  template <class T>
2  T square(T x) {
3    T result = x * x;
4    return result;
5  }
6
7  int main() {
8    double input1 = 2.718;
9    double input2(3.142);
10   double out;
11   out = input1 + square<double>(input2);
12   return 0;
13 }
```

Listing A.5. *Listing of a reimplementation of listing A.1 with a template function to calculate the square.*

value. For small variables like the built-in types this is not a problem, but for large classes this can be a large amount of data, meaning that the program is slower and requires more memory. Another option is called *pass by reference*, meaning that the function is only given a reference to the argument variable. This is demonstrated in listing A.3 line 19. Since only a reference to the original data has to be passed to the function it is a lot faster and requires less memory. Because the function now has access to the original data (instead of to a copy of the data), the function can alter this data, meaning that this effectively acts as a return mechanism. In contrast to the standard return mechanism it is not limited to a single variable, since a function can have multiple arguments, each passed as reference.

A similar functionality can be reached using pointers. If a pointer is passed as argument to a function, the pointer is copied (pass by value), but through the pointer the original data can be accessed.

A.5 Templates

The `square()`-function in listing A.1 only works with variables of type `double`. It would be convenient to have a general `square()`-function that we can use with any type of variable. This can be accomplished with *templates*.

Listing A.5 show how the `square()`-function is implemented as a function template. The actual function is preceded with a template declaration in line 1. Between angled brackets the template parameter `T` is given, which represents any type, it does not have to be defined. In the rest of the function the type `double` is replaced with the type `T`. Only when the function is called, do we need to tell the function which type we are dealing with, line 11, again using angled brackets.

```
1  template <class T>
2  class Container {
3    public:
4      T m_value;
5  };
6
7  int main() {
8    double input = 2.718;
9    Container<double> my_contain;
10   my_contain.m_value = input;
11   return 0;
12 }
```

Listing A.6. *Listing of a simple program using a template class.*

This single function can now be used with any type. The only thing that is required is that the type must support a multiplication with a variable of the same type, required on line 3*. Any type means that we can also use it with classes. For instance, we could define a class for complex numbers containing a real and imaginary value. What a multiplication does to this kind of variable must be explicitly defined in the class.

Apart from functions, we can also use templates in classes: template classes. This is demonstrated in listing A.6.

As with the function template, the class is preceded by a `template` declaration, naming the random variable type `T`. The class only contains a member variable of this type `T` (line 4). On line 9 an object of this `Container<>` type is created, where the desired type (`double`) is passed between angled brackets. The `my_contain` variable can now store a `double` variable which is done in line 10.

Templates allow general functions and classes to be defined. Only when an actual call to such a function or an object of such a class is required does the type have to be defined. This offers more flexibility and modularity.

A.6 Frequently used operators

We conclude this appendix with a number of frequently used operators. Because in C++ intricate combinations of data types can be used the syntax is often obscured by long, complex variable types and names. The key in understanding a piece of C++ code is to recognize the different elements: variable type, variable name, operator. In table A.1 the most commonly used operators are listed together with their meaning. The table can be used as a legend when reading C++.

* This particular implementation also requires that the assignment operator (`operator=`) is defined.

operator	meaning
`{ }`	section of code
`( )`	(member) function definition or call, or variable declaration and initialization
`.`	class member (function)
`:`	parent class(es) for derived class, or initializer list
`::`	namespace (of class)
`*`	pointer (or multiplication)
`&`	reference or obtain address
`< >`	template

Table A.1. *Frequently used C++ operators and their meaning.*

Appendix B

Racah & Paschen notation

The following two tables can be used to translate Paschen notation into Racah notation and vice versa.

Orbital	Block			
s	4s	5s	6s	7s
	1s	*2s*	*3s*	*4s*
p	4p	5p	6p	7p
	2p	*3p*	*4p*	*5p*
d	3d	4d	5d	6d
	3d	*4d*	*5d*	*6d*

Table B.1. *The blocks in Racah notation (top row) and the corresponding blocks in Paschen notation (bottom row). Note that the four primed d levels in Racah notation are denoted as s in Paschen notation.*

Orbital	Levels					
s	$[3/2]_2$	$[3/2]_1$	$'[1/2]_0$	$'[1/2]_1$		
	s_5	s_4	s_3	s_2		
p	$[1/2]_1$	$[5/2]_3$	$[5/2]_2$	$[3/2]_1$	$[3/2]_2$	
	p_{10}	p_9	p_8	p_7	p_6	
	$[1/2]_0$	$'[3/2]_1$	$'[3/2]_2$	$'[1/2]_1$	$'[1/2]_0$	
	p_5	p_4	p_3	p_2	p_1	
d	$[1/2]_0$	$[1/2]_1$	$[7/2]_4$	$[7/2]_3$	$[3/2]_2$	$[3/2]_1$
	d_6	d_5	d'_4	d_4	d_3	d_2
	$[5/2]_2$	$[5/2]_3$	$'[5/2]_2$	$'[5/2]_3$	$'[3/2]_2$	$'[3/2]_1$
	d''_1	d'_1	s''''_1	s'''_1	s''_1	s'_1
f	$[7/2]_{3,4}$	$[9/2]_{4,5}$	$'[7/2]_{3,4}$	$[3/2]_{1,2}$	$[5/2]_{2,3}$	$'[5/2]_{2,3}$
	U	V	W	X	Y	Z

Table B.2. *The orbitals (Racah notation) and the levels they contain. The top line shows the Racah notation and the bottom line the corresponding Paschen notation (in italics). The prime in the Racah notation denotes the configuration of the core:* $\mathrm{P}_{3/2}$ *for unprimed,* $\mathrm{P}_{1/2}$ *for primed.*

Appendix C

Ar CRM data

CRM model data for Ar model of chapter 8.

Table C.1. *All levels included in the Ar model of chapter 8, their energy, and statistical weight.*

name	energy	weight	name	energy	weight
ground	0.000000	1			
$4s[3/2]_2$	11.548354	5	$4s[3/2]_1$	11.623592	3
$4s'[1/2]_0$	11.723160	1	$4s'[1/2]_1$	11.828071	3
$4p[1/2]_1$	12.907015	3	$4p[5/2]_3$	13.075715	7
$4p[5/2]_2$	13.094872	5	$4p[3/2]_1$	13.153143	5
$4p[3/2]_2$	13.171777	3	$4p[1/2]_0$	13.273038	1
$4p'[3/2]_1$	13.282638	3	$4p'[3/2]_2$	13.302227	5
$4p'[1/2]_1$	13.327856	3	$4p'[1/2]_0$	13.479886	1
$3d[1/2]_0$	13.845038	1	$3d[1/2]_1$	13.863668	3
$3d[3/2]_2$	13.903454	5	$3d[7/2]_4$	13.979237	9
$3d[7/2]_3$	14.012738	7	$3d[5/2]_2$	14.063027	5
$3d[5/2]_3$	14.099055	7	$3d[3/2]_1$	14.152514	3
$3d'[5/2]_2$	14.213671	5	$3d'[3/2]_2$	14.234022	5

name	energy	weight	name	energy	weight
$3d'[5/2]_3$	14.236105	7	$3d'[3/2]_1$	14.303668	3
$5s[3/2]_2$	14.068297	5	$5s[3/2]_1$	14.089968	3
$5s'[1/2]_0$	14.241027	1	$5s'[1/2]_1$	14.255085	3
$5p[1/2]_1$	14.463995	3	$5p[5/2]_3$	14.499053	7
$5p[5/2]_2$	14.506067	5	$5p[3/2]_1$	14.524913	3
$5p[3/2]_2$	14.528913	5	$5p[1/2]_0$	14.575948	1
$5p'[3/2]_1$	14.680650	3	$5p'[1/2]_1$	14.687118	3
$5p'[3/2]_2$	14.688290	5	$5p'[1/2]_0$	14.738115	1
6s	14.842395	8	6s′	15.020082	4
7s	15.182461	8	7s′	15.359150	4
8s	15.363559	8	8s′	15.541244	4
9s	15.470217	8	9s′	15.647963	4
10s	15.539020	8	10s′	15.716050	4
6p	15.027492	24	6p′	15.204578	12
7p	15.285607	24	7p′	15.441823	12
8p	15.418657	24	8p′	15.595913	12
9p	15.505837	24	9p′	15.687000	12
4d	14.780255	40	4d′	14.967431	20
5d	15.146943	40	5d′	15.316640	20
6d	15.343672	40	6d′	15.515428	20
7d	15.454865	40	7d′	15.632548	20
8d	15.530544	40	8d′	15.713471	20
4f	14.905574	56	4f′	15.083141	28
5f	15.213270	56	5f′	15.390949	28
6f	15.380265	56	6f′	15.557897	28
7f	15.481085	56	7f′	15.658571	28
ion	15.75961	6			

Table C.2. *All radiative transitions included in the Ar CRM of chapter 8, their wavelength and transition probability.*

upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$	upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$
4s[3/2]1	ground	106.7	119	4s′[1/2]1	ground	104.8	510
4p[1/2]1	4s′[1/2]0	1047.3	0.98	4p[1/2]1	4s′[1/2]1	1149.1	0.19
4p[1/2]1	4s[3/2]1	966.0	5.43	4p[1/2]1	4s[3/2]2	912.5	18.9
4p[5/2]3	4s[3/2]2	811.8	33.1	4p[5/2]2	4s′[1/2]1	978.7	1.47
4p[5/2]2	4s[3/2]1	842.7	21.5	4p[5/2]2	4s[3/2]2	801.7	9.28
4p[3/2]1	4s′[1/2]1	935.7	5.03	4p[3/2]1	4s[3/2]1	810.6	4.9
4p[3/2]1	4s[3/2]2	772.6	24.5	4p[3/2]1	4s′[1/2]0	867.0	2.43
4p[3/2]2	4s′[1/2]1	922.7	1.06	4p[3/2]2	4s[3/2]1	800.8	25.0
4p[3/2]2	4s[3/2]2	763.7	5.18	4p[1/2]0	4s[3/2]1	751.7	40.2
4p′[3/2]1	4s′[1/2]0	795.0	18.6	4p′[3/2]1	4s′[1/2]1	852.4	13.9
4p′[3/2]1	4s[3/2]1	747.3	0.022	4p′[3/2]1	4s[3/2]2	714.9	0.625
4p′[3/2]2	4s′[1/2]1	841.1	22.3	4p′[3/2]2	4s[3/2]1	738.6	8.47
4p′[3/2]2	4s[3/2]2	706.9	3.8	4p′[1/2]1	4s′[1/2]0	772.6	11.7
4p′[1/2]1	4s′[1/2]1	826.7	15.3	4p′[1/2]1	4s[3/2]1	727.5	1.83
4p′[1/2]1	4s[3/2]2	696.7	6.39	4p′[1/2]0	4s′[1/2]1	750.6	44.5
4p′[1/2]0	4s[3/2]1	667.9	0.236				
3d[1/2]0	4p[1/2]1	1321.8	8.1	3d[1/2]0	4p′[1/2]1	2397.3	0.36
3d[1/2]0	4p′[3/2]1	2204.6	0.12	3d[1/2]1	4p[1/2]1	1296.0	7.4
3d[1/2]1	4p′[1/2]1	2314.0	0.17	3d[1/2]1	4p′[3/2]1	2133.9	0.032
3d[1/2]1	4p′[3/2]2	2208.3	0.14	3d[1/2]1	4p[5/2]2	1612.7	0.039
3d[3/2]2	4p[1/2]1	1244.3	4.9	3d[3/2]2	4p′[1/2]1	2154.0	0.11
3d[3/2]2	4p[3/2]1	1694.5	0.26	3d[3/2]2	4p[3/2]2	1652.4	2.5
3d[3/2]2	4p′[3/2]2	2062.2	0.39	3d[3/2]2	4p[5/2]2	1533.4	0.12
3d[7/2]3	4p[3/2]2	1442.4	0.088	3d[7/2]3	4p[5/2]2	1350.8	11
3d[5/2]2	4p[3/2]1	1391.1	7.3	3d[5/2]2	4p[5/2]2	1280.6	5.7
3d[5/2]2	4p[5/2]3	1255.8	0.12	3d[5/2]3	4p′[3/2]2	1556.0	0.0098
3d[5/2]3	4p[5/2]2	1234.7	2	3d[5/2]3	4p[5/2]3	1211.6	3.1
3d[3/2]1	4p[1/2]0	1409.7	4.3	3d[3/2]1	4p[3/2]1	1264.2	11
3d[3/2]1	4p[5/2]2	1172.3	0.952	3d[3/2]1	ground	87.6	270

upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$	upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$
3d′[5/2]2	4p′[3/2]1	1331.7	13	3d′[5/2]2	4p′[3/2]2	1360.3	2.2
3d′[5/2]2	4p[5/2]2	1108.2	0.83	3d′[3/2]2	4p′[1/2]1	1368.2	6.2
3d′[3/2]2	4p[3/2]1	1167.2	0.369	3d′[3/2]2	4p[3/2]2	1147.1	3.76
3d′[5/2]3	4p′[3/2]2	1327.6	15	3d′[3/2]1	4p[1/2]0	1203.0	0.42
3d′[3/2]1	4p′[1/2]0	1505.1	5.2	3d′[3/2]1	4p′[1/2]1	1270.6	7.1
3d′[3/2]1	4p′[3/2]1	1214.3	4.5	3d′[3/2]1	4p[3/2]2	1077.6	0.396
3d′[3/2]1	ground	86.7	313				
5s[3/2]2	4p[1/2]1	1067.6	4.9	5s[3/2]2	4p′[1/2]1	1674.5	0.31
5s[3/2]2	4p[3/2]1	1382.9	0.46	5s[3/2]2	4p′[3/2]1	1578.1	0.059
5s[3/2]2	4p[3/2]2	1354.8	3.3	5s[3/2]2	4p′[3/2]2	1618.4	0.12
5s[3/2]2	4p[5/2]2	1273.7	1.1	5s[3/2]2	4p[5/2]3	1249.1	11
5s[3/2]1	4p[1/2]0	1517.7	1.3	5s[3/2]1	4p′[1/2]0	2032.3	0.16
5s[3/2]1	4p[1/2]1	1048.1	2.44	5s[3/2]1	4p′[1/2]1	1626.9	0.03
5s[3/2]1	4p[3/2]1	1350.3	4.6	5s[3/2]1	4p′[3/2]1	1535.7	0.45
5s[3/2]1	4p[3/2]2	1323.5	2.7	5s[3/2]1	4p′[3/2]2	1573.9	0.029
5s[3/2]1	4p[5/2]2	1246.0	8.9	5s[3/2]1	ground	88.0	77
5s′[1/2]0	4p′[1/2]1	1357.7	5.1	5s′[1/2]0	4p[1/2]1	929.4	3.26
5s′[1/2]0	4p[3/2]1	1159.5	2.22	5s′[1/2]0	4p′[3/2]1	1293.7	10
5s′[1/2]1	4p[1/2]0	1262.5	0.38	5s′[1/2]1	4p′[1/2]0	1599.4	1.9
5s′[1/2]1	4p′[1/2]1	1337.1	3.4	5s′[1/2]1	4p[1/2]1	919.7	1.76
5s′[1/2]1	4p[3/2]1	1144.5	0.28	5s′[1/2]1	4p′[3/2]1	1275.0	2.0
5s′[1/2]1	4p[3/2]2	1125.1	1.39	5s′[1/2]1	4p′[3/2]2	1301.2	8.9
5s′[1/2]1	4p[5/2]2	1068.6	0.21	5s′[1/2]1	ground	87.0	35
5p[1/2]1	4s′[1/2]0	452.4	0.0898	5p[1/2]1	4s′[1/2]1	470.4	0.109
5p[1/2]1	4s[3/2]1	436.5	0.012	5p[1/2]1	4s[3/2]2	425.2	0.111
5p[5/2]3	3d[3/2]2	2081.7	0.076	5p[5/2]3	3d[7/2]4	2385.2	1.1
5p[5/2]3	4s[3/2]2	420.2	0.967	5p[5/2]2	4s′[1/2]1	463.0	0.0383
5p[5/2]2	4s[3/2]1	430.1	0.377	5p[5/2]2	4s[3/2]2	419.2	0.28
5p[3/2]1	4s′[1/2]0	442.5	0.0073	5p[3/2]1	4s′[1/2]1	459.7	0.0947
5p[3/2]1	4s[3/2]1	427.3	0.797	5p[3/2]1	4s[3/2]2	416.5	0.288
5p[3/2]2	4s′[1/2]1	459.1	0.0062	5p[3/2]2	4s[3/2]1	426.7	0.312
5p[3/2]2	4s[3/2]2	416.0	1.4	5p[1/2]0	4s′[1/2]1	451.2	1.18
5p[1/2]0	4s[3/2]1	419.9	2.57	5p′[3/2]1	4s′[1/2]0	419.2	0.539

upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$	upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$
5p′[3/2]1	4s′[1/2]1	434.6	0.297	5p′[3/2]1	4s[3/2]1	405.6	0.027
5p′[1/2]1	3d[3/2]2	1582.1	0.087	5p′[1/2]1	4s′[1/2]0	418.3	0.561
5p′[1/2]1	4s′[1/2]1	433.7	0.387	5p′[1/2]1	4s[3/2]1	404.7	0.041
5p′[1/2]1	4s[3/2]2	395.0	0.455	5p′[3/2]2	4s′[1/2]1	433.5	0.568
5p′[3/2]2	4s[3/2]1	404.6	0.333	5p′[3/2]2	4s[3/2]2	394.9	0.056
5p′[1/2]0	4s′[1/2]1	426.1	3.98				
4d	4p[1/2]0	822.6	0.069	4d	4p[1/2]1	661.9	0.53
4d	4p′[1/2]1	853.7	0.16	4d	4p[3/2]1	770.8	0.094
4d	4p′[3/2]1	827.9	0.018	4d	4p[3/2]2	762.0	0.19
4d	4p′[3/2]2	838.8	0.21	4d	4p[5/2]2	735.6	0.20
4d	4p[5/2]3	727.4	0.48				
6s	4p[1/2]0	790.0	0.13	6s	4p′[1/2]0	910.0	0.036
6s	4p[1/2]1	640.6	0.88	6s	4p′[1/2]1	818.6	0.11
6s	4p[3/2]1	742.1	0.77	6s	4p′[3/2]1	794.9	0.066
6s	4p[3/2]2	734.0	0.83	6s	4p′[3/2]2	805.0	0.089
6s	4p[5/2]2	709.5	1.03	6s	4p[5/2]3	701.8	1.67
6s′	4p[1/2]0	709.7	0.11	6s′	4p′[1/2]0	805.0	0.27
6s′	4p[1/2]1	586.7	0.52	6s′	4p′[1/2]1	732.7	1.02
6s′	4p[3/2]1	670.8	0.23	6s′	4p′[3/2]1	713.6	0.98
6s′	4p[3/2]2	664.1	0.12	6s′	4p′[3/2]2	721.7	1.86
6s′	4p[5/2]2	644.0	0.038				
7s	4p[1/2]0	649.3	0.035	7s	4p[1/2]1	544.9	0.37
7s	4p[3/2]1	616.6	0.22	7s	4p′[3/2]1	652.6	0.020
7s	4p[3/2]2	611.0	0.50	7s	4p′[3/2]2	659.4	0.023
7s	4p[5/2]2	593.9	0.53	7s	4p[5/2]3	588.5	0.81
7s′	4p[1/2]0	594.3	0.09	7s′	4p[1/2]1	505.6	0.48
7s′	4p′[1/2]1	610.4	0.33	7s′	4p[3/2]1	566.8	0.25
7s′	4p′[3/2]1	597.1	0.41	7s′	4p′[3/2]2	602.8	0.68
7s′	4p[5/2]2	547.6	0.15				
8s	4p[1/2]1	504.7	0.29	8s	4p′[1/2]1	609.0	0.028
8s	4p′[3/2]1	595.8	0.056	8s	4p[3/2]2	560.9	0.21

upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$	upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$
8s	4p′[3/2]2	601.5	0.052	8s	4p[5/2]2	546.5	0.18
8s	4p[5/2]3	541.9	0.38	8s′	4p[3/2]1	523.3	0.20
8s′	4p′[3/2]1	548.9	0.14	8s′	4p′[3/2]2	553.7	0.20
9s	4p[1/2]0	564.3	0.079	9s	4p[1/2]1	483.7	0.064
9s	4p[3/2]2	535.1	0.060	9s	4p[5/2]2	522.0	0.049
9s	4p[5/2]3	517.8	0.15				
10s	4p[5/2]2	507.3	0.098	10s	4p[5/2]3	503.3	0.051
6p	4s′[1/2]0	375.2	0.0088	6p	4s′[1/2]1	387.5	0.038
6p	4s[3/2]1	364.2	0.072	6p	4s[3/2]2	356.4	0.088
6p′	4s′[1/2]0	356.1	0.030	6p′	4s′[1/2]1	367.2	0.092
6p′	4s[3/2]1	346.2	0.028				
7p	4s′[1/2]1	358.6	0.021	7p′	4s′[1/2]1	343.1	0.033
4d′	4p[1/2]0	731.7	0.0087	4d′	4p[1/2]1	601.7	0.31
4d′	4p′[1/2]1	756.2	0.15	4d′	4p[3/2]1	690.5	0.11
4d′	4p′[3/2]1	735.9	0.11	4d′	4p[3/2]2	683.4	0.22
4d′	4p′[3/2]2	744.6	0.24	4d′	4p[5/2]2	662.1	0.047
4d′	4p[5/2]3	655.4	0.11				
5d	4p[1/2]0	661.6	0.029	5d	4p[1/2]1	553.5	0.42
5d	4p′[1/2]1	681.6	0.038	5d	4p[3/2]1	627.7	0.11
5d	4p′[3/2]1	665.0	0.026	5d	4p[3/2]2	621.8	0.18
5d	4p′[3/2]2	672.1	0.069	5d	4p[5/2]2	604.2	0.29
5d	4p[5/2]3	598.6	0.60	5d′	4p[1/2]0	606.7	0.012
5d′	4p[1/2]1	514.5	0.37	5d′	4p′[1/2]1	623.4	0.23
5d′	4p[3/2]1	578.1	0.24	5d′	4p′[3/2]1	609.6	0.30
5d′	4p[3/2]2	573.1	0.21	5d′	4p′[3/2]2	615.5	0.43
5d′	4p[5/2]2	558.0	0.27	5d′	4p[5/2]3	553.3	0.084
6d	4p[1/2]0	598.8	0.023	6d	4p′[1/2]0	665.2	0.0091
6d	4p[1/2]1	508.8	0.21	6d	4p′[1/2]1	615.1	0.016

upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$	upper level	lower level	λ [nm]	A $[10^6 s^{-1}]$
6d	4p[3/2]1	570.9	0.023	6d	4p′[3/2]1	601.6	0.015
6d	4p[3/2]2	566.0	0.19	6d	4p′[3/2]2	607.3	0.015
6d	4p[5/2]2	551.3	0.074	6d	4p[5/2]3	546.7	0.40
6d′	4p[1/2]1	475.3	0.22	6d′	4p[3/2]1	529.0	0.090
6d′	4p′[3/2]1	555.3	0.067	6d′	4p[3/2]2	524.8	0.064
6d′	4p′[3/2]2	560.2	0.18	6d′	4p[5/2]2	512.2	0.10
6d′	4p[5/2]3	508.2	0.016				
7d	4p[1/2]0	568.3	0.013	7d	4p[1/2]1	486.6	0.24
7d	4p′[1/2]1	582.9	0.0041	7d	4p[3/2]1	543.1	0.034
7d	4p[3/2]2	538.7	0.035	7d	4p′[3/2]2	576.0	0.016
7d	4p[5/2]2	525.4	0.13	7d	4p[5/2]3	521.1	0.22
7d′	4p[1/2]1	454.9	0.0095	7d′	4p′[3/2]2	532.0	0.091
8d	4p[1/2]0	549.2	0.0090	8d	4p[1/2]1	472.6	0.043
8d	4p′[3/2]1	551.6	0.0059	8d	4p[3/2]2	521.5	0.038
8d	4p[5/2]2	509.0	0.028	8d	4p[5/2]3	505.1	0.083
4f	3d[1/2]0	1169.1	0.036	4f	3d[1/2]1	1190.0	0.66
4f	3d[3/2]1	1646.4	0.76	4f	3d[3/2]2	1237.2	0.75
4f	3d′[5/2]2	1791.9	0.11	4f	3d[5/2]3	1537.3	1.77
4f	3d[7/2]4	1338.4	2.3	4f	5s[3/2]2	1480.8	0.053
4f′	3d′[3/2]1	1590.6	1.38	4f′	3d[3/2]2	1051.0	0.68
4f′	3d′[3/2]2	1460.2	2.3	4f′	3d′[5/2]3	1463.7	5.1
4f′	3d[7/2]3	1158.3	0.20				

Bibliography

[1] W. Baumjohann and R.A. Treumann. *Basic space plasma physics*. Imperial College Press, 1997.

[2] G.J.M. Hagelaar. *Modeling of Microdischarges for Display Technology*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2000.

[3] H.W.P. van der Heijden. *Modelling of Radiative Transfer in Light Sources*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2003.

[4] C.W. Johnston. *Transport and Equilibrium in Molecular Plasmas: the sulfur lamp*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2003.

[5] V. Banine and R. Moors. Plasma sources for euv lithography exposure tools. *Journal of Physics D: Applied Physics*, 37(23):3207, 2004.

[6] R.S. Tipa and G.M.W. Kroesen. Plasma-stimulated wound healing. *Plasma Science, IEEE Transactions on*, 39(11):2978 –2979, nov. 2011.

[7] G.J.J. Winands, Keping Yan, A.J.M. Pemen, S.A. Nair, Zhen Liu, and E.J.M. van Heesch. An industrial streamer corona plasma system for gas cleaning. *Plasma Science, IEEE Transactions on*, 34(5):2426 –2433, oct. 2006.

[8] D.R. Bates, A.E. Kingston, and R.W.P. McWhirter. Recombination between electrons and atomic ions, I. Optically thin plasmas. *Proc. R. Soc.*, A267:297, 1962.

[9] J. van Dijk, A. Hartgers, J. Jonkers, and J.J.A.M. van der Mullen. Collisional radiative models with multiple transport-sensitive levels - application to high electron density mercury discharges. *Journal of Physics D: Applied Physics*, 34(10):1499, 2001.

[10] A. Flitti and S. Pancheshnyi. Gas heating in fast pulsed discharges in N_2-O_2 mixtures. *The European Physical Journal Applied Physics*, 45:21001, 2009.

[11] J.J. Munro and J. Tennyson. Global plasma simulations using dynamically generated chemical models. *Journal of Vacuum Science and Technology A: Vacuum, Surfaces, and Films*, 26(4):865–869, 2008.

[12] V. Guerra and J. Loureiro. Non-equilibrium coupled kinetics in stationary N_2-O_2 discharges. *Journal of Physics D: Applied Physics*, 28(9):1903, 1995.

[13] M. Jiménez Díaz. *Modelling of Microwave Induced Plasmas. The interplay between electromagnetism, plasma chemistry and transport*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2011.

[14] M.J. van den Donker. *Modelling microwave plasmas for deposition purposes; exploring the freedom in space and chemistry*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2008.

[15] T.H. Chung, H.J. Yoon, and D.C. Seo. Global model and scaling laws for inductively coupled oxygen discharge plasmas. *Journal Applied Physics*, 86(7):3536–3542, 1999.

[16] S. Ashida, C. Lee, and M.A. Lieberman. Spatially averaged (global) model of time modulated high density argon plasmas. *American Vacuum Society*, 13(5):2498, 1995.

[17] S. Kim. *An improved global model for electronegative discharge and ignition conditions for peripheral plasma connected to a capacitive discharge*. PhD thesis, University of California, Berkeley, 2006.

[18] B.H.P. Broks and J.J.A.M. van der Mullen. Creating a global plasma model using disturbed bilateral relations. *Journal of Physics: Conference Series*, 44(1):53, 2006.

[19] R.D. Present. *Kinetic theory of gases*. Number v. 222 in International series in pure and applied physics. McGraw-Hill, 1958.

[20] M.A. Lieberman and A.J. Lichtenberg. *Principles of plasma discharges and materials processing*. Wiley-Interscience, 2005.

[21] V.A. Godyak. *Soviet radio frequency discharge research*. Monograph series on Soviet Union. Delphic Associates, 1986.

[22] J.J.A.M. van der Mullen. *Excitation equilibria in plasmas; a classification*. PhD thesis, Eindhoven University of Technology, The Netherlands, 1986.

[23] Takashi Fujimoto. Kinetics of ionization-recombination of a plasma and population density of excited ions. IV. Recombining plasma—low-temperature case. *Journal of the Physical Society of Japan*, 49(4):1569–1576, 1980.

[24] J. van Dijk, A. Hartgers, J. Jonkers, and J.J.A.M. van der Mullen. A collisional radiative model for mercury in high-current discharges. *Journal of Physics D: Applied Physics*, 33(21):2798, 2000.

[25] J.J.A.M. and van der Mullen. Excitation equilibria in plasmas; a classification. *Physics Reports*, 191(2-3):109–220, 1990.

[26] E.R. Kieft. *Transient behavior of EUV emitting plasmas, a study by optical methods*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2005.

[27] J. van Dijk. *Modelling of Plasma Light Sources — an object-oriented approach*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2001.

[28] C.W. Johnston, H.W.P. van der Heijden, G.M. Janssen, J. van Dijk, and J.J.A.M. van der Mullen. A self-consistent LTE model of a microwave-driven, high-pressure sulfur lamp. *Journal of Physics D: Applied Physics*, 35(4):342, 2002.

[29] H.W.P. van der Heijden and J.J.A.M. van der Mullen. General treatment of the interplay between fluid and radiative transport phenomena in symmetric plasmas: the sulphur lamp as a case study. *Journal of Physics D: Applied Physics*, 35(17):2112, 2002.

[30] L. Vriens and A.H.M. Smeets. Cross-section and rate formulas for electron-impact ionization, excitation, deexcitation, and total depopulation of excited atoms. *Phys. Rev. A*, 22:940–951, Sep 1980.

[31] D.A. Benoy. *Modelling of Thermal Argon Plasmas*. PhD thesis, Eindhoven University of Technology, The Netherlands, 1993.

[32] L. C. Johnson. Approximations for Collisional and Radiative Transition Rates in Atomic Hydrogen. *The Astrophysical Journal*, 174:227, May 1972.

[33] D.A. Benoy, J.J.A.M. van der Mullen, B. Van Der Sijde, and D.C. Schram. A novel collisional radiative model with a numerical bottom and an analytical top. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 46(3):195 – 210, 1991.

[34] H.W.P. van der Heijden, M.G.H. Boogaarts, S. Mazouffre, J.J.A.M. van der Mullen, and D.C. Schram. Time-resolved experimental and computational study of two-photon laser-induced fluorescence in a hydrogen plasma. *Phys. Rev. E*, 61:4402–4409, Apr 2000.

[35] H.W. Drawin. Collision and transport cross-sections. Technical Report EUR-CEA-FC-383, Association Euratom-C.E.A., 1967.

[36] Yong-Ki Kim and Mitio Inokuti. Generalized oscillator strengths of the helium atom. I. *Phys. Rev.*, 175:176–188, Nov 1968.

[37] A. Kimura, H. Kobayashi, M. Nishida, and P. Valentin. Calculation of collisional and radiative transition probabilities between excited argon levels. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 34(2):189 – 215, 1985.

[38] J. Vlček. A collisional-radiative model applicable to argon discharges over a wide range of conditions. I. Formulation and basic data. *Journal of Physics D: Applied Physics*, 22(5):623–631, 1989.

[39] Fumihiko Kannari, Minoru Obara, and Tomoo Fujioka. An advanced kinetic model of electron-beam-excited KrF lasers including the vibrational relaxation in $KrF^*(B)$ and collisional mixing of $KrF^*(B,C)$. *Journal of Applied Physics*, 57(9):4309–4322, 1985.

[40] R.C. Tolman. *The principles of statistical mechanics*. Dover books on physics and chemistry. Dover Publications, 1938.

[41] T. Holstein. Imprisonment of resonance radiation in gases. *Phys. Rev.*, 72:1212–1233, Dec 1947.

[42] A. Unsöld. Quantitative spektralanalyse der sonnenatmosphäre. *Z. Astrophys.*, 24:355, 1948.

[43] G. Ecker and W. Weizel. Zustandssumme und effektive ionisierungsspannung eines atoms im inneren des plasmas. *Annalen der Physik*, 452(2-3):126–140, 1956.

[44] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions*. National Bureau of Standards, tenth edition, 1964.

[45] B.F.M. Pots, B. van der Sijde, and D.C. Schram. A collisional radiative model for the argon ion (laser) system with an experimental test. *Physica B+C*, 94(3):369 – 393, 1978.

[46] A. Hartgers, J. van Dijk, J. Jonkers, and J.J.A.M. van der Mullen. CRModel: A general collisional radiative modeling code. *Computer Physics Communications*, 135(2):199 – 218, 2001.

[47] G.M. Janssen. *Design of a General Plasma Simulation Model, Fundamental Aspects and Applications*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2000.

[48] A. Hartgers. *Modelling of a Fluorescent Lamp Plasma*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2003.

[49] B.H.P. Broks. *Multi-fluid Modeling of Transient Plasmas*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2006.

[50] M.L. Beks. *Modelling additive transport in metal halide lamps*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2008.

[51] W.J.M. Brok. *Modelling of Transient Phenomena in Gas Discharges*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2005.

[52] D. Mihailova. *Sputtering Hollow Cathode Discharges designed for Laser Applications. Experiments and Theory*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2010.

[53] M.H.L. van der Velden. *Radiation generated plasmas, a challenge in modern lithography*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2008.

[54] J.J.A.M. van der Mullen, J.M. Palomares Linares, E.A.D. Carbone, W.A.A.D. Graef, and S. Hübner. High rep-rate laser induced fluorescence applied to surfatron induced plasmas: the method, proof of principle and preliminary results. *Journal of Instrumentation*, 2012. Submitted.

[55] Jim Beveridge. Self-registering objects in C++. *Dr. Dobb's Journal*, 288:38, 1998.

[56] The TBCI templated C++ numerical library. `http://plasimo.phys.tue.nl/TBCI`.

[57] W.H. Press. *Numerical recipes: the art of scientific computing*. Cambridge University Press, 2007.

[58] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, 2010.

[59] J.R. Dormand and P.J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19 – 26, 1980.

[60] Alan C. Hindmarsh. LSODE and LSODI, two new initial value ordinary differential equation solvers. *SIGNUM Newsl.*, 15:10–11, December 1980.

[61] Linda Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing*, 4(1):136–148, 1983.

[62] N. Saito and I.H. Suzuki. Multiple photoionization in Ne, Ar, Kr and Xe from 44 to 1300 eV. *Int. J. Mass Spectrom. Ion Processes*, 115:157–172, 1992.

[63] M.C.M. van de Sanden, J.M. de Regt, and D.C. Schram. Recombination of argon in an expanding plasma jet. *Phys. Rev. E*, 47(4):2792–2797, Apr 1993.

[64] A.V. Phelps. The application of scattering cross sections to ion flux models in discharge sheaths. *Journal of Applied Physics*, 76(2):747–753, 1994.

[65] P.M. Bellan. *Fundamentals of plasma physics*. Cambridge University Press, Cambridge, UK, first edition, 2006.

[66] NIST atomic spectra database. `http://physics.nist.gov/PhysRefData/ASD/index.html`.

[67] Àngel Yanguas-Gil, José Cotrino, and Luís L. Alves. An update of argon inelastic cross sections for plasma discharges. *Journal of Physics D: Applied Physics*, 38(10):1588, 2005.

[68] M.A. Khakoo, P. Vandeventer, J.G. Childers, I. Kanik, C.J. Fontes, K. Bartschat, V. Zeman, D.H. Madison, S. Saxena, R. Srivastava, and A.D. Stauffer. Electron impact excitation of the argon $3p^5 4s$ configuration: differential cross-sections and cross-section ratios. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 37(1):247, 2004.

[69] J. Ethan Chilton, John B. Boffard, R. Scott Schappe, and Chun C. Lin. Measurement of electron-impact excitation into the $3p^5 4p$ levels of argon using Fourier-transform spectroscopy. *Phys. Rev. A*, 57:267–277, Jan 1998.

[70] M. Hayashi. *Bibliography of electron and photon cross sections with atoms and molecules published in the 20th century: argon*. NIFS data series. National Inst. for Fusion Science, 2003.

[71] Tobin Weber, John B. Boffard, and Chun C. Lin. Electron-impact excitation cross sections of the higher argon $3p^5 np$ $(n = 5, 6, 7)$ levels. *Phys. Rev. A*, 68:032719, Sep 2003.

[72] J.E. Chilton and C.C. Lin. Measurement of electron-impact excitation into the $3p^5 3d$ and $3p^5 5s$ levels of argon using Fourier-transform spectroscopy. *Phys. Rev. A*, 60(5):3712–3721, Nov 1999.

[73] A.I. Imre, A.I. Dashchenko, I.P. Zapesochnyi, and V.A. Kel'Man. Cross Sections for the Excitation of Ar II Laser Lines in Electron-Ion Collisions. *ZhETF Pis ma Redaktsiiu*, 15:712, June 1972.

[74] J. B. Boffard, C. C. Lin, and C. A. DeJoseph, Jr. TOPICAL REVIEW: Application of excitation cross sections to optical plasma diagnostics. *Journal of Physics D Applied Physics*, 37:143–161, June 2004.

[75] J.B. Boffard, B.C., T. Weber, and C.C. Lin. Electron-impact excitation of argon: Optical emission cross sections in the range of 300-2500 nm. *Atomic Data and Nuclear Data Tables*, 93(6):831 – 863, 2007.

[76] C.B. Opal, W.K. Peterson, and E.C. Beaty. Measurements of secondary-electron spectra produced by electron impact ionization of a number of simple gases. *J. Chem. Phys.*, 55:4100–4106, 1971.

[77] J. Jonkers. High power extreme ultra-violet (EUV) light sources for future lithography. *Plasma Sources Science and Technology*, 15(2):S8–S16, 2006.

[78] Lennart Minnhagen. Spectrum and the energy levels of neutral argon, ArI. *J. Opt. Soc. Am.*, 63(10):1185–1198, Oct 1973.

[79] O. Zatsarinny and K. Bartschat. B-spline Breit-Pauli R-matrix calculations for electron collisions with argon atoms. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 37(23):4693, 2004.

[80] Donald Rapp and Paula Englander-Golden. Total cross sections for ionization and attachment in gases by electron impact. I. Positive ionization. *The Journal of Chemical Physics*, 43(5):1464–1479, 1965.

[81] C.B. Collins. Analog to the Saha equation for recombining high-pressure plasmas in which the electron temperature exceeds the gas temperature. *Phys. Rev.*, 158:94–96, Jun 1967.

[82] K. Dzierżęga, K. Musioł, E. C. Benck, and J. R. Roberts. Electron density measurement in a rf helium plasma by laser-collision induced fluorescence method. *Journal of Applied Physics*, 80:3196–3201, September 1996.

[83] R.S. Stewart, D.J. Smith, I.S. Borthwick, and A.M. Paterson. Model for cw laser collisionally induced fluorescence in low-temperature discharges. *Phys. Rev. E*, 62:2678–2683, Aug 2000.

[84] E.V. Barnat and K. Frederickson. Two-dimensional mapping of electron densities and temperatures using laser-collisional induced fluorescence. *Plasma Sources Science and Technology*, 19(5):055015, 2010.

[85] M. Moisan, Z. Zakrzewski, and R. Pantel. The theory and characteristics of an efficient surface wave launcher (surfatron) producing long plasma columns. *Journal of Physics D: Applied Physics*, 12(2):219, 1979.

[86] N. de Vries. *Spectroscopic study of microwave induced plasmas : exploration of active and passive methods*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2008.

[87] J.M. Palomares, E. Iordanova, E.M. van Veldhuizen, L. Baede, A. Gamero, A. Sola, and J.J.A.M. van der Mullen. Thomson scattering on argon surfatron plasmas at intermediate pressures: Axial profiles of the electron temperature and electron density. *Spectrochimica Acta Part B: Atomic Spectroscopy*, 65(3):225 – 233, 2010.

[88] J.T. Gudmundsson. Ionization mechanism in the high power impulse magnetron sputtering (HiPIMS) discharge. *Journal of Physics: Conference Series*, 100(8):082013, 2008.

[89] Vladimir Kouznetsov, Karol Macák, Jochen M. Schneider, Ulf Helmersson, and Ivan Petrov. A novel pulsed magnetron sputter technique utilizing very high target power densities. *Surface and Coatings Technology*, 122(2-3):290 – 293, 1999.

[90] J.T. Gudmundsson, J. Alami, and U. Helmersson. Spatial and temporal behavior of the plasma parameters in a pulsed magnetron discharge. *Surface and Coatings Technology*, 161(2-3):249 – 256, 2002.

[91] B.M. DeKoven, P.R. Ward, R.E. Weiss, R.A. Christie, W. Scholl, D. Sproul, F. Tomasel, and A. Anders. Carbon thin film deposition using high power pulsed magnetron sputtering. In *SVC - Society of Vacuum Coaters, Albuquerque,NM*, page 158, 2003.

[92] Johan Bohlmark, Jones Alami, Chris Christou, Arutiun P. Ehiasarian, and Ulf Helmersson. Ionization of sputtered metals in high power pulsed magnetron sputtering. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 23(1):18–22, 2005.

[93] J.T. Gudmundsson. Notes on the electron excitation rate coefficients for argon and oxygen discharge. Technical Report RH-21-2002, Science Institute, University of Iceland, 2002.

[94] K. Tachibana. Excitation of the $1s_5$,$1s_4$, $1s_3$, and $1s_2$ levels of argon by low-energy electrons. *Phys. Rev. A*, 34:1007–1015, Aug 1986.

[95] C. Lee and M.A. Lieberman. Global model of Ar, O_2, Cl_2, and Ar/O_2 high-density plasma discharges. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 13(2):368–380, 1995.

[96] J.A. Hopwood. *Thin films*. Number v. 27 in Thin Films. Academic Press, 2000.

[97] Junqing Lu and Mark J. Kushner. Effect of sputter heating in ionized metal physical vapor deposition reactors. *Journal of Applied Physics*, 87(10):7198–7207, 2000.

[98] S.M. Rossnagel, J.J. Cuomo, and W.D. Westwood. *Handbook of plasma processing technology: fundamentals, etching, deposition, and surface interactions*. Materials science and process technology series. Noyes Publications, 1990.

[99] W.H. Hayward and A.R. Wolter. Sputtering yield measurements with low-energy metal ion beams. *Journal of Applied Physics*, 40(7):2911 –2916, jun 1969.

[100] B.W. Kernighan and D.M. Ritchie. *The C Programming Language*. Prentice-Hall software series. Prentice Hall, 1988.

[101] B. Stroustrup. *The C++ programming language*. Addison-Wesley, 1991.

Acknowledgements / Dankwoord

Toen ik in 2007 als afstudeerder officieel lid werd van de groep EPG had ik nooit kunnen denken dat dat zou resulteren in het werk dat voor u ligt. Met name omdat de eerste taak die ik kreeg bestond uit het verdedigen van het doel tijdens het (eerste) jaarlijkse voetbaltoernooi van de faculteit natuurkunde. Alhoewel dat niet illustratief is voor de productiviteit binnen de groep, is het dat wel voor de aangename sfeer. Vanaf het begin heb ik me in de groep thuis gevoeld, temeer omdat ik met open armen werd ontvangen (het toernooi verliep voorspoedig).

Afgezien van experimentele strubbelingen is het afstudeeronderzoek onder begeleiding van Joost van der Mullen me goed bevallen. Het kostte hem dan ook weinig overredingskracht om mij nog wat langer in de groep te laten blijven om hem te assisteren met de cursus numerieke plasmafysica. Er werd al gepolst of ik zin had in een promotie, maar zo'n lang traject zag ik niet zitten.

Na ruim een jaar heeft Joost me toch overgehaald om eens wat dingen op te gaan schrijven, wat geresulteerd heeft in dit proefontwerp. Het moge duidelijk zijn dat ik Joost veel dank verschuldigd ben. Hij heeft me van begin af aan met veel kennis, energie en humor begeleid. Joost, bedankt voor je tomeloze inzet, al de tijd die je aan mij hebt opgeofferd en al de moeite die je hebt gedaan mij langer van deze groep deel te laten zijn.

Daarnaast wil ik mijn tweede promotor Wim Goedheer bedanken. Ondanks zijn drukke schema zag hij toch kans mij van nuttig commentaar te voorzien, zelfs op materiaal dat ik hem pas op het laatste nippertje had opgestuurd. Ook mijn copromotor Jan van Dijk ben ik dankbaar voor zijn feedback, niet zelden binnen één minuut na een cvs commit.

Furthermore, I would like to thank my other committee members Gerrit Kroesen, Gérard Degrez, Charles de Izarra, Leon Kamp, and Yi-Kang Pu. Thank you for taking the time to read my thesis and to attend my defense.

Mijn positie als promovendus binnen de groep EPG heb ik als een zeer prettige werkplek ervaren. Het is een groep met een grote verscheidenheid, maar ik heb met iedereen goed kunnen opschieten. Op mijn "privé"-kamer, die me na de

verhuizing vanuit n-laag is toebedeeld, was het soms wel erg rustig zonder het af en toe kunnen kletsen met mijn voormalige kamergenoten Niels, Simon en Jesper, alhoewel dat wel goed is gebleken voor de productiviteit.

De overige leden van de groep wil ik ook graag danken voor de gezelligheid en hulpvaardigheid, met name secretaresse extraordinaire Rina, de leden van het plasimo-team Manolo, Lei, Efe, Diana, Kim en Sara, en de experimentalisten José, Ana en Emile. Vooral met Emile heb ik vele nuttige discussies gehad, en zijn vaardigheden in het testen van mijn code, hoewel soms frustrerend, zijn zeer waardevol geweest. Dank daarvoor.

Niet alleen een goede werkplek is belangrijk voor het succesvol afronden van een promotie traject. Ook de privésfeer speelt een belangrijke rol door de steun en nodige afleiding die ze bieden. Al mijn vrienden, die ik eigenlijk wel kan samenvatten als de leden en aanhang van het illustere (helaas ter ziele gegane) TOP-Eindhoven en huidige TOB-7, wil ik daarvoor danken. Speciaal dank aan mijn twee paranimfen Roel en Frank, en mijn vaste kompaan voor punk concerten, Casper.

Dank ook aan mijn ouders, broers, zussen en geweldige kinderen die jullie op deze aardbol hebben neergezet. Ik heb jullie in de laatste fase van mijn promotie veel te weinig opgezocht, ik zal proberen mijn leven te beteren.

Verder wil ik mijn familie in Eindhoven bedanken. Ria, maar zeker ook Henk en Joek. Jullie hebben deze "eeuwige student" met heel jullie hart geaccepteerd. Ik zal nooit vergeten hoe Joek me elke maand weer kwam motiveren met een grote zak drop. Joek, het heeft gewerkt!

Ten slotte wil ik de belangrijkste persoon in mijn leven bedanken. Sarah, dank je wel dat je het na al die avonden en weekenden dat ik me op de studeerkamer terugtrok nog steeds met me uithoudt. Ik hou van je.

Wouter Graef, Eindhoven 2012

Curriculum vitæ

24 January 1976

Born in Horn, the Netherlands.

1988–1994

Pre-university education, Sg. St. Ursula, Horn, the Netherlands.

1994–2009

M.Sc. in Applied Physics at Eindhoven University of Technology, the Netherlands.

- Traineeship in the Vortex Dynamics group, department of Applied Physics at the Eindhoven University of Technology, the Netherlands.
  Subject: Numerical simulation of tripoles on the gamma-plane using contour dynamics.
- Traineeship at Máxima Medisch Centrum, Veldhoven, the Netherlands.
  Subject: Validation of the Finometer in sympathovagal research in adults and neonates.
- Master's thesis project at ASML, Veldhoven, the Netherlands.
  Subject: Time dependent collisional radiative model of an extreme ultraviolet driven plasma.

2009–2012

Ph.D. candidate in the Elementary Processes in Gas discharges group, department of Applied Physics at Eindhoven University of Technology, the Netherlands.
Subject: Zero-dimensional models for plasma chemistry.

Glossary

(E)EDF — (Electron) Energy Distribution Function — *Description of the energy distribution over a species (or electrons).*

ASDF — Atomic State Distribution Function — *Description of the distribution of the density of an atomic species over its excited energy levels.*

BDF — Backward Differentiation Formula — *Algorithm for solving stiff ordinary differential equations.*

CB — Corona Balance — *Condition in which levels are populated by electron collisional excitation from the ground state and depopulated by radiative decay.*
CRM — Collisional Radiative Model — *A model of the various collisional and radiative processes in an atomic or ionic system.*
CV — Control Volume method — *Fluid modeling method using a mesh of discrete static volumes.*

ECC — Effective Conversion Coefficient — *The coefficient describing the conversion between two species including direct and indirect reaction paths.*
EEK — Electron Excitation Kinetics
ESB — Excitation Saturation Balance — *Condition in which levels are populated and depopulated by electron impact.*
EUV(L) — Extreme UltraViolet (Lithography) — *(IC manufacturing technique using) light at a wavelength of 13.5 nm*

GPM — Global Plasma Model — *A volume averaged plasma model, predicting mean plasma parameters as function of external control parameters, such as power density.*
GUI — Graphical User Interface

HEK — Heavy particle Excitation Kinetics
HID — High Intensity Discharge
HiPIMS — High Power Impulse Magnetron Sputtering — *Method for achieving high plasma densities through the use of short, intense pulses to a magnetron plasma.*

IC — Integrated Circuit
ICP — Inductively Coupled Plasma — *Plasma source in which the energy is supplied through electromagnetic induction.*

LC — Local Chemistry — *Classification of a species whose (de)population mechanics are predominantly in excitation space.*
LCIF — Laser Collisional Induced Fluorescence — *LIF experiment were the radiation from a level linked to a laser pumped or laser depleted level is observed.*
LIF — Laser Induced Fluorescence — *Experimental technique in which an excited level is populated by a tuned laser and the radiation of the excited level is observed.*
LSODA — Livermore Solver for Ordinary Differential equations, with Automatic method switching — *Solver for stiff and non-stiff ordinary differential equations that automatically switches to the most appropriate method.*
LTE — Local Thermal Equilibrium

MAR — Molecular Assisted Recombination — *A reaction sequence in which the conversion of an atomic ion into a molecular ion is followed by dissociative recombination.*
MD2D — Micro-Discharge 2-Dimensional — *Code aimed at modeling plasma–electrode interactions.*
MEK — Mixed Excitation Kinetics
MIP — Microwave Induced Plasma

OES — Optical Emission Spectroscopy
OOP — Object Oriented Programming — *Programming paradigm using objects that contain both data and functionality.*

PDR — Principal Density Reservoir — *A species with a relatively high density.*
PIC-MC — Particle-In-Cell Monte-Carlo — *Particle-in-cell is a modeling technique that tracks trajectories of individual (super)particles, while their interaction is modeled through an averaged field assigned to a mesh. Monte-Carlo is a stochastic technique to determine collisional occurrences and their outcome.*
PLASIMO — Plasma Simulation Model — *General plasma modeling framework.*
PMT — PhotoMultiplier Tube

QSS(S) — Quasi Steady-State (Solution) — *State or solution achieved when the densities of excited levels (LC) can be described in terms of the densities of a small number of species that are governed by transport (TS).*

REM — Reaction Exploration Model — *A model focusing on the interplay between species through chemical reactions.*
RRD — Reaction Rate Domain — *Description of the source vector originating from individual reaction rates.*
RTCV — Ray-Trace Control-Volume method — *CV method to which the tracing of light paths through control volumes has been added.*

SDD — Species Density Domain — *Description of the source vector originating from species densities.*
SIP — Surfatron Induced Plasma — *A Surfatron is a launcher that produces surface wave discharges.*
SRO — Self Registering Object — *Programming concept allowing for classes to be registered at run time.*
STL — Standard Template Library — *A C++ software library.*

TALIF — Two-photon Absorption Laser Induced Fluorescence — *LIF technique in which a transition of double the photon energy is pumped.*
TBCI — Temporary Base Class Idiom — *A high performance C++ library for numerical calculations.*
TS — Transport Sensitive — *Classification of a species whose (de)population is also affected by transport in configuration space.*

XML — Extensible Markup Language — *Language defining rules for encoding documents.*

ZDM — Zero-Dimensional Model — *The* plasimo *model plug-in implementing the GPM described in this thesis.*

ISBN 978-90-386-3168-4