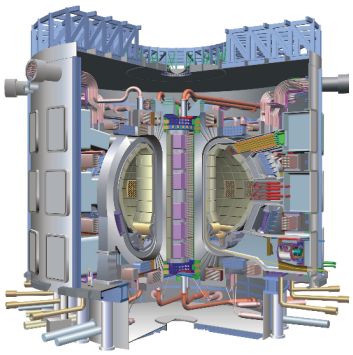
# Modelling of dust behaviour in fusion plasmas

Guillermo Suarez Lopez



Education and Culture DG

Master of Science in Nuclear Fusion and Engineering Physics

Erasmus Mundus

# Modelling of dust behaviour in fusion plasmas

Master Thesis
presented by

**Guillermo Suárez López**

Thesis Promoters

| First Promoter | Second Promoter |
|---|---|
| Dr. Carsten Lechte | Dr. Stéphane Heuraux |

Thesis Supervisors
Dr. Frédéric Brochard
Andrey Shalpegin

July 6th, 2015

UNIVERSITEIT GENT

UNIVERSIDAD COMPLUTENSE MADRID

Universidad Carlos III de Madrid

Universität Stuttgart

UNIVERSITÉ DE LORRAINE

# Modelling of dust behaviour in fusion plasmas

Master Thesis
presented by

**Guillermo Suárez López**

Thesis Promoters

| First Promoter | Second Promoter |
|---|---|
| Dr. Carsten Lechte | Dr. Stéphane Heuraux |

Thesis Supervisors
Dr. Frédéric Brochard
Andrey Shalpegin

Erasmus Mundus Program on Nuclear Fusion Science and Engineering Physics

July 6th, 2015

# Abstract

Fusion energy and specially modern fusion devices such as tokamaks and stellarators face a remarkable number of challenges in order to become the energy of the future. One of this difficulties arises from the production of microscopic material structures that are sputtered from the reactor walls and divertor plates when plasma heat loads impinge on them, namely, dust particles. These particles, once they are released from the bulk material, are incorporated into the clean plasma flow becoming a source of impurities as they evaporate and even a threat for plasma facing components and diagnostics, since dust collisions with the reactor walls may severely damage essential components. Furthermore, fusion efficiency is drastically lost when these particles reach the core region and in limit situations they may terminate the discharge. Hence, the scientific community has been increasingly interested in understand the basic physics of dust-plasma interactions and how to develop methods for modeling and control these dust particles. One of these methods comes from numerical models able to reproduce the behavior of observed dust grains and make accurate predictions on their behavior; in this line of though several codes have been developed in the past few years, e.g. DTOKS, DUSTT, and MIGRAINe, specially orientated to study dust-plasma and more recently, dust-wall interactions. A new code is presented in this thesis, DUCAD, namely, DUst Characterization And Dynamics. DUCAD aims to reproduce previous numerical results and also to push the boundaries a little bit further by introducing some new concepts in this terrain. DUCAD has also been coupled to the transport code EMC3/EIRENE and is able to study dust trajectories as well as dust intrinsic properties for relevant fusion machines such as Asdex Upgrade tokamak. In this thesis, the theoretical foundation for DUCAD is first presented and then, the code is used to study dust reconstructed trajectories in Asdex Upgrade tokamak.

# Contents

# Chapter 1

# Introduction

In order to achieve a viable fusion power plant in the future, certain conditions must be met. The rate of energy production needs to be considerably higher than the required energy to set the fusion reactions and in order to do so, critical temperature and density conditions in the core of the reactor are necessary. However, impurities from the erosion of the reactor walls, helium ash which is an intrinsic product of fusion reactions and seeded impurities, often built up in the core and thermal radiation from these impurities difficult a steady operational scenario (e.g. see Ref. [**1**]). In addition, impurities reduce the concentration of deuterium and tritium in the core which lowers the efficiency of the process, for instance, if we consider the density of deuterium and tritium ions being $n_D$ and $n_T$, the density of helium ions $n_{He}$ and some impurities with charge number $Z_i$ and density $n_{Zi}$, the concentration of "diluted" fuel can be derived as:

$$\begin{aligned} & n_D + n_T + 2n_{He} + \sum_i Z_i n_{Zi} = n_e^{Tot} \rightarrow \\ & \rightarrow n_D + n_T = n_e^{Tot} - 2n_{He} - \sum_i Z_i n_{Zi} \rightarrow C_{DT} = 1 - 2C_{He} - \sum_i Z_i C_{Zi} \end{aligned} \tag{1.1}$$

Where $n_e^{Tot}$ is the total electron density in the core and $C_i$ are the concentrations of the different $i$ species. In divertor configuration, the plasma bulk is confined in a central region of the reactor where closed magnetic field lines exist, however, due to transport processes, some plasma structures abandon this zone and escape through the separatrix (the boundary between closed and opened magnetic field lines). Plasma that diffuses outside the core follows the open magnetic lines to the divertor plates, a set of metallic surfaces that receive the heat loads from the hot plasma and act as a removal system of the helium exhaust, impurities, and plasma fuel thanks to the pumping systems. They also act as an active recycling region for the plasma in which plasma ions can impinge, recombine as neutrals and be used again as fuel when not trapped due to retention processes. The intense heat of the plasma often partially melts thin layers of the divertor plates and this is considered as one of

the main source of impurities in modern fusion machines; traditionally, the same effects were found in the total limiter configuration and this motivated the exploration of the divertor configuration as a more efficient confining and clean magnetic geometry. Moreover, plasma transport along the magnetic lines does not guarantee a complete isolation of the first wall and turbulent transport often exposes it to heat loads that sputter material as well. Furthermore, transient events such as edge localized modes and disruptions are also a source of impurity creation [**2, 3**] because of the remarkable power loads to the divertor and walls during them. When these impurities attain a considerable size that ranges from a few tens of nanometers to several hundreds of micrometers (the most usual cases), we no longer refer to them as impurities but dust grains, microscopic formations of material that are removed from the reactor walls and enter the plasma circulation. The principal causes for dust formation are flaking of re-deposited material layers on the plasma facing components, arcing, chemical vapor deposition [**4**], agglomeration due to electrostatic processes and sputtering. All of these phenomena have been observed in the majority of current fusion devices and the creation of these dust particles may become a crucial problem for long discharges in ITER as a major increase in dust production is expected from the direct scaling of the reactor dimensions and plasma parameters. Discharge terminations caused by dust migration have been observed in long pulse discharges in the Large Helical Device (LHD) [**5**], Tore Supra [**6**], Alcator C-mod, Asdex Upgrade [**7**] and several other machines, which highlights the importance of a proper dust physics understanding, modeling and control. Furthermore, there are also various hazards and safety issues related to dust formation, for instance, chemical toxicity of metals such as Beryllium, radioactive activation by neutrons and tritium retention, which is a main concern for both safety and economical perspectives. Dust layers formed by deposition may also cause problems related to coolant leaks and chemical reactivity, for instance, in the case of steam injection from a cooling line, dust can react forming hydrogen and leading to an explosive combustion [**8**]. In addition, since remarkable amounts of dust are expected for ITER, dust inventory poses a problem of total radioactivity and degradation of in-vessel components such as engineered surfaces as well as diagnostics [**9**]. Because of these scenarios, increasing efforts have been made to understand and make predictions of dust behavior and numerical codes have been developed in the past few years. Specially it is worth to mention the extraordinary work in DTOKS [**10**], DUSTT [**11**] and MIGRAINe [**12**] which have served as a guide for the development of the code that will be presented in this thesis, **DUCAD**, and will be used for future benchmarking. **DUCAD**, namely DUst Characterization And Dynamics is the result of the implentation of several physical effects affecting dust grains and serves as an attempt to reproduce previous numerical results and even explore experimental data of dust particles in modern fusion devices. The code is written in Python language and focuses on four different observables: dust charge number, temperature, mass evolution and dynamics by solving the corresponding differential equa-

tions. The numerical tools used, such as finite differences, will be indicated when needed but for an insight on these techniques the reader is referred to the extensive bibliography about these topics.
This thesis is organized in the following manner: in section **2** an overview of the different physical phenomena related to dust grains embeded in fusion plasmas and their implementation into DUCAD is presented, in section **3** first simulations using plasma parameters from the transport code EMC3-EIRENE and experimental data registered with fast cameras is shown, finally, in section **4** some conclusion are drawn and future perspectives will be discussed.

# Chapter 2

# Theoretical aspects

## 2.1 Central considerations

The central purpose of this thesis is the study of dust behavior and dynamics in fusion plasmas. To achieve this, some assumptions are imposed on the dust grains:

1. Dust grains' radiuses will be mainly in the micrometer size range. This is the most common population of dust in fusion plasmas [**13**] and therefore, represents the most desirable size for study.

2. Magnetic field effects are neglected on ion and electron currents to the dust grain. For standard SOL parameters we find that $R_d < \lambda_D \sim \rho_e \ll \rho_i$ where $R_d$ is the dust grain radius, $\lambda_D$ is the dust Debye length, $\rho_e$ is the electron gyroradius and $\rho_i$ is the ion gyroradius. For instance, for a magnetic field $B \sim 1T$, an electron temperature of $T_e \sim 20eV$, an electron density of $n_e \sim 10^{18}m^{-3}$, a tungsten grain of radius $R_d \sim 1\mu m$ and considering $v_{\perp}^{e^-,i^+} \sim v_{th}^{e^-,i^+}$, we find $R_d = 1\mu m \ll \rho_e = 10\mu m \sim \lambda_D = 33\mu m \ll \rho_i \sim 447\mu m$. In this regime, Lorentz force and other magnetic effects are marginal on the impact parameters of plasma species.

3. Dust spinning is not considered. Even though there exist several effects contributing to dust spinning such as rocket force, the shear of plasma flow [**14**] and the interplay between the shape of the dust grain and the angular momenta of the plasma ions [**15**], the short lifetime of dust in fusion plasmas usually prevents the grain to achieve high angular velocities.

4. Dust grains will be considered as perfectly spherical. For computational and analytical reasons, this represents an elegant advantage. Furthermore, dust shape diverging from spherical, plays only an important role when dust spinning is important and because of the previous reasoning, this can be disregarded.

## 2.2 The OML Theory

Plasma interactions with surfaces have been widely studied during the past few decades. Dust grains in fusion plasmas can be treated as small spherical biased probes which collect ions and electrons depending upon their potential and develop an electrostatic sheath surrounding them. Therefore, it is possible to calculate the current-voltage characteristic curve using probe theory. The first solid contributions were given by H. M. Mott-Smith and Irving Langmuir in [**16**] where several ion and electron currents were derived for the cases of cylindrical, spherical and planar probes in different velocity distribution function regimes. The main assumption in this work revolved around having a probe embedded in an isotropic plasma which collected ions and electrons depending on the applied potential to the probe and the considered ion velocity distribution function. For micrometer dust in fusion plasmas where the Debye length is much larger than the grain radius and the plasma can be regarded as collisionless, the most convenient variety of the spherical probe theory to use is the **Orbited motion limited theory**. The OML framework assumes that ion orbits bend towards the dust grain when entering the grain sheath and ions might be collected by the grain depending on their impact parameter; if the force field acting upon the plasma species can be considered as conservative, assumptions over the conservation of linear and angular momentum before and after entering the sheath region allows a simple calculation for the currents. Following the work of J. E. Allen in [**17**] currents for attracting and retarding potentials were derived according to this scheme. However, these approximations hold only when the plasma is isotropic, but if we were to consider a dust grain moving with some velocity, the isotropy would be broken due to the apparent motion of the plasma with respect to the dust frame of reference. In this case, both the velocity of the grain or the velocity of the plasma play an important role as it will be shown in the next section.

## 2.3 Dust charging

Dust charging can be described by the differential equation:

$$\frac{dQ_d}{dt} = \sum_Z I_i + I_e + I_{th} + I_{see} \tag{2.1}$$

Where $Q_d = Z_0 e$ is the dust grain charge, $I_i$ is the current of ions with charge number $Z$ and $I_e$ is the current of electrons from the plasma to the dust, $I_{th}$ is the thermionic emission and $I_{see}$ is the secondary electron emission current from dust. However, dust charging can be considered as an instantaneous process and the local charge value can be used, meaning $dQ_d/dt = 0$. Dust charging time can be calculated by assuming small deviations from the equi-

librium charge, yielding [**18**]:

$$\tau_{ch} = \frac{\lambda_{Di}}{R_d} \frac{\sqrt{2}}{w_{pi}\left(1 + \frac{T_i}{T_e} - \frac{e\phi_d}{T_e}\right)} \tag{2.2}$$

Where $w_{pi}$ is the ion plasma frequency, $\lambda_{Di}$ is the ion Debye length and $\phi_d$ is the grain potential. For usual SOL parameters $\tau_{ch} \sim 10^{-8} - 10^{-9}s$, which is much faster than the residence time of the dust grain in one of the cells of the computational grid. Thus:

$$I_{tot} = \sum_Z I_i + I_e + I_{th} + I_{see} = 0 \tag{2.3}$$

Which means the grain would acquire a floating potential. Regarding the potential expression itself, we will derive it from first principles; If we consider ions and electrons following a Maxwell-Boltzmann distribution, we have:

$$f_e(\vec{r}, \vec{v}) = \frac{m_e^{2/3}}{(2\pi K_B T_e)^{2/3}} 4\pi v_e^2 exp\left(\frac{\frac{-1}{2} m_e v_e^2 + e\phi_d}{K_B T_e}\right) \tag{2.4}$$

$$f_i(\vec{r}, \vec{v}) = \frac{m_i^{2/3}}{(2\pi K_B T_i)^{2/3}} 4\pi v_i^2 exp\left(\frac{\frac{-1}{2} m_i v_i^2 + e\phi_d}{K_B T_i}\right) \tag{2.5}$$

The charge density, considering thermal equilibrium $T_e = T_i$ can be obtained as:

$$\rho_i = e\int_0^\infty f_i(\vec{r}, \vec{v}) d^3v \sim e n_0 exp\left(\frac{e\phi_d}{K_B T_i}\right) \tag{2.6}$$

$$\rho_e = e\int_0^\infty f_e(\vec{r}, \vec{v}) d^3v \sim e n_0 exp\left(\frac{e\phi_d}{K_B T_e}\right) \tag{2.7}$$

If ions have enough temperature that they are not affected by the dust potential, the charge density surrounding an spherical object is:

$$\rho = n_0 e\left[1 - exp\left(\frac{\phi_d e}{K_B T_e}\right)\right] \tag{2.8}$$

Where the first term is the unperturbed ion density and the second term is the electron density as a function of the dust potential. With this we may write Poisson's equation in spherical coordinates, yielding:

$$\frac{d^2\phi_d}{dr^2} + \frac{2}{r}\frac{d\phi_d}{dr} \sim \frac{n_0 e}{\epsilon}\left(\frac{\phi_d e}{K_B T_e}\right) \tag{2.9}$$

Where $-(\phi_d e/K_B T_e) \sim 1 - exp(\phi_d e/K_B T_e)$. The solution of this equation gives the Debye potential:

$$\phi_d = \phi_0 \frac{R_d}{r} exp\left(\frac{-(r - R_d)}{\lambda_e}\right) \tag{2.10}$$

Where $R_d$ is the sphere radius and $\lambda_e$ is the electron Debye length. The capacitance of a body can then be obtained from the ratio of this potential to the body charge, thus:

$$C = 4\pi R_d \epsilon_0 \left(\frac{1}{R_d} + \frac{1}{\lambda_e}\right) \sim 4\pi R_d \epsilon_0 \left(\frac{1}{R_d}\right) \tag{2.11}$$

### 2.3.1 Ion and electron currents

The ion current to the dust grain may be calculated from the OML theory considering a shifted Maxwellian distribution due to the relative velocity between the grain and the ion flow [**19, 20**], yielding:

$$I_i = \pi R_d^2 Z e n_i v_{ti} \left[\left(\tilde{u_i} + \frac{1}{2\tilde{u_i}}\left(1 - \frac{2Ze\phi_d}{T_i}\right)\right) erf(\tilde{u_i}) + \frac{1}{\sqrt{\pi}} exp(-\tilde{u_i^2})\right] \quad for \quad \phi_d \leq 0 \tag{2.12}$$

And:

$$I_i = \frac{1}{2}\pi R_d^2 Z e n_i V_{ti} \left[\left(\tilde{u_i} + \frac{1}{2\tilde{u_i}}\left(1 - \frac{2Ze\phi_d}{T_i}\right)\right) [erf(\tilde{u_{ip}}) + erf(\tilde{u_{im}})]\right] + \frac{1}{2}\pi R_d^2 Z e n_i V_{ti} \left[\frac{1}{\sqrt{\pi}} \frac{1}{\tilde{u_i}} \left[\tilde{u_{ip}} exp(-\tilde{u_{im}}^2) + \tilde{u_{im}} exp(-\tilde{u_{ip}}^2)\right]\right] \quad for \quad \phi_d > 0 \tag{2.13}$$

Where, $n_i$ is the ion density, $T_i$ is the ion temperature, $v_{ti} = \sqrt{2T_i/m_i}$ is the ion thermal speed, $\tilde{u_i} = |\vec{u_i} - \vec{V_d}|/v_{ti}$ is the normalized ion flow velocity relative to the dust grain, $\tilde{u_{ip}} = \tilde{u_i} + \sqrt{Ze\phi_d/T_i}$ and $\tilde{u_{im}} = \tilde{u_i} - \sqrt{Ze\phi_d/T_i}$, being $\sqrt{Ze\phi_d/T_i} = (\sqrt{2Ze\phi_d/m_i})/v_{ti}$ the normalized velocity acquired due to the potential of the grain.
On the other hand, the electron current can be casted into a much simpler expression, the reason being that since the thermal velocity of electrons, $v_{te} = \sqrt{2T_e/m_e}$ is much higher than the flow velocity of the grain, e.g. $v_{te} \sim 1.8 * 10^6 m/s$ for an electron temperature of $T_e \sim 10eV$ in comparison with a grain velocity of $V_d \sim 10 - 100m/s$, their distribution does not appear to be shifted in the frame of reference of the dust grain. Thus:

$$I_e = -2\sqrt{\pi} R_d^2 e n_e v_{te} exp\left(\frac{e\phi_d}{T_e}\right) \quad for \quad \phi_d \leq 0 \tag{2.14}$$

$$I_e = -2\sqrt{\pi} R_d^2 e n_e v_{te} \left(1 + \frac{e\phi_d}{T_e}\right) \quad for \quad \phi_d > 0 \tag{2.15}$$

Where $n_e$ is the electron density.

### 2.3.2 Thermionic and secondary emission currents

Thermionic emission currents can be calculated for negatively charged dust using the relation derived by Richardson and Dushman [**21**]. However, as pointed out in the same article, it is convenient to include a small correction for the Schottky effect; electrons emitted from a metal surface must overcome a potential barrier in order to escape the sample, namely the Work function. When there exists an electric field in the metal surface, which is often the case since emitters are negatively biased, the work function is slightly shifted depending on the strength of the electric field, therefore, leading to a correction of the current:

$$I_{th} = \lambda_R \frac{4\pi A_d T_d^2 m_e e}{h^3} exp\left(-\frac{W_f - e\sqrt{zT_e/Rd}}{Td}\right) \quad for \quad \phi \leq 0 \tag{2.16}$$

Where $\lambda_R$ is a material dependent constant, $A_d$ is the surface area of the dust grain, $m_e$ is the electron mass, $h$ is the Planck constant, $W_f$ is the work function of the material and $T_d$ is the dust temperature. It is often the case that the dust grain may achieve a positive potential when intensively heated. In this case, some of the emitted electrons might be reflected back due to the electric field between the vapor electron cloud in front of the emitter and the positively charged emitter itself. This introduces a factor $(1 + e\phi_d/Td)exp(-e\phi_d/T_d)$ [12, 22] leading to:

$$I_{th} = \lambda_R \frac{4\pi A_d T_d^2 m_e e}{h^3} \left(1 + \frac{e\phi_d}{T_d}\right) exp\left(-\frac{e\phi_d}{T_d}\right) \quad for \quad \phi > 0 \tag{2.17}$$

In addition to these effects, secondary electron emission also plays an important role in dust charging dynamics. Secondary emitted electrons are defined as those that are produced due to primary electron collisions from the plasma to the dust grain. When a plasma electron impinges on the dust surface, it penetrates until its energy is lost due to inelastic collisions with the bulk; these collisions, often ionize atomic electrons that can escape the solid into the plasma. This effect is modeled through the secondary electron emission yield which depends on the energy of the primary incident electrons $E_0$ and the incidence angle $\alpha$, $\delta(E_0, \alpha)$. This yield can be conveniently split into energy dependence at normal incidence $\delta_E(E_0)$ and angular dependence $\delta_\alpha(\alpha)$ such that $\delta(E_0, \alpha) = \delta_E(E_0) * \delta_\alpha(\alpha)$. It is a matter of discussion what expression should be used to describe the energy yield $\delta_E(E_0)$ and several semi-empirical formulas have been proposed based on different power stopping laws for the electrons penetrating the solid, the main ones being the Baroody-Dekker formula assuming a generalized power loss law model, the Young-Dekker formula assuming a constant energy loss model and the Sternglass formula using the non relativistic Bethe model. Recent development in the MIGRAINe code suggest that the Young-Dekker formula better approximates experimental results,

hence DUCAD incorporates this yield, which can be expressed as [**23**]:

$$\delta_E(E_0) = \frac{\delta_m}{1 - e^{-r_m}} \left(\frac{E_0}{E_m}\right)^{1-k} \left[1 - exp\left(-r_m \left(\frac{E_0}{E_m}\right)^k\right)\right] \tag{2.18}$$

Where $\delta_m$ is the maximum yield, $E_m$ is the energy corresponding to the maximum yield, and $r_m$ is the non-trivial solution of the equation:

$$r_m = \left[1 - \frac{1}{k}\right] [e^{r_m} - 1] \tag{2.19}$$

Being k a fitting parameter such that $k\epsilon(1, 2]$. The angular dependence is modeled as [**24**]:

$$\delta_\alpha(\alpha) = cos^{-\beta}(\alpha) \tag{2.20}$$

Where $\beta$ is a material dependent constant (see Table **??** for Tungsten parameters). The secondary electron emission currents can be calculated by integrating the electron velocity distribution function along with the SEE cross section, yielding [**12**]:

$$I_{see} = -\frac{2}{2 - \beta} I_e \int_0^\infty x e^{-x} \delta_E(x T_e) dx \quad for \quad \phi_d \leq 0 \tag{2.21}$$

$$I_{see} = -\frac{2}{2 - \beta} \frac{1 + 3\xi}{(1 + \xi)^3} \frac{e^{-z}}{1 - z} I_e \int_{-z}^\infty x e^{-x} \delta_E(x T_e) dx \quad for \quad \phi_d > 0 \tag{2.22}$$

Where $\xi = e\phi_d / W_f$. The integrals are numerically calculated by splitting the x-variable domain in small steps and using a simple rectangle rule:

$$\int_a^b f(x) dx \sim (b - a) f\left(\frac{a + b}{2}\right) \tag{2.23}$$

The domain is chosen to be a compromise between computational time and accuracy of the calculation.

### 2.3.3 Charging numerical scheme

Based on equation **2.3**, DUCAD computes the final potential of the dust grain following three simple steps:

1. A whole range of *candidate* charge numbers $Z_0$ is considered and the calculation of the individual currents is performed for each of the possible charge numbers.

2. The currents are summed thus giving $I_{tot} = f(Z0)$.

3. Finally, the minimum of $abs(I_{tot})$ is found yielding the $Z_0$ for which $I_{tot} \sim 0$

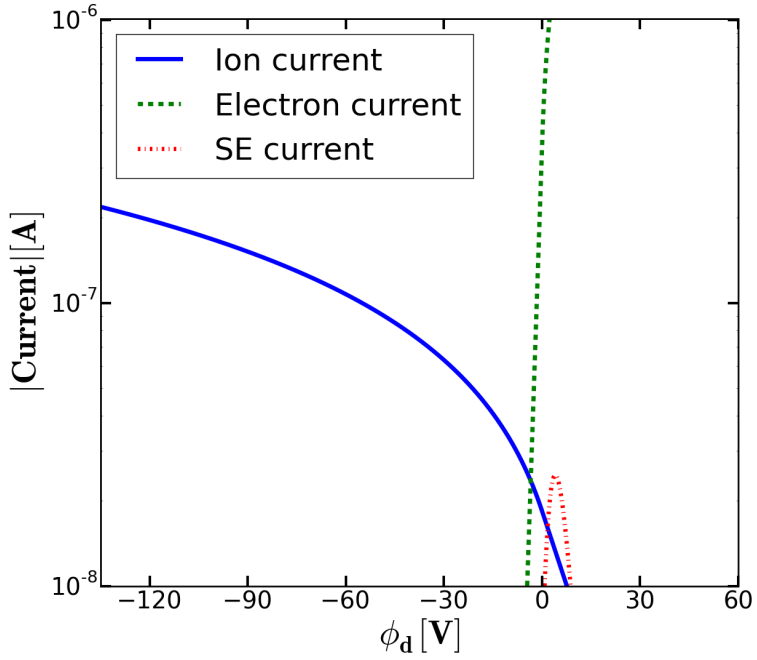



Figure 2.1: Ion, electron and secondary emission currents calculated by DUCAD, affecting the dust grain as a function of its potential for typical SOL parameters, $n_e \sim n_i = 9.4 * 10^{15} m^{-3}, T_i = 12.41eV, T_e = 1.27eV, R_d = 10\mu m$.

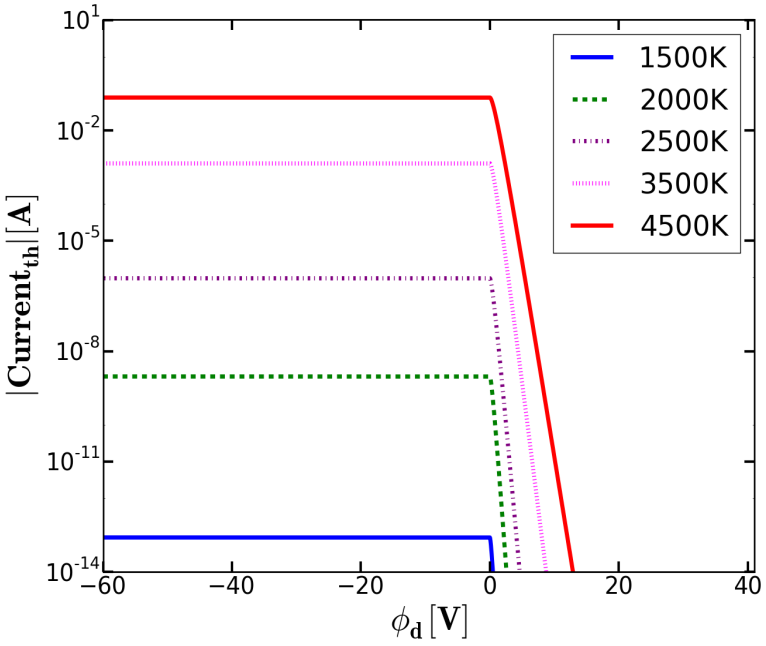



Figure 2.2: Thermionic current calculated by DUCAD, as a function of the dust grain potential for five different temperatures and $R_d = 10\mu m$.

In addition to the already discussed phenomena, there exist other mechanisms by which the dust grain charge is further influenced; electron backscattering is produced due to inelastic reflection within the solid and it is left to be included within DUCAD in a future work. Electron reflection is produced due to elastic reflection at the dust grain surface and it accounts to a small fraction of the total electron current reaching the grain, therefore it is neglected. Ion backscattering is produced when an ion is inelastically reflected from the dust grain, however, since ions are considered to be always recombined when they reach the grain, there is no net ion current leaving the grain. In addition, ion reflection is also neglected.

## 2.4 Dust heating

Another important aspect of dust evolution in fusion plasmas is the continuous heating grains withstand. Energy fluxes acting upon dust grains are due to ion and electron fluxes impinging on them, but more importantly, there exist energy losses from the grain due to thermionic emission, melting and vaporization, secondary electron emission, electron backscattering, ion surface neutralization and thermal radiation. All of these phenomena are accounted through the computation of the molar enthalpy of the dust. The enthalpy of a given element is dependent on pressure, volume and magnetic state; by neglecting magnetization, we are left with an expression of the enthalpy as a function of the temperature and pressure which is available in bibliography [**25**], therefore it is possible to associate an enthalpy change with the corresponding temperature variation, see Figure **2.3**. The differential equation that governs this evolution can be written as:

$$\frac{d(M_d h_d)}{dt} = \sum_Z Q_i + Q_e + Q_{th} + Q_{see} + Q_{isn} + Q_{rad} + Q_{vap} \tag{2.24}$$

Where $h_d$ is the specific enthalpy, $Q_i$ is the heat flux due to absorbed ions, $Q_e$ is the heat flux due to absorbed electrons, $Q_{th}$ is the heat flux associated with thermionic emission, $Q_{see}$ is the heat flux due to secondary electron emission, $Q_{rad}$ accounts for the thermal emissivity losses and $Q_{vap}$ is the heat flux due to vaporized atoms.

### 2.4.1 Ion and electron heating fluxes

Energy fluxes due to impinging ions and electrons differ from the energy value provided by their distribution functions due to the dust sheath that accelerates or decelerates the incoming particles. Therefore, the ion and electron power fluxes to the grain must also be computed following the OML approximation by integrating the currents along with the dust potential for each distribution

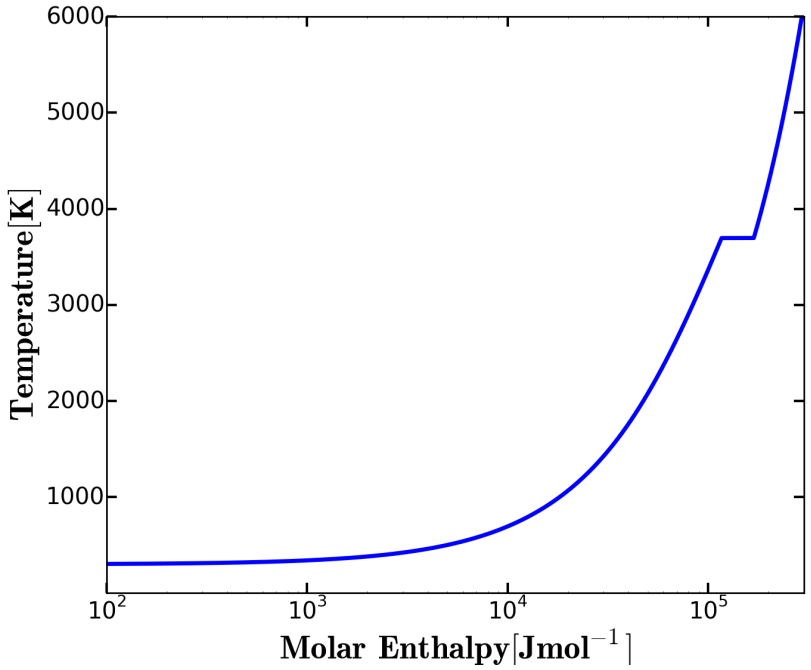



Figure 2.3: Molar enthalpy of Tungsten as a function of temperature for $P \sim 250Pa$.

function in velocity space, yielding [**19**]:

$$Q_i = \pi R_d^2 n_i v_{ti} T_i \left[ \frac{1}{2\sqrt{\pi}} \left[ 5 + 2\tilde{u_i}^2 - \frac{2Ze\phi_d}{T_i} \right] exp(-\tilde{u_i}^2) \right.$$
$$\left. + \frac{1}{4\tilde{u_i}} \left[ 3 + 12\tilde{u_i}^2 + 4\tilde{u_i}^4 - \frac{2Ze\phi_d(1+2\tilde{u_i}^2)}{T_i} \right] erf(\tilde{u_i}) \right] \quad for \quad \phi_d \leq 0 \quad (2.25)$$

$$Q_i = \pi R_d^2 n_i v_{ti} T_i \left[ \frac{1}{4\sqrt{\pi}} \left[ \left( 5 + 2\tilde{u_i}^2 - \frac{3+2\tilde{u_i}^2}{\tilde{u_i}} \sqrt{\frac{Ze\phi_d}{T_i}} \right) exp(-\tilde{u_{ip}}^2) \right. \right.$$
$$\left. + \left( 5 + 2\tilde{u_i}^2 - \frac{3+2\tilde{u_i}^2}{\tilde{u_i}} \sqrt{\frac{Ze\phi_d}{T_i}} \right) exp(-\tilde{u_{im}^2}) \right]$$
$$\left. + \frac{1}{8\tilde{u_i}} \left[ 3 + 12\tilde{u_i}^2 + 4\tilde{u_i}^4 - \frac{2Ze\phi_d(1+2\tilde{u_i}^2)}{T_i} \left[ erf(\tilde{u_{ip}}) + erf(\tilde{u_{im}}) \right] \right] \right] \quad for \quad \phi_d > 0 \quad (2.26)$$

As well as for the currents, since the electron thermal velocity is much higher than the dust grain flow velocity, we can consider that the electron distribution

function is not shifted in the dust frame of reference and we can write:

$$Q_e = 2\sqrt{\pi}R_d^2 n_e v_{te} T_e \left(\frac{2 - e\phi_d}{T_e}\right) exp\left(\frac{e\phi_d}{T_e}\right) \quad for \quad \phi_d \leq 0 \tag{2.27}$$

$$Q_e = 2\sqrt{\pi}R_d^2 n_e v_{te} T_e \left(\frac{2 + e\phi_d}{T_e}\right) \quad for \quad \phi_d > 0 \tag{2.28}$$

### 2.4.2 Thermionic and secondary electron emission heat fluxes

For the thermionic emission heat flux, we can multiply the thermionic current times the total energy removed from the grain per emitted electron, which is the work function of the metal plus the average kinetic energy of each electron. For negative potentials, we need to also consider the reduction of the work function due to the Schottky effect leading to:

$$Q_{te} = -I_{te}\left(W_f - e\sqrt{\frac{-e\phi_d}{R_d}} + 2T_d\right) \quad for \quad \phi_d < 0 \tag{2.29}$$

Where the term $2T_d$ is the average kinetic energy per emitted electron [22]. For positive potentials, we must add a correction factor for reabsorbed electrons, leading to [**12**]:

$$Q_{te} = -I_{te}\left(W_f + \frac{2 + 2\xi + \xi^2}{1 + \xi} Td\right) \quad for \quad \phi_d > 0 \tag{2.30}$$

Where $\xi = e\phi_d/T_d$. Similarly, the secondary electron emission heat flux can be computed by multiplying the secondary electron current times the total energy per emitted electron yielding [**12**]:

$$Q_{see} = -3I_{see}W_f \quad for \quad \phi_d < 0 \tag{2.31}$$

Where $3W_f$ is the sum of the average kinetic energy $\bar{E} \sim 2W_f$ and the work function $W_f$. For the positive grain potential, we must take into account that some electrons are reabsorbed by the grain if their energy is not enough to overcome the potential barrier, therefore:

$$Q_{see} = -3I_{see}W_f\left[\frac{(1 + \xi)(1 + 2\xi)}{(1 + 3\xi)}\right] \quad for \quad \phi_d > 0 \tag{2.32}$$

### 2.4.3 Thermal radiation

Thermal emission from the dust grain follows a gray body radiation law, where:

$$Q_{rad} = -A_d\sigma_{SB}\epsilon_d(R_d, T_d)(T_d^4 - T_w^4) \tag{2.33}$$

Where, $\sigma_{SB}$ is the Stephan-Boltzmann constant, $\epsilon_d(R_d, T_d)$ is the emissivity of the dust grain as a function of its radius and temperature, $T_d$ is the dust temperature and $T_w$ is the wall temperature. Thermal emission plays a main role in dust survival within a hot plasma since it is one of the main cooling channels and therefore, it is essential to perform accurate predictions of dust emissivity to assure a proper characterization of its behavior in current and future fusion reactors. First, we can define the emissivity from Kirchoff's law which states that for a body in thermal equilibrium the energy absorbed and emitted must be equal, thus:

$$\epsilon(\lambda, R_d, T_d)P(\lambda, T_d) = \pi Q_{abs}(\lambda, T_d, R_d)P(\lambda, T_d) \tag{2.34}$$

Where $\epsilon(\lambda, R_d, Td)$ is the emissivity, $Q_{abs}(\lambda, T_d, R_d)$ is the absorption efficiency and $P(\lambda, T_d)$ is Planck's radiation function, namely:

$$P(\lambda, T_d) = \frac{2c^2h}{\lambda^5}\frac{1}{exp\left(\frac{hc}{\lambda K_b T_d}\right) - 1} \tag{2.35}$$

Being $c$ the speed of light and $h$ Planck's constant. Therefore, by integrating we may obtain the emissivity as a function of dust radius and temperature:

$$\epsilon(R_d, T_d) = \int_0^\infty \frac{\pi Q_{abs}(\lambda, T_d, R_d)P(\lambda, T_d)}{P(\lambda, T_d)}d\lambda = \frac{I_{rad}}{\sigma_{SB}T_d^4} \tag{2.36}$$

The computation of $I_{rad}$ can be performed using Mie theory which precisely derives the cross sections for absorbed and scattered light of a homogeneous sphere; following the exquisite derivation in **[26]**, the absorption cross section can be written as:

$$Q_{abs} = \frac{1}{\pi R_d^2}\frac{2}{k^2}\left(\sum_{\nu=1}^{\infty}(2\nu+1)\left(Re[a_\nu + b_\nu] - [|a_\nu|^2 + |b_\nu|^2]\right)\right) \tag{2.37}$$

With $a_\nu$ and $b_\nu$ the scattering coefficients expressed as:

$$a_\nu = \frac{m\psi_\nu(mx)\psi'_\nu(x) - \psi_\nu(x)\psi'_\nu(mx)}{m\psi_\nu(mx)\xi'_\nu(x) - \xi_\nu(x)\psi'_\nu(mx)} \tag{2.38}$$

$$b_\nu = \frac{\psi_\nu(mx)\psi'_\nu(x) - m\psi_\nu(x)\psi'_\nu(mx)}{\psi_\nu(mx)\xi'_\nu(x) - m\xi_\nu(x)\psi'_\nu(mx)} \tag{2.39}$$

These coefficients are dependent of the Riccati-Bessel functions $\psi_\nu$ and $\xi_\nu$ and a detailed explanation can be found in Appendix **A.1**. Also, the variable $m$ is the relative index of refraction defined as the index of refraction of the dust grain $\tilde{n}$ over the real index of refraction of the environment $n_0$ which for the case of a plasma is taken to be a transparent medium, meaning $n_0 = 1$; $x = 2\pi n_0 R_d/\lambda$ is the so-called size parameter and $k = 2\pi n_0/\lambda$ is the usual wave

vector. The index of refraction $\tilde{n}$ can be computed for a wide range of metals by following Drude approximation, yielding the well-known expressions:

$$\tilde{n} = n + i\kappa = \sqrt{\frac{|\epsilon| + \epsilon_1}{2}} + i\sqrt{\frac{|\epsilon| - \epsilon_1}{2}} \tag{2.40}$$

Where $\epsilon = \epsilon_1 + i\epsilon_2$ is the dielectric function characteristic of every metal, which can be expressed as **[27, 28]**:

$$\epsilon_1 = 1 - \frac{w_p^2}{w^2 + w_\tau^2} \tag{2.41}$$

$$\epsilon_2 = \frac{w_p^2 w_\tau}{w^3 + w w_\tau^2} \tag{2.42}$$

Where $w$ is the angular frequency, $w_p$ is the plasma frequency of the metal and $\omega_\tau$ is the Drude collision frequency for the electrons. These two expressions can be rewritten as a function of the metal optical conductivity by using:

$$\sigma_{opt} = \frac{w_p^2}{\epsilon_0 \omega_\tau} \rightarrow \omega_\tau = \frac{w_p^2}{\epsilon_0 \sigma_{opt}} \tag{2.43}$$

Which is convenient, since we can express the optical conductivity of a metal as a function of the DC conductivity which is temperature dependent and thus obtain the temperature dependent index of refraction of the dust grain. The plasma frequency for infrared emission, which is the main range of emission of the micrometer size dust grains, can be found in literature **[28]**. Furthermore, we can relate the optical conductivity to the DC conductivity in infrared range, by introducing a correction factor $f$ **[29]** such that:

$$\sigma_{DC} = f\sigma_0 \tag{2.44}$$

Fits of $\sigma_{DC}$ as a function of temperature were obtained from **[30]**. With these ingredients, $Q_{abs}$ is numerically calculated by DUCAD following the scheme presented in **[26]**. First, the Riccati-Bessel functions $\psi_\nu$ and $\xi_\nu$ are evaluated using upward recurrence until the step $\nu_{max} = x + 4x^{1/3} + 2$, such that:

$$\psi_\nu(z) = \frac{2\nu - 1}{z}\psi_{\nu-1}(z) \tag{2.45}$$

$$\phi_\nu(z) = \frac{2\nu - 1}{z}\psi_{\nu-1}(z) \tag{2.46}$$

Where $\psi_\nu(z)$ and $\phi_\nu(z)$ are the Riccati-Bessel functions of the first and second kind and $\xi_\nu = \psi_\nu - i\phi_\nu$. Truncation at $\nu_{max}$ is performed because $\psi_\nu$ is unstable against upward recurrence and upper terms induce a big roundoff error. From here, $a_\nu$ and $b_\nu$ can be evaluated, however, even though the shape of equations [2.29] and [2.30] are the most elegant ones, they are not suitable

for computational calculus and a logarithmic derivative must be introduce, such that:

$$D_\nu(z) = \frac{d}{dz} ln\psi_\nu(z) = \frac{-\nu}{z} + \frac{\psi_{\nu-1}(z)}{\psi_\nu(z)} \tag{2.47}$$

Now, $a_\nu$ and $b_\nu$ can be recast into:

$$a_\nu = \frac{(D_\nu(mx)/m + \nu/m)\psi_\nu(x) - \psi_{\nu-1}(x)}{(D_\nu(mx)/m + \nu/m)\xi_\nu(x) - \xi_{\nu-1}(x)} \tag{2.48}$$

$$b_\nu = \frac{(mD_\nu(mx) + \nu/m)\psi_\nu(x) - \psi_{\nu-1}(x)}{(mD_\nu(mx) + \nu/m)\xi_\nu(x) - \xi_{\nu-1}(x)} \tag{2.49}$$

The computation of $D_\nu$ is performed by downward recurrence which assures numerical stability, starting at $\nu'_{max} = int(max(\nu_{max}, |z|)+15)$. The recurrence relation is [**31**]:

$$D_{\nu-1}(z) = \frac{n}{z} - \frac{1}{D_\nu(z) + \frac{\nu}{z}} \tag{2.50}$$

In the last step, $Q_{abs}$ is summed by DUCAD and truncated at $\nu_{max}$ which yields the final absorption efficiency factor. Now, the integral $I_{rad}$ is evaluated; since Drude model is not suitable for the computation of the dielectric function in high frequencies, we consider a frequency cut-off $\lambda_c$ which has a minimal impact in the integral evaluation [**32**]:

$$I_{rad} = \pi \int_{\lambda_c}^{\infty} Q_{abs}(T_d, R_d, \lambda) P(t_D, \lambda) d\lambda \tag{2.51}$$

Where $\lambda_c = 4\pi c/w_p$. The integral is numerically calculated by splitting the wavelength domain in small steps and using a simple rectangle rule, as explained in equation **2.23**. The final emissivities can be visualized in Figure **2.4**. It is clear from the use of the Drude model that the obtained emissivities should only be valid as long as the metal preserves its solid state and thus any result for melted or vaporized materials has to be remodeled using a different theory. However, for the sake of simplicity, DUCAD uses this scheme even for temperatures beyond the melting point of the material, for which the scarce bibliographic data has been compensated by using linear fits to the available data points. It is a work for the future to introduce suitable analytical considerations for the liquid state.

### 2.4.4 Ion surface neutralization

Interactions between an excited atom or an ion and a metallic surface include resonance, Auger and radiative processes. However, the probability of the radiative interactions is very low [**33**] and therefore they will be neglected. Auger processes of interest are, Auger neutralization and Auger resonant neutralization accompanied with Auger de-excitation. To give a simple picture, in the first case, the ion interacts with the dust grain and initially one of the metal

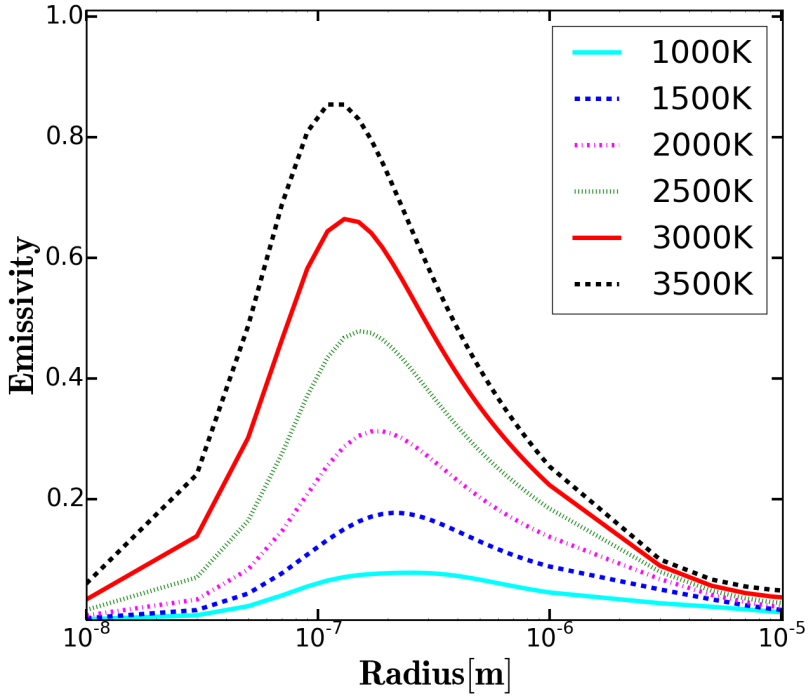



Figure 2.4: Tungsten dust grain emissivities calculated by DUCAD, as a function of the grain radius, for different temperatures.

electrons tunnels into the ion ground state. The excess of energy excites the metal surface and can be picked up by a second metal electron, thus:

$$E_k(e^-) = E_i' - \alpha - \beta \tag{2.52}$$

Where $E_k(e^-)$ is the energy acquired by the second electron, $E_i'$ is the effective ionization energy for the atom at a distance s from the grain, $\alpha$ is the energy below the vacuum level of the electron that tunnels into the atom and $\beta$ is the energy below the vacuum level of the electron that picks up the extra energy in the metal. Overall, for a metal with n electrons the process can be described as:

$$X^+ + ne_M^- \rightarrow X + e^- + (n-2)e_M^- \tag{2.53}$$

And the energy of the final state lies within a range of energies corresponding to all the possible initial $\alpha$ and $\beta$ for the two electrons in the solid. The effective ionization energy $E_i'$ differs from the original ionization energy $E_i'$ due to interaction between the ion and the metal originally and the atom with the metal once the process has occurred and can be described by:

$$E_i'(s) = E_i + E(i-M) - E(n-M) \tag{2.54}$$

Where $E(i-M)$ is the energy of interaction between the neutral atom and the metal and $E(n-M)$ is the energy of interaction of the ion with the

metal. These terms can be derived by assuming that the interaction is produced mainly due to the Coulomb image force for ions, the Van der Waals attraction resulting from the particle's polarizability and the repulsion from the overlap of the electron clouds of the ion/atom and the metal. Following the derivation and assumptions in **[12]**, the maximal transferred energy can be expressed as:

$$E_k^{max}(e^-) = E_i - \frac{e^2}{4z_m} - W_f \tag{2.55}$$

Where $z_m \sim 3.5a_0$ (with $a_0$ being Bohr radius) so $e^2/4z_m \sim 2eV$. On the other hand, the other important process is the resonant neutralization. In this case, an electron from the conduction band of the metal tunnels to a discrete state in the atom; this process occurs without a release of energy, however, the final atom is metastable and may decay via direct Auger de-excitation by which the excited electron occupies a lower state of energy in the atom and the released energy excites the metal. In the process, the released energy is similar to the Auger neutralization process. Following the line of thought of the MIGRAINe code, DUCAD also considers that no electron emission occurs when the metal is excited, however, in order to assure a complete description of these phenomena, this feature might be included in the future along with a precise computation of the electron yields. The heating flux associated with ion surface neutralization can then be expressed as:

$$Q_{isn} = \frac{I_i}{Ze} E_k^{max}(e^-) \tag{2.56}$$

### 2.4.5 Vaporization

Vaporization in dust grains acts as a cooling and mass-loss mechanism. The loss of mass will be analytically introduced in next section and it is given by the Hertz-Knudsen formula. The heat flux can be calculated from this mass defect times the specific enthalpy of the grain:

$$Q_{vap}^{sol,mel} = \frac{dM_d}{dt}(h_d + \Delta h_{sub}) \tag{2.57}$$

$$Q_{vap}^{liq} = \frac{dM_d}{dt}(h_d + \Delta h_{vap}) \tag{2.58}$$

Where $Q_{vap}^{sol,mel}$ is the heat flux for solid/melting grain, $\Delta h_{sub}$ is the specific enthalpy of sublimation, $Q_{vap}^{liq}$ is the heat flux for a liquid grain and $\Delta h_{vap}$ is the specific enthalpy of vaporization, which can be found in Table **??**.

### 2.4.6 Heating numerical scheme

The numerical scheme is based on the following three steps:

1. First, the initial enthalpy of the dust grain is calculated from the plasma parameters and the initial dust temperature.

2. Second, the heat fluxes to the dust grain are evaluated and a finite differences scheme is applied in order to find the new enthalpy.

3. Finally, the new dust temperature is computed by using the enthalpy/temperature correlation used in the first step but in reverse order.

The finite differences scheme for the differential equation is implemented as:

$$\frac{H_d^{n+1} - H_d^n}{\Delta t} = Q_{tot} \rightarrow H_d^{n+1} = H_d^n + Q_{tot} * \Delta t \tag{2.59}$$

Where $n$ is the time step.
The heat fluxes in Figure **2.5** and **2.6** are a priori similar to the calculation performed in the Migraine code [**12**]. We can see that when the dust grain achieves a positive potential at around $T_d \sim 3000K$ the electron current greatly increases which favors an increment in the grain heating rate.

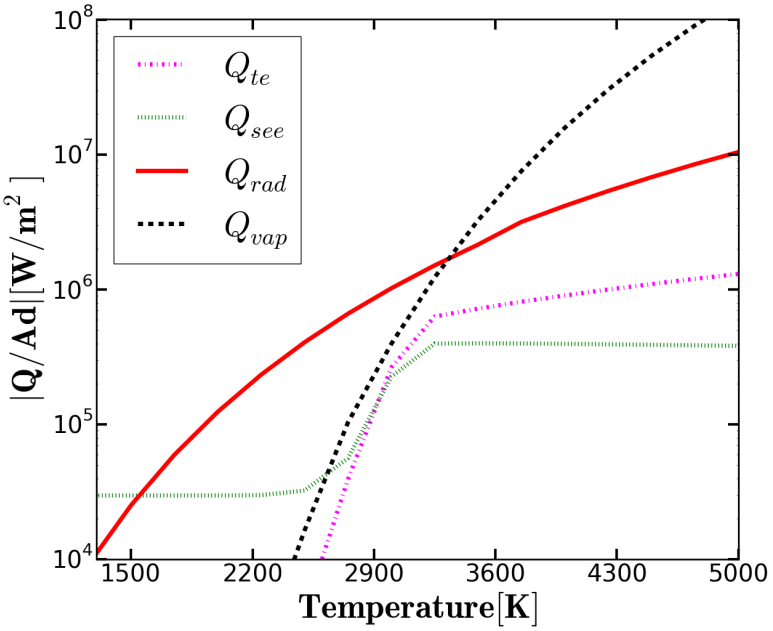



Figure 2.5: Thermionic, Secondary Electron emission, Radiation and Vaporization cooling fluxes calculated by DUCAD for $R_d = 1\mu m$, $T_e = T_i = 18.7eV$, $n_e = n_i = 10^{18}m^{-3}$.

## 2.5 Dust mass evaluation

Strong heat fluxes as those encountered in the core and divertor region of a tokamak may induce a change of phase in dust grains. Therefore, grains evaporate and sublimate under certain conditions, which in the end reduces

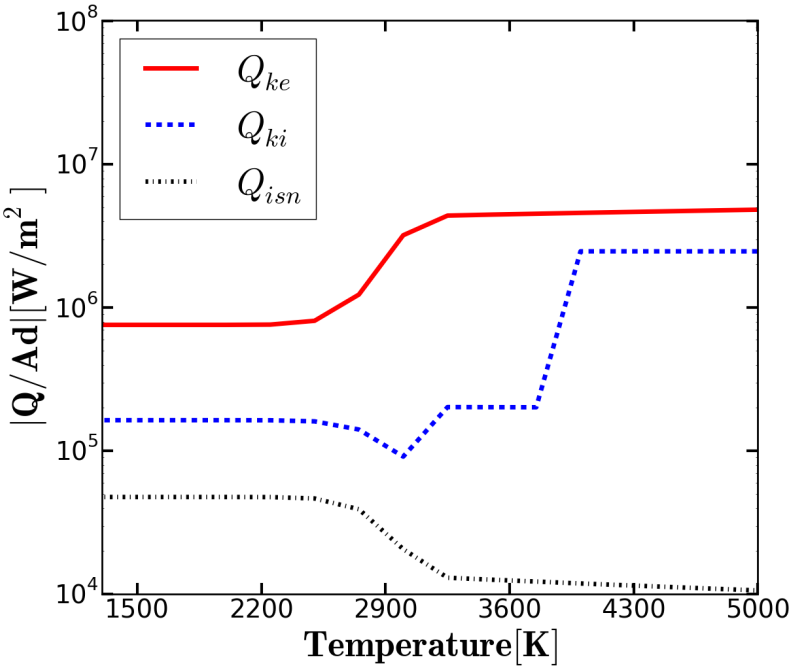



Figure 2.6: Electron, ion and ion surface neutralization heating fluxes calculated by DUCAD for $R_d = 1\mu m$, $T_e = T_i = 18.7eV$, $n_e = n_i = 10^{18}m^{-3}$.

the size of the grain. The radius of dust is an important parameter in the modeling of trajectories as well as for the calculation of the emissivities and heat fluxes, therefore it is crucial to assess with precision the mass loss due to ablation of the grains. The differential equation that models this mass change can be written as:

$$\frac{dM_d}{dt} = \frac{dM_d^{vap}}{dt} + \frac{dM_d^a}{dt} \tag{2.60}$$

Where $dM_d^{vap}/dt$ is the mass lost due to vaporization and $dM_d^a/dt$ is the mass lost/gained due to other mechanisms such as sputtering, collisions with plasma facing components, built up stresses within the dust, backscattering of ions of the same species as the grain's material, ion implantation, etc.

### 2.5.1 Mass loss due to evaporation

The maximum vaporized mass per unit time can be calculated using the Hertz-Knudsen formula:

$$\frac{dM_d}{dt} = -A_d\sqrt{\frac{m_{ad}}{2\pi T_d}}P_v(T_d) \tag{2.61}$$

Where $m_{ad}$ is the atomic mass of the dust material and $P_v(T_d)$ is the vapor pressure. The last term can be evaluated as a function of the dust temperature and is available in bibliography **[34, 35]**, (see Figure **2.7**).

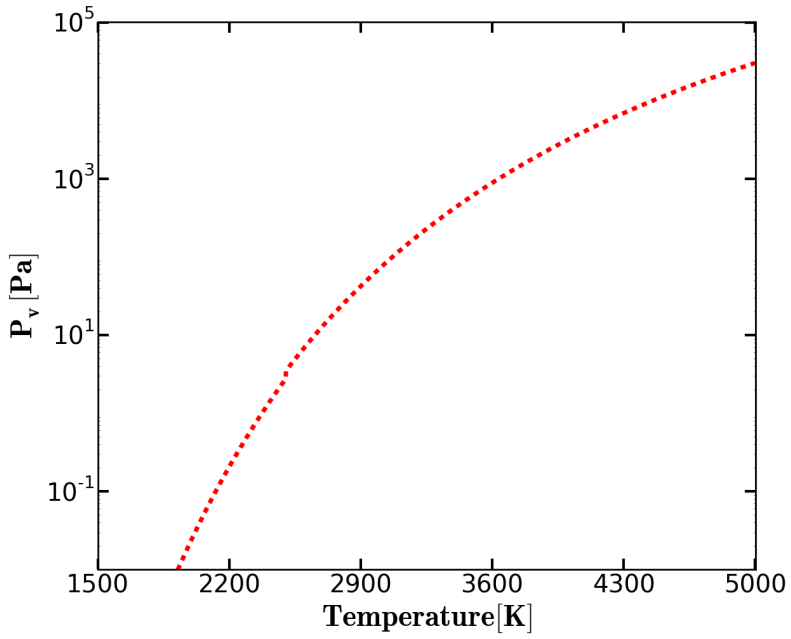



Figure 2.7: Vapor pressure for W as a function of its temperature.

### 2.5.2 Mass-loss numerical scheme

The mass defect is evaluated by a finite differences scheme each time step, meaning:

$$\frac{M_d^{n+1} - M_d^n}{\Delta t} = \frac{dM_d^{vap}}{dt} \rightarrow M_d^{n+1} = M_d^n + \Delta t \left( \frac{dM_d^{vap}}{dt} \right) \tag{2.62}$$

Where $n$ is the time step. By evaluating $M_d/(M_d^{vap}/dt)$ we can estimate the evaporation times for a dust grain, yielding:
So far dust evaporation is the only mass-loss mechanism incorporated into DUCAD. Sputtering, ion implantation, ion backscattering and other mechanisms will be added in the future for a more precise calculation.

## 2.6 Forces and dynamics

Dust grains are under constant bombardment of plasma species which transfer momentum and lead to arising forces. The modeling of these forces is an essential component in order to be able to determine accurately the trajectory a dust grain will follow, and therefore, it allows us to make predictions of wether a grain will reach the core and potentially cause a disruption or it will be ablated instead.

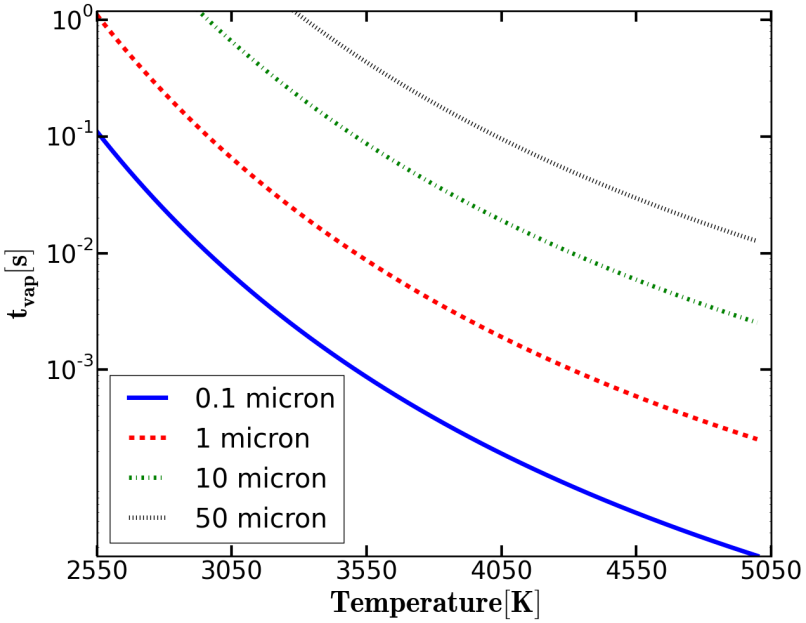



Figure 2.8: DUCAD Calculation of evaporation times for a Tungsten $R_d = 0.1\mu m$, $R_d = 1\mu m$, $R_d = 10\mu m$ and $R_d = 50\mu m$ dust grain.

### 2.6.1 Collection and scattering of plasma species

Ions may be collected as discussed in previous sections or scattered in Coulomb binary collisions by the dust grain. Ion collection transfers momentum to the grain and gives rise to a force, namely the ion collection force, which can be approximated by using the OML expression [**19**]:

$$\vec{F_{i,ab}} = \pi R_d^2 m_i n_i v_{ti} (\vec{u_i} - \vec{V_d}) \frac{1}{2\tilde{u_i}} \left( \frac{1}{\sqrt{\pi}} (1 + 2\tilde{w_{im}}) exp(-\tilde{u_i}^2) + \tilde{u_i} \left[ 1 + 2\tilde{w_{im}} - \frac{1}{2\tilde{u_i}^2} \right] erf(\tilde{u_i}) \right) \quad for \quad \phi_d \leq 0 \quad (2.63)$$

$$\vec{F_{i,ab}} = \pi R_d^2 m_i n_i v_{ti} (\vec{u_i} - \vec{V_d}) \frac{1}{4\tilde{u_i}} \left( \frac{1}{\sqrt{\pi}} \left( \left[ 1 + 2\tilde{u_i}^2 + \frac{1 - 2\tilde{u_i}^2}{\tilde{u_i}} \sqrt{\frac{e\phi_d}{T_i}} \right] exp(-\tilde{u_{ip}}^2) + \left[ 1 + 2\tilde{u_i}^2 - \frac{1 - 2\tilde{u_i}^2}{\tilde{u_i}} \sqrt{\frac{e\phi_d}{T_i}} \right] exp(-\tilde{u_{im}}^2) \right) + \tilde{u_i} \left[ 1 + 2\tilde{w_{im}}^2 - \frac{1 - 2\tilde{w_{ip}}}{2\tilde{u_i}^2} \right] [erf(\tilde{u_{ip}}) + erf(\tilde{u_{im}})] \right) \quad for \quad \phi_d > 0 \quad (2.64)$$

Where $\tilde{w_{ip}} = \tilde{u_i}^2 + Ze\phi_d/T_i$ and $\tilde{w_{im}} = \tilde{u_i}^2 - Ze\phi_d/T_i$. Ion scattering is produced by inelastic Coulomb collisions in which the grain is considered to

be an unshielded charge since $\lambda_e/R_d > 1$. This collisions lead to a diffusion of momentum both for the ions and the grain. The force acting upon the grain for this case reads:

$$\vec{F_{i,sc}} = 2\pi R_d^2 m_i n_i v_{ti}(\vec{u_i} - \vec{V_d}) \left(\frac{Ze\phi}{T_i}\right) \frac{G(\tilde{u_i})}{\tilde{u_i}} ln\Lambda_d \tag{2.65}$$

For both positive and negative dust potential, where:

$$G(x) = \frac{(erf(x) - 2x \cdot exp(-x^2)/\sqrt{\pi})}{2x^2} \tag{2.66}$$

is the Chandrasekhar function and $ln\Lambda_d$ is the Coulomb logarithm or the ratio between the upper and lower cut-off for the impact parameter. When a particle is involved in a Coulomb collision, it is deflected depending upon its proximity to the other particle, namely its impact parameter, the velocity of the particle and its charge. The determination of the Coulomb logarithm is a matter of discussion and several theoretical approaches have been taken for the case of ion collisions with dust particles. The analytical formula which presents a better match to numerical calculations is the Khrapak form:

$$ln\Lambda_d = ln\left[\frac{b_{90} + \lambda_s}{b_{90} + R_d}\right] \tag{2.67}$$

Where $b_{90}$ is the 90$^o$ scattering impact parameter for Coulomb collisions and $\lambda_s$ is the normalized Debye length that comprises the electron Debye length and the ion Debye length. However, Khrapak expression does not give a complete representation of reality and suitable fits have been made by the PIC code SCEPTIP [**36**] to achieve a better compromise, yielding for the parameters in the logarithm:

$$b_{90} = \frac{R_d Ze|\phi_d|}{m_i v_{eff}^2} \tag{2.68}$$

$$\lambda_s^2 = \frac{\lambda_e^2}{(1 + 2ZT_e/m_i v_{eff}^2)} + R_d^2 \tag{2.69}$$

Where the effective velocity is represented as:

$$v_{eff}^2 = \frac{2T_i}{m_i} + v_f^2 \left[1 + \left(\frac{|v_f|/\sqrt{2ZT_e/m_i}}{0.6 + 0.05ln(m_i/m_pZ) + (\lambda_e/5R_d)(\sqrt{T_i/ZT_e} - 0.1)}\right)^3\right] \tag{2.70}$$

And where $v_f$ is the ion flow velocity which equals the ion thermal velocity times the Mach number, $v_f = Mv_{ti}$. Forces arising from electron absorption and scattering can be safely neglected since electron mass is comparably low.

### 2.6.2 Macroscopic fields: $\vec{E}$ and $\vec{g}$

Dust grains experience also forces due to macroscopic fields, the most direct influence being the gravitational field:

$$\vec{F_g} = M_d \vec{g} \tag{2.71}$$

Where $\vec{g} \sim 9.81 m/s^2$ is the gravitational acceleration vector. Since all the phenomena affecting the dust grains are coupled, as the mass varies due to heating and vaporization, so does the gravitational interaction. Electrical forces may also play an important role in dust dynamics and since data is available for the plasma potential, DUCAD incorporates them. The local electric field is computed as a gradient of the potential from the cell the grain occupies to the adjacent cells:

$$\vec{E} = -\nabla \phi_{plasma} = -\left[ \frac{\phi_p^{r_1} - \phi_p^{r-1}}{\Delta r} \vec{u_r} + \frac{\phi_p^{\theta_1} - \phi_p^{\theta-1}}{\Delta \theta} \vec{u_\theta} + \frac{\phi_p^{\Phi_1} - \phi_p^{\Phi-1}}{\Delta \Phi} \vec{u_\Phi} \right] \tag{2.72}$$

Where $\phi_p^{r_1}$ is the bigger radial value of the plasma potential in either the two radial adjacent cells and $\phi_p^{r-1}$ is the smaller radial value of the plasma potential correspondingly, $\phi_p^{\theta_1}$ is the bigger poloidal value of the plasma potential in either the two poloidal adjacent cells and $\phi_p^{\theta-1}$ is the smaller poloidal value of the plasma potential correspondingly, $\phi_p^{\Phi_1}$ is the bigger toroidal value of the plasma potential in either the two toroidal adjacent cells and $\phi_p^{\Phi-1}$ is the smaller toroidal value of the plasma potential as well, $\Delta r$ is the module of the distance between the two radial adjacent cells, $\Delta \theta$ is the module of the distance between the two poloidal adjacent cells and $\Delta \Phi$ is the module of the distance between the two toroidal adjacent cells. The reason for this bigger/smaller order is to preserve the correct directionality of the field vector. The electric force resulting can be simply computed as:

$$\vec{F_e} = q_d \vec{E} = Z_0 e \vec{E} \tag{2.73}$$

Again as the electric charge depends upon the floating potential condition, as the grain travels to different parts of the reactor and the environmental conditions such as temperature and density change, the force can attain significant variations.

### 2.6.3 Neutral friction force

High densities of neutral atoms can be found in regions where plasma ions impinge on the reactor walls and get consequently neutralized, e.g. the divertor plates. Neutrals also transfer their momentum to dust grains upon collisions. If we consider that the neutrals give up all their momentum to the grain in the collision process, we can use equation **2.63** by taking $\phi_d = 0$, since neutrals

are not affected by the dust sheath, to express the resultant force [**19**]:

$$\vec{F_n} = \pi R_d^2 m_n n_n v_{tn} (\vec{u_n} - \vec{V_d}) \left( \frac{1}{\sqrt{\pi}} \left[ 1 + \frac{1}{2\tilde{u_n}^2} \right] exp(-\tilde{u_n}^2) + \right.$$
$$\left. \frac{1}{\tilde{u_n}} \left[ 1 + \tilde{u_n} - \frac{1}{4\tilde{u_n}^2} \right] erf(\tilde{u_n}) \right) \quad (2.74)$$

Where $m_n$ is the atomic mass of the neutral species, $n_n$ is the density of neutrals, $v_{tn}$ is the neutral thermal speed, $\vec{u_n}$ is the neutral flow velocity and $\tilde{u_n} = |\vec{u_n} - \vec{V_d}|/v_{tn}$ is the normalized neutral flow velocity with respect to the dust grain.

### 2.6.4 Rocket force

Even within the assumption of spherical dust particles, one of the sphere's side is exposed to the ion and electron fluxes meanwhile the other side is not. This situation generates different temperatures in both sides of the sphere and therefore, different vaporization rates. The thrust from the vaporized atoms can result in a force, namely the rocket force. DUCAD includes a really simple scheme based on restrictively assumptions in order to implement the rocket force without the computational cost of a more sophisticated method. Since the heating of a sphere is not instantaneous but rather, it is a dynamical process, (which is the main reason why $dH/dt$ must be computed each time step) the heat fluxes to the grain are those determined by the ambient parameters whereas the cooling fluxes are determined by the initial dust grain temperature. This means that the heating is "always one step ahead" the cooling in a numerical calculation set as explained in section **2.4.6**, because is not until the new temperature is computed that the cooling processes "catch up". Of course, this will be improved in future developments of the code by, for example, switching to a leap-frogging scheme.
With this in mind, the heat balance presented in previous sections, is used to derive the new dust temperature which is attributed only to the exposed half of the sphere, the one in direct contact with the plasma stream meanwhile the other half of the sphere preserves its temperature as if it was not affected by any heat flux during the whole process, which is a good approximation as long as the time step is kept reasonably small. This procedure differs from the one used by the authors in [**37**] where it is consider that the exposed surface is in thermal equilibrium with the impinging plasma heat flux. Strictly speaking, it can be shown by DUCAD numerical simulations (see Figure **2.9**) that the effective time for achieving thermal equilibrium can often be of the order of some miliseconds meaning that for fast particles $Vd \sim 10 - 100 m/s$ the dust grain hardly approaches thermal equilibrium before migrating to a different part of the reactor vessel with different plasma parameters and hence, different equilibrium temperature. Therefore, the implemented scheme in DUCAD seems a better approximation:

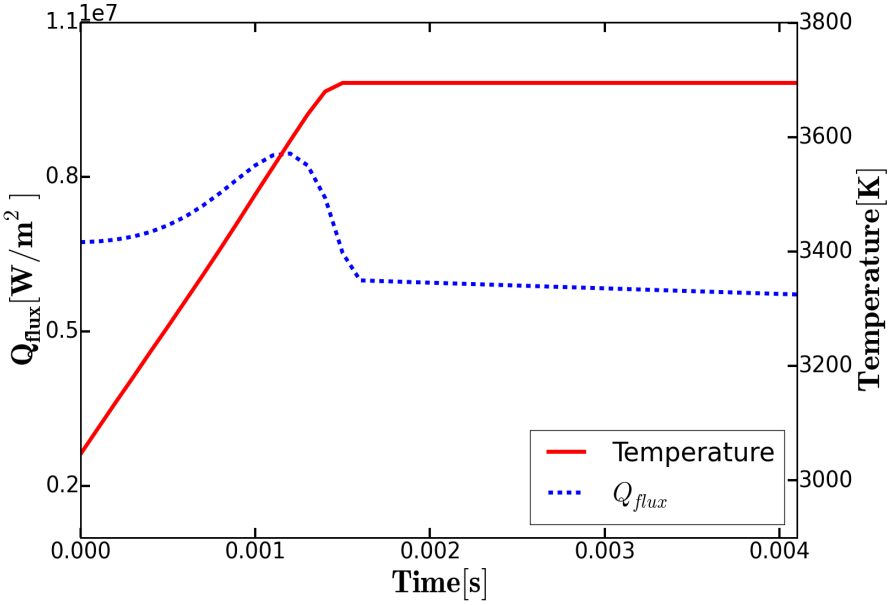



Figure 2.9: DUCAD Calculation of the dust grain temperature according to the heat flux acting on it, with radius $R_d = 10\mu m$ and plasma conditions $T_e = T_i \sim 5eV$, $n_e = n_i \sim 5 * 10^{19}$ and Mach number $M \sim -0.4$. Dust initial temperature is $T_0 = 3000K$.

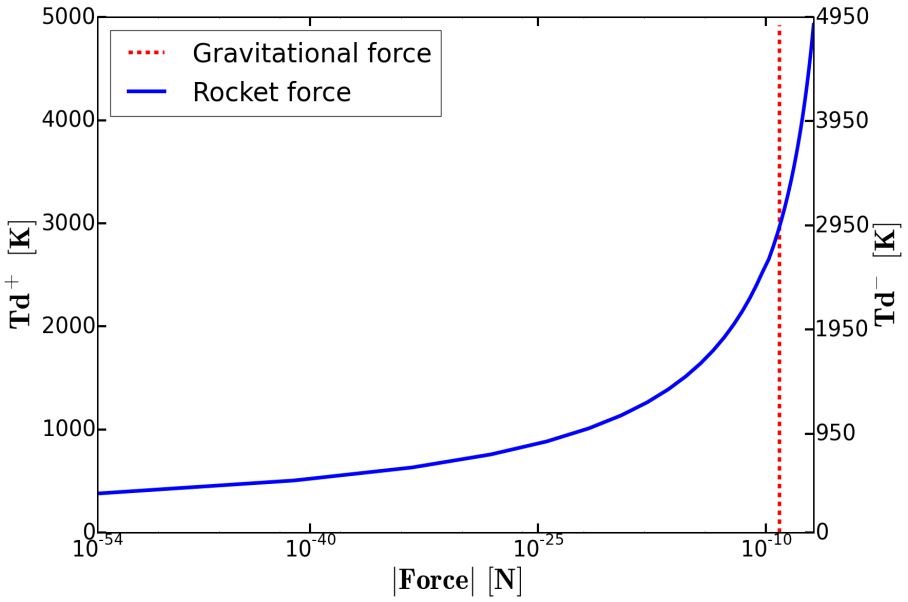



Figure 2.10: DUCAD Calculation of the rocket force acting on a tungsten grain with radius $R_d = 10\mu m$ and a temperature difference of $\Delta T$=50K and gravitational force plotted to serve as a reference of the magnitude.

$$\begin{cases} dH/dt = P_{tot} = P_{heat} - P_{loss} \quad for\ T^{+}_{n+1}\ \ calculation \\ dH/dt = 0 \rightarrow T^{-}_{n+1} = T^{+}_{n} \quad for\ T^{-}_{n+1}\ \ calculation \end{cases} \tag{2.75}$$

Where n is the time step.
Overall, this allows the definition of two temperatures $T^+$ and $T^-$ for the two sides of the sphere. Following the work presented in [**38**], the rocket force can be written as:

$$\vec{F_{roc}} = M_d \frac{3}{4\sqrt{2\pi}\rho_a} \frac{P_{vap}(T^+) - P_{vap}(T^-)}{R_d} \vec{u}_{\vec{B}} \tag{2.76}$$

Where $\rho_a$ is the density of the element $a$, $M_d$ is the grain's mass, $R_d$ is the grain's radius, $P_{vap}(T^+)$ is the vapor pressure at the exposed grain surface, $P_{vap}(T^-)$ is the vapor pressure at the shadowed surface and $\vec{u}_{\vec{B}}$ is the unitary vector along the magnetic field lines.
Finally, it is further assumed that from one time step to the other, the temperature in the sphere is homogenized by heat conductivity along the dust grain. The direction of the rocket force follows the ion and electron flow (which is the magnetic field direction taking into account the Mach number sign) since the exposed half of the sphere is always in contact with the plasma flow and being this half the one that "pushes" the grain through thrust generation, the dust moves in opposite direction, meaning, following the flow.

### 2.6.5 Forces numerical scheme

Forces acting upon the dust grain are calculated following previous indicated scheme and summed into $\vec{F_{tot}}$. Basic dynamical equations in the form of finite differences are used to anticipate the trajectories:

$$M_d \frac{d\vec{V_d}}{dt} = \vec{F_{tot}} \rightarrow V_d^{n+1} = \frac{\vec{F_{tot}^n} \Delta t}{M_d} + V_d^n \tag{2.77}$$

$$\frac{d\vec{R_d}}{dt} = \vec{V_d} \rightarrow R_d^{n+1} = V_d^n \Delta t + R_d^n \tag{2.78}$$

In addition to the already discussed phenomena, magnetic field effects can be neglected on dust dynamics. For micrometer size grains we find that $R_d < \lambda_D \sim \rho_e \ll \rho_D$, where $R_d$ is the dust grain radius, $\lambda_D$ is the dust Debye length, $\rho_e$ is the electron gyroradius and $\rho_D$ is the grain gyroradius. For instance, for a magnetic field $B \sim 1T$, an electron temperature of $T_e \sim 20eV$, an electron density of $n_e \sim 10^{18} m^{-3}$, a tungsten grain of radius $R_d \sim 1\mu m$, a grain perpendicular velocity of $v_\perp \sim 1m/s$ and considering $v_\perp^{e^-} \sim v_{th}^{e^-}$, we find $R_d = 1\mu m \ll \rho_e = 10\mu m \sim \lambda_e = 33\mu m \ll \rho_D \sim 12m$ by following the probe theory approximation $e\phi_D \sim 3T_e$. In this regime, Lorentz force and other magnetic effects are marginal since the dust gyro frequency is $\omega_D = v_\perp/\rho_D \sim 0.08 rad/s$ which is considerably small [**39**].

| Parameter | Value (units) | Reference |
|---|---|---|
| Maximum SEE yield $\delta_m$ | 0.93 | [**12**] |
| Work function Wf | 4.91(eV) | [**40**] |
| Energy at maximum yield $E_m$ | 600 (eV) | [**12**] |
| Correction factor $f$ | 1/0.49 | [**28**] |
| Fitting parameter $k$ | 1.38 | [**23**] |
| Enthalphy of sublimation $\Delta h_{sub}$ | 859.90(kJ/mol) | [**34**] |
| Angular dependence exponent $\beta$ | 0.8 | [**24**] |
| Enthalpy of vaporization $\Delta h_{vap}$ | 806(kJ/mol) | [**12**] |
| Density $\rho_W$ | 19300 ($kg/m^3$) | [**35**] |
| Melting Temperature $T_W$ | 3695 (K) | [**35**] |

Table 2.1: Tungsten parameters

# Chapter 3

# Numerical study: Dust in Asdex Upgrade Tokamak

## 3.1 EMC3/EIRENE Monte Carlo code

The EMC3/EIRENE is a transport code capable of solving the fluid equations for mass, momentum and energy of plasma ions, electrons and neutrals. It is based on a Monte Carlo solver which allows to rewrite the fluid equations into a conductive-convective shape and approximate the spatial/temporal step by assuming a stochastic transport term in the local transition probability function [**41**]. A poloidal slice of the grid is presented in Figure **3.1**

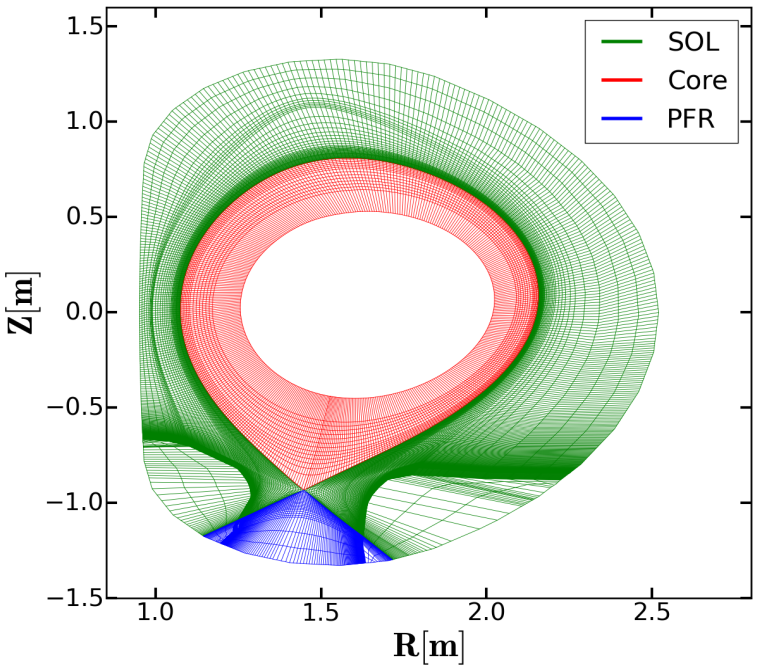



Figure 3.1: Poloidal Slice of the EMC3/EIRENE grid.

Lately it has also incorporated the effects of the main chamber plasma facing components and divertor plates into its simulations [**42**]. The computational grid from EMC3/EIRENE is separated into 24 zones representing the SOL, the private flux region and the core of the 8 sectors toroidally distributed in the upper half of the Asdex Upgrade torus. Each zone consists in a 3-dimensional subgrid with $N_{ir} \times N_{ip} \times N_{it}$ knots forming $(N_{ir}-1) \times (N_{ip}-1) \times (N_{it}-1)$ cells able to store data for electron density $n_e$, electron temperature $T_e$, ion temperature $T_i$, plasma potential $\phi_{plasma}$, neutral density $n_H$, Mach number $M$ as well as the coordinates of the knots that comprise the grid. Furthermore, the grid is aligned with the magnetic field lines such that the vector that goes from the position $(R_{ir,ip,it}, \phi_{ir,ip,it}, z_{ir,ip,it}) \rightarrow (R_{ir,ip,it+1}, \phi_{ir,ip,it+1}, z_{ir,ip,it+1})$ points in the magnetic field direction. Reconstruction of shot nº 29464 at $t = 2500ms$ by EMC3/EIRENE will be used as input plasma parameters for DUCAD dust simulations in this section. Short nº 29464 was set in L-mode with Deuterium gas and a toroidal magnetic field of $B_t = -2.519T$. The safety factor at 95% of the toroidal flux is $q_{95} = 5.365$, the average electron density was $n_e = 2.16 * 10^{19} m^{-3}$ and the plasma current was $I_p = 0.8MA$. At $t = 2500ms$ only the ECRH was providing external heating power $P_{ECRH} = 0.592MW$ and the current flattop was reached. These parameters can be seen in Figures **3.2** and **3.3**.

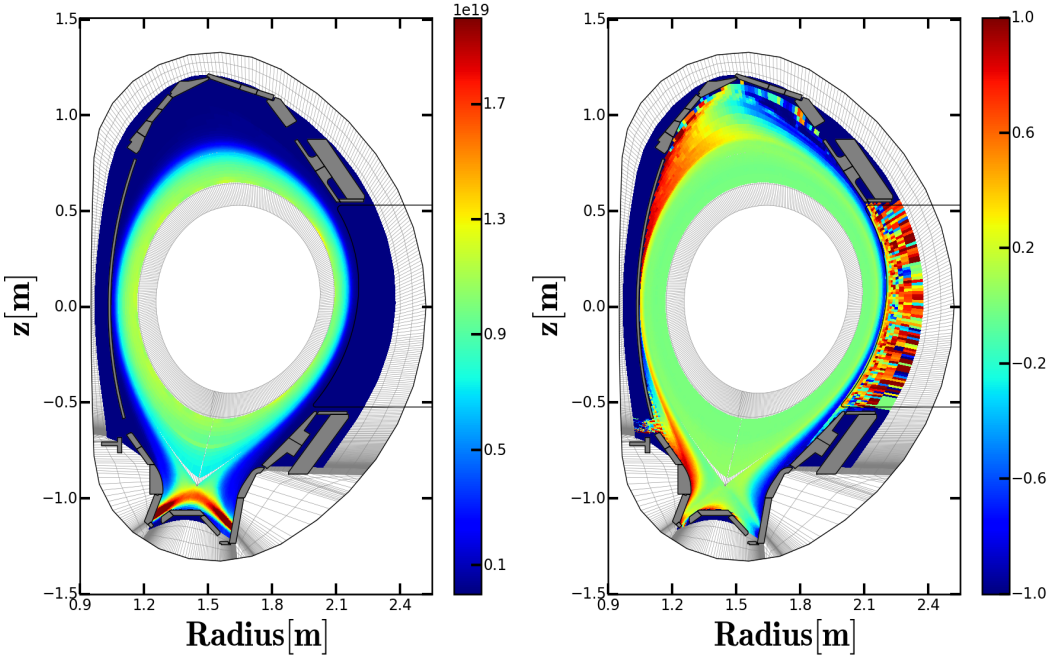



Figure 3.2: Electron density $[m^{-3}]$ (left) and Mach number (right) for shot nº 29464, 2500ms.

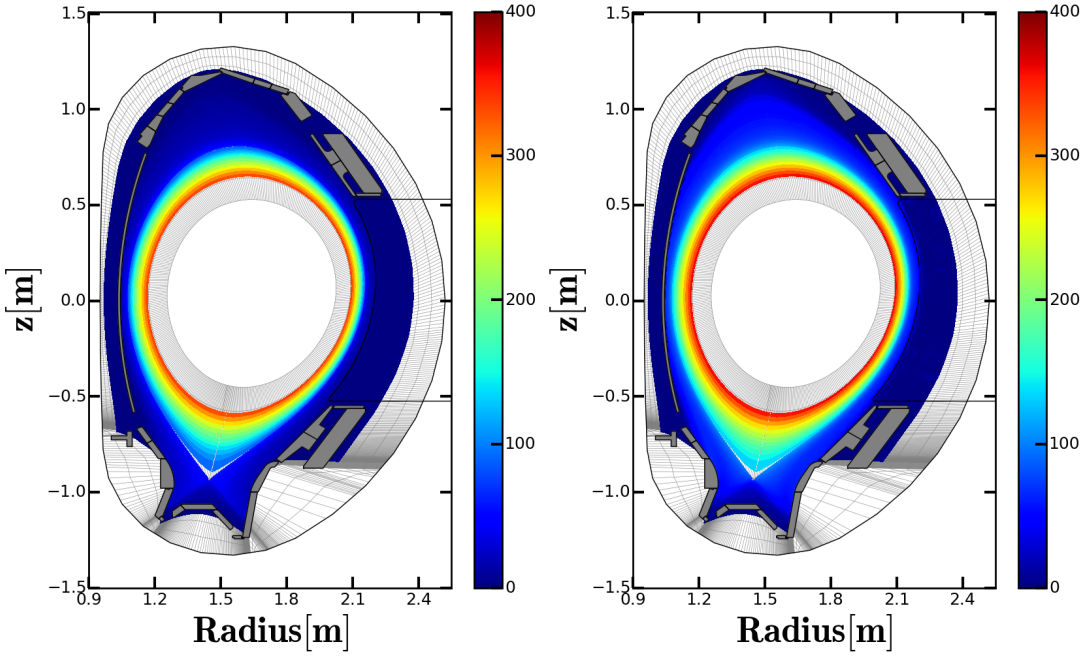



Figure 3.3: Electron temperature $[eV]$ (left) and Ion temperature $[eV]$ (right) for shot nº 29464, 2500ms.

## 3.2 First trajectory: Shot nº 28590

Short nº 28590 was set in H-mode with Deuterium gas. The flat-top plasma parameters were: toroidal magnetic field of $B_t = -1.811T$, safety factor at 95% of the toroidal flux $q_{95} = 3.860$, the average electron density was $n_e = 6.01 * 10^{19} m^{-3}$ and the plasma current was $I_p = 0.8MA$. Both the ECRH and the NBI were providing external heating power. A disruption ends the discharge at $t = 4.997s$. In this case, a trajectory among the several ones registered by a set of two fast cameras will be studied. The trajectory is reconstructed following the TRACE algorithm developed in the Institute Jean Lamour which consists in identifying the bright spots registered by the fast cameras and tracking them over time to differentiate among hot spots, dust particles and noise. It is still unclear how the ablated material from the dust grain affects the detected signal since it contributes with line radiation to the total thermal emissivity from the dust grain and therefore modifies the detected radiation pattern, yet it has been studied for some cases such as carbon dust grains [**43**]. This is the reason why dust size is difficult to characterize just from the recorded trajectories in fast cameras, due to the increased volume that is seen because of the surrounding ablation cloud. The experimental trajectory and DUCAD simulations for two cases $Rd = 0.5\mu m$ and $R_d = 30\mu m$ are presented in Figures **3.4,3.5,3.6** and **3.7**. Initially dust particles are considered to begin their motion at $(R, \phi, z) = (2.06m, 57.38^o, 0.58m)$ with an initial velocity of

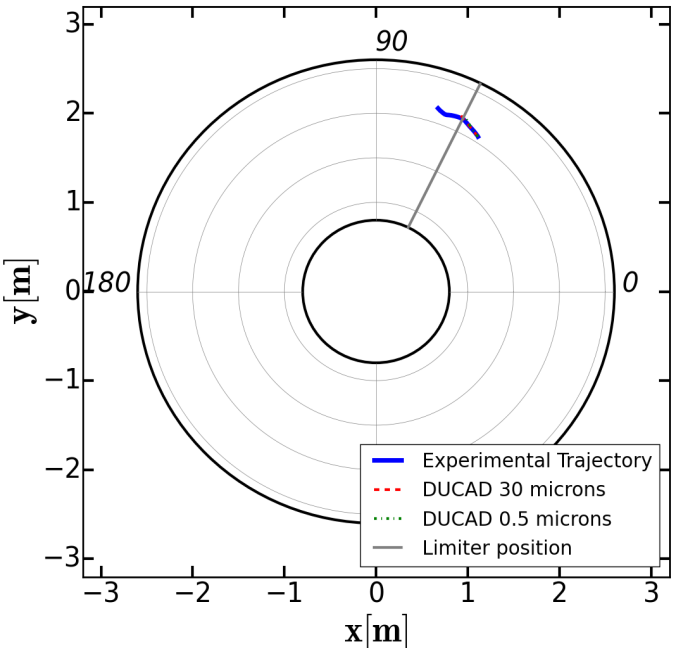



Figure 3.4: Toroidal view. The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations. The grey line indicated the position of the auxiliary limiter.

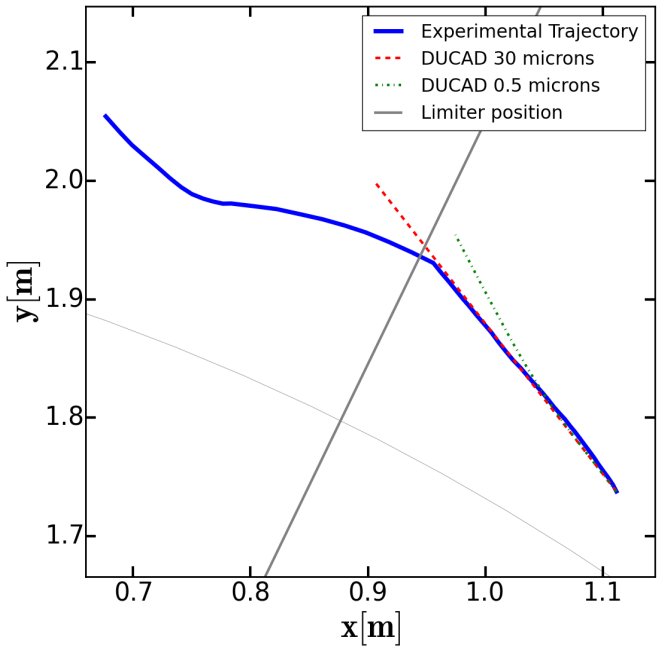



Figure 3.5: Close approach of the toroidal view. The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations. The grey line indicated the position of the auxiliary limiter.

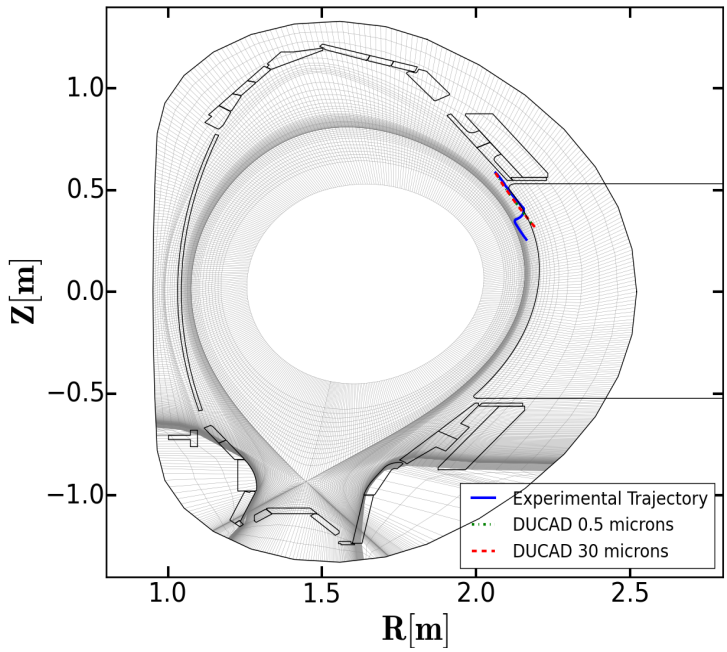



Figure 3.6: Poloidal view of the Asdex Upgrade Tokamak. The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations.

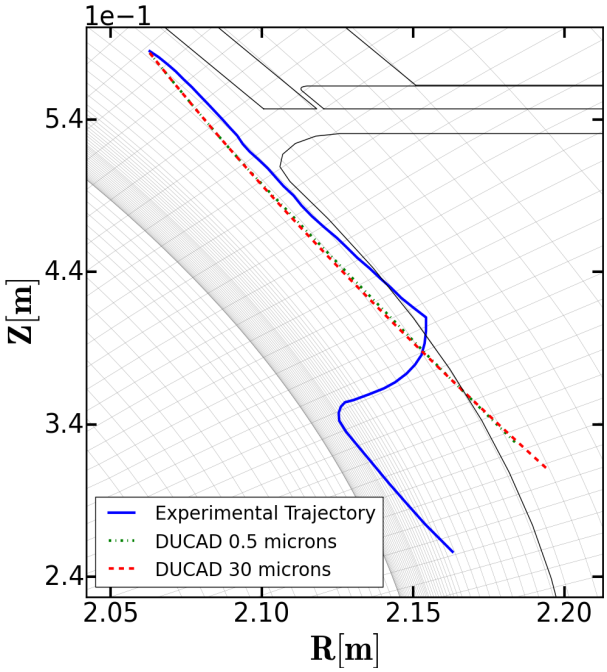



Figure 3.7: Close approach poloidal view of the Asdex Upgrade Tokamak The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations.

$(v_x, v_y, v_z) = (-14.57, 18.48, -19.25) m/s$. It can be observed that DUCAD models correctly the experimental trajectory when it is homogeneous and it even allows to estimate a lower limit for the size of the dust particle since for small particles, drift forces would deviate the followed path as can be seen for the $R_d = 0.5\mu m$ case. It is also seen that the particle makes an abrupt change of trajectory when getting close to the plasma facing components (see Figure **3.7** in detail). One of the Lithium beams in Asdex Upgrade is situated $33cm$ above the mid plane [**44**]. Therefore, it is believed that this collision may be due to a direct encounter with the Lithium Beam limiter situated at the Toroidal angle $\Phi \sim 64^o$ (grey lines in Figures **3.4** and **3.5**). Of course, since a description for dust-wall interactions is still missing within DUCAD, this effect cannot yet be addressed, but it will certainly be studied in future extensions of the code. Dust parameters can be studied for the $R_d = 30\mu m$ case such as the temperature and heat fluxes to the grain (Figure **3.8**), the acceleration that the grain acquires due to the collection of forces (Figure **3.9**) and the size evolution due to ablation processes (Figure **3.10**).

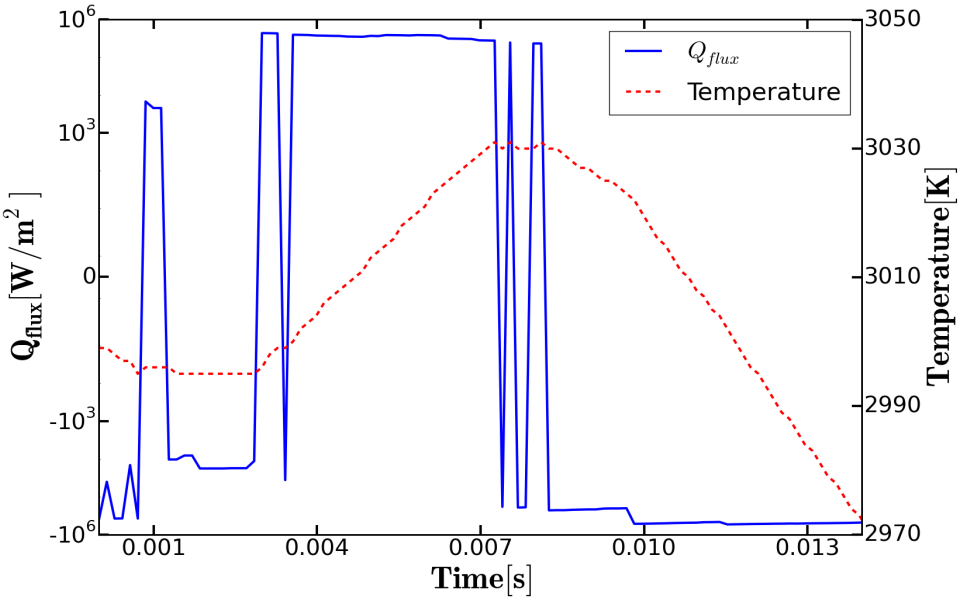



Figure 3.8: Heat flux to the grain and temperature evolution

Negative heat fluxes in Figure **3.8** are found when the dust grain is hotter than the plasma environment and thus the grain emits more power than it receives from the bulk plasma. It is also appreciated that for SOL usual parameters the dust grain rarely experiences a phase transition. Regarding the forces, it is seen that the ion drag force and the gravitational interaction play an important role in dust dynamics, the latter being specially important for the case of big size grains as this $30\mu m$ one. The rocket force arises when there are temperature differences in the grain and hence it drops intermittently to zero as the temperature is from time to time homogenized. Electric forces can play

a main role when approaching the sheath in the plasma facing components as noted in [**45**] but for the most part of the SOL environment its role is minor.

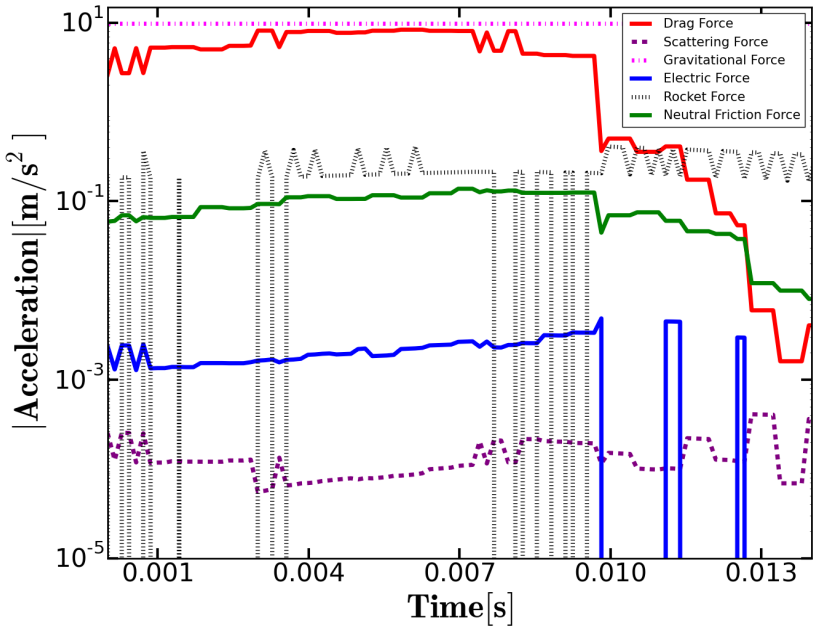



Figure 3.9: Acceleration of the dust grain depending on the force contribution

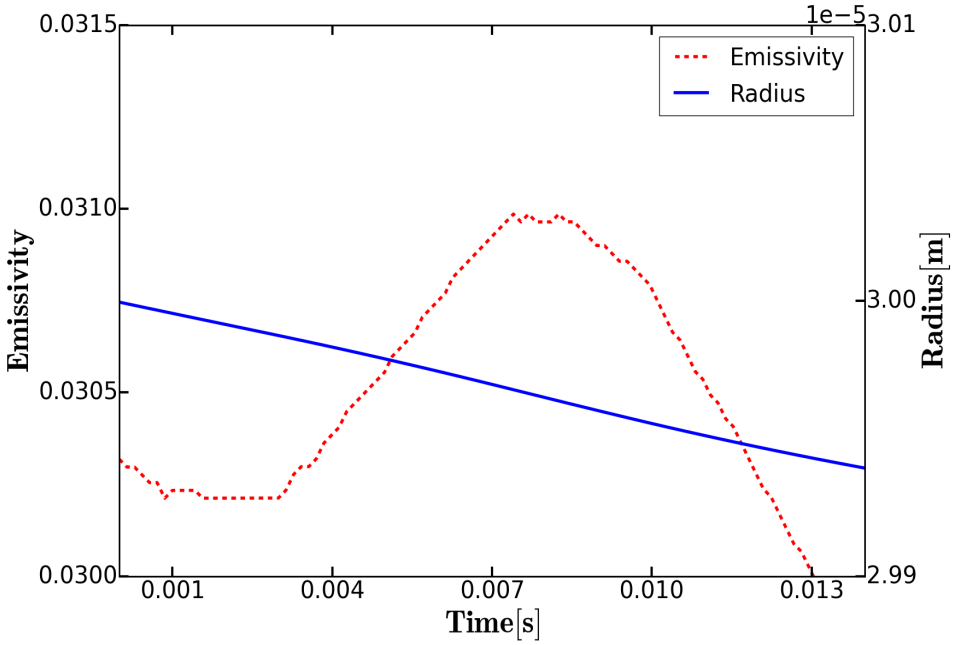



Figure 3.10: Emissivity and Radius evolution of the $R_d = 30\mu m$ dust particle over time.

However it is important to keep in mind that the plasma parameters obtained from the EMC3/EIRENE code belong to an L-mode discharge, meanwhile the reconstructed trajectory, corresponds to an H-mode discharge, meaning that temperature, potential and density gradients as well as absolute values are not perfectly represented, specially near the edge plasma region. It is is also observed that the grain's ablation is small, as was expected from a large size grain (recall Figure **2.8**).

## 3.3 Second trajectory: Shot nº 28695

Short nº 28695 was set in H-mode with Deuterium gas. The flat-top plasma parameters were: toroidal magnetic field of $B_t = -2.424T$, safety factor at 95% of the toroidal flux $q_{95} = 5.330$, the average electron density was $n_e = 7.46 * 10^{19} m^{-3}$ and the plasma current was $I_p = 0.8MA$. Both the ECRH and the NBI were providing external heating power. A disruption ends the discharge at $t = 5.972s$. A registered trajectory is analyzed in this case due to its high velocity, and thus the interest of fast particles crossing the camera and the possible conclusions that can be obtained from them. Initially dust particles are considered to begin their motion at $(R, \phi, z) = (2.10m, 35.12^o, -0.35m)$ with an initial velocity of $(v_x, v_y, v_z) = (-43.41, 78.55, 14.5)m/s$.

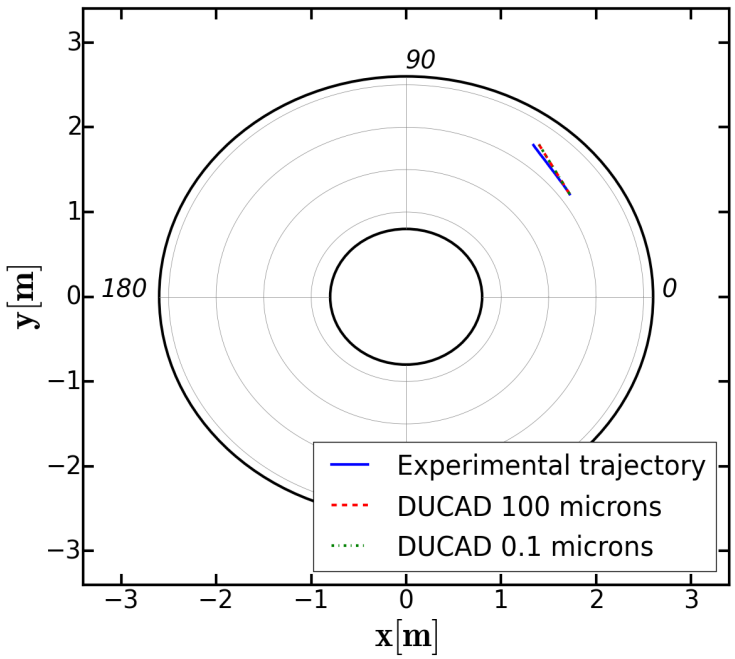



Figure 3.11: Toroidal view. The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations.

As it can be appreciated in Figures **3.11**, **3.12**, **3.13**, **3.14**, the simulated

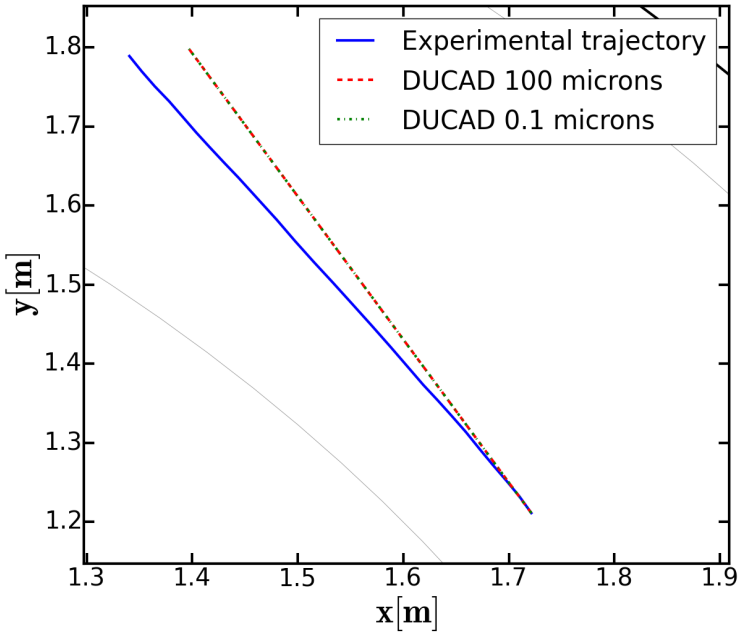



Figure 3.12: Close approach of the toroidal view. The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations.

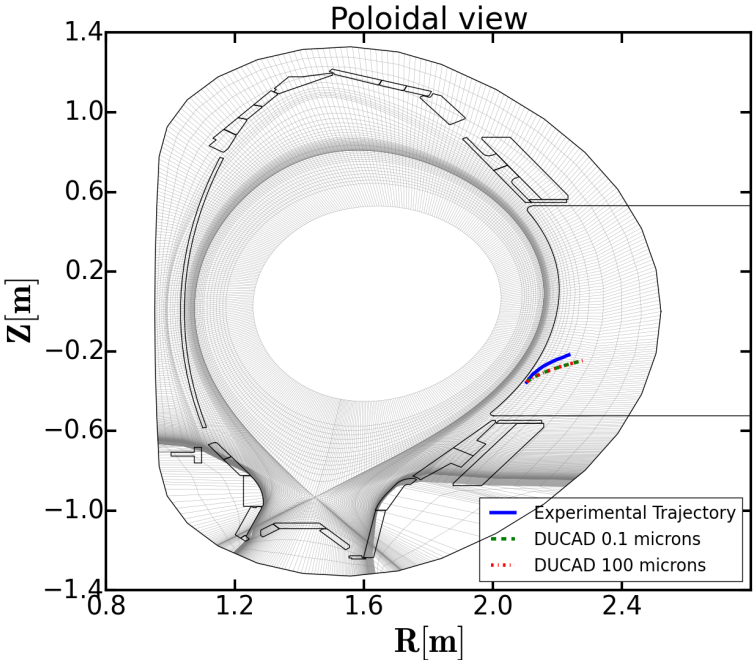



Figure 3.13: Poloidal view of the Asdex Upgrade Tokamak. The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations.

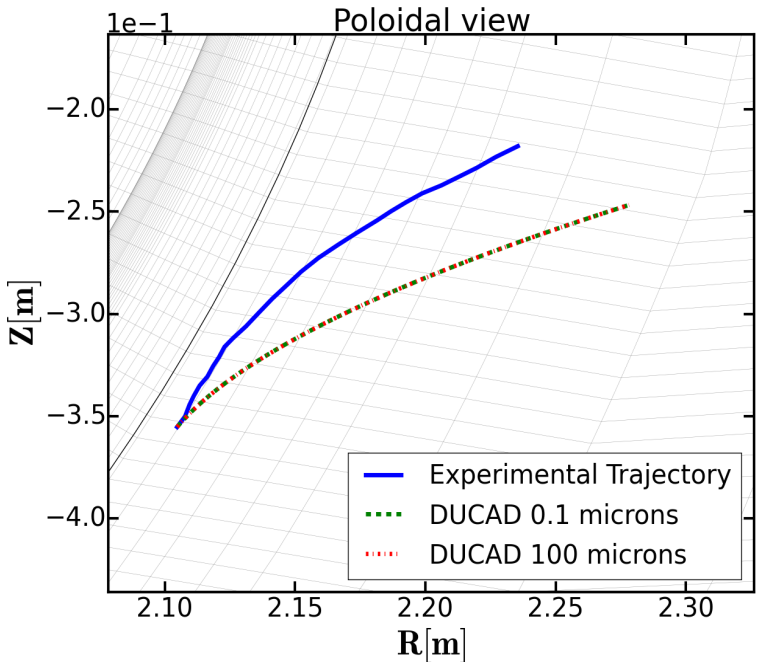



Figure 3.14: Close approach poloidal view of the Asdex Upgrade Tokamak The blue trajectory represents the experimental reconstruction and the dotted red and green lines DUCAD simulations.

trajectories are completely independent of the size election. The reason for this is that since the initial particles have such high velocities, the time they expend in the plasma environment is not enough to appreciably modify their trajectory by means of drag forces. Thus in this limit situation it is not possible to make an assessment on the particle size.

## 3.4 Third trajectory: Shot nº 28642

Short nº 28642 was set in H-mode with Deuterium gas. The flat-top plasma parameters were: toroidal magnetic field of $B_t = -2.444T$, safety factor at 95% of the toroidal flux $q_{95} = 5.342$, the average electron density was $n_e = 9.16 * 10^{19} m^{-3}$ and the plasma current was $I_p = 0.8MA$. The ECRH, the NBI and the ICRH were providing external heating power. A disruption ends the discharge at $t = 7.785s$. In this case a trajectory close to the plasma core is analyzed. Initially dust particles are considered to begin their motion at $(R, \phi, z) = (1.81m, 11.71^o, -0.71m)$ with an initial velocity of $(v_x, v_y, v_z) = (-13.68, 47.95, 25.06)m/s$. Three trajectories were simulated with $R_d = 0.1\mu m$, $R_d = 10\mu m$ and $R_d = 50\mu m$. Observing the Figures **3.15**, **3.16**, **3.17** and **3.18** one can come to the conclusion that since the $R_d = 1\mu m$ grain is ablated before it completes half of the trajectory and since the $R_d = 50\mu m$ grain does not get accelerated far enough toroidally, the ex-

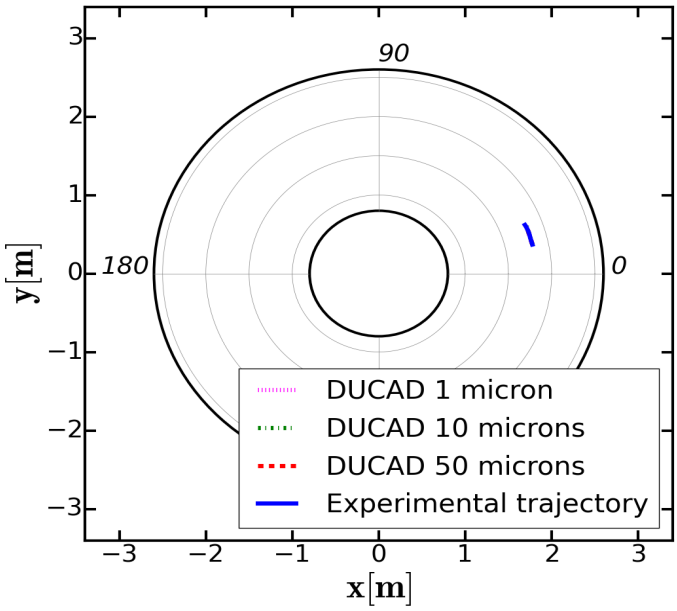



Figure 3.15: Toroidal view. The blue trajectory represents the experimental reconstruction and the dotted red, green and magenta lines DUCAD simulations.

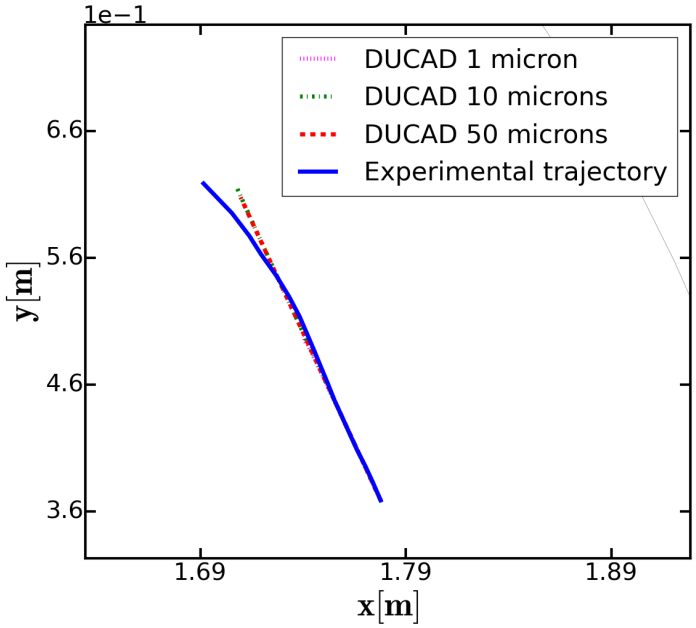



Figure 3.16: Close approach of the toroidal view. The blue trajectory represents the experimental reconstruction and the dotted red, green and magenta lines DUCAD simulations.

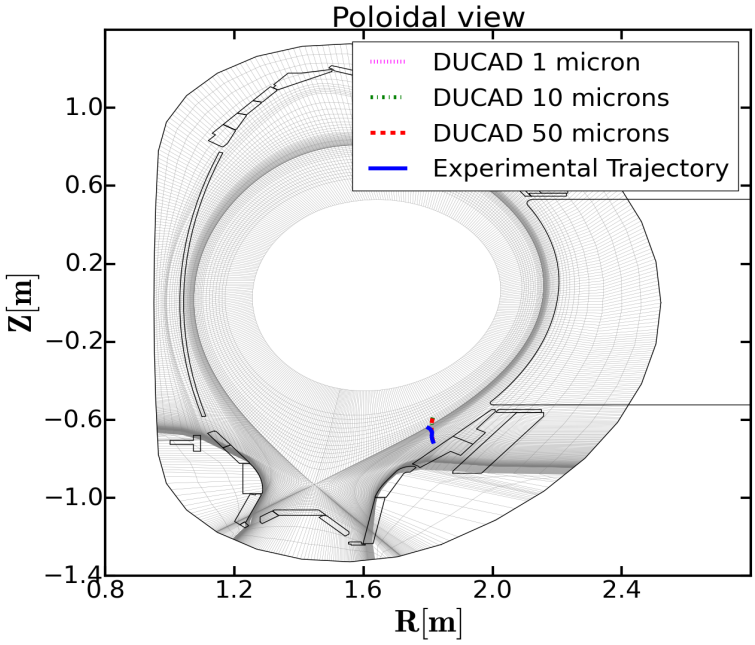



Figure 3.17: Poloidal view of the Asdex Upgrade Tokamak. The blue trajectory represents the experimental reconstruction and the dotted red, green and magenta lines DUCAD simulations.

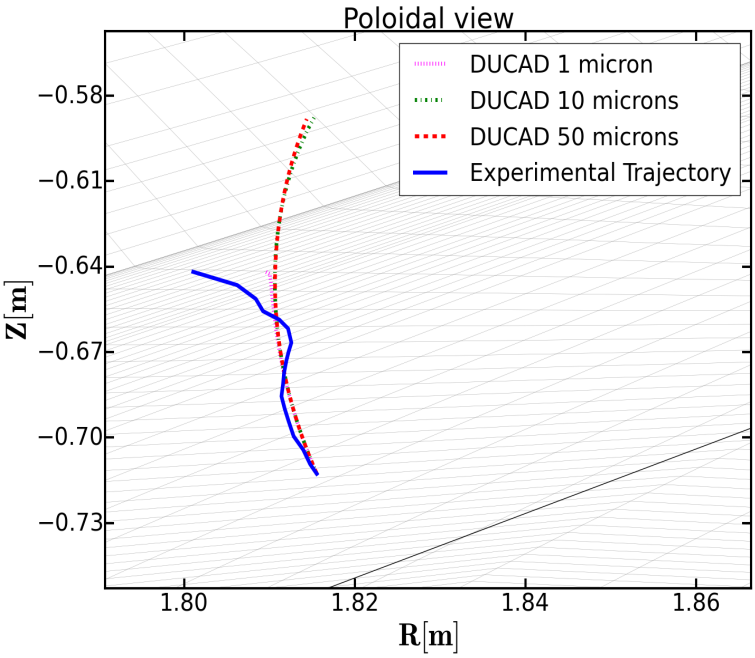



Figure 3.18: Close approach poloidal view of the Asdex Upgrade Tokamak The blue trajectory represents the experimental reconstruction and the dotted red, green and magenta lines DUCAD simulations.

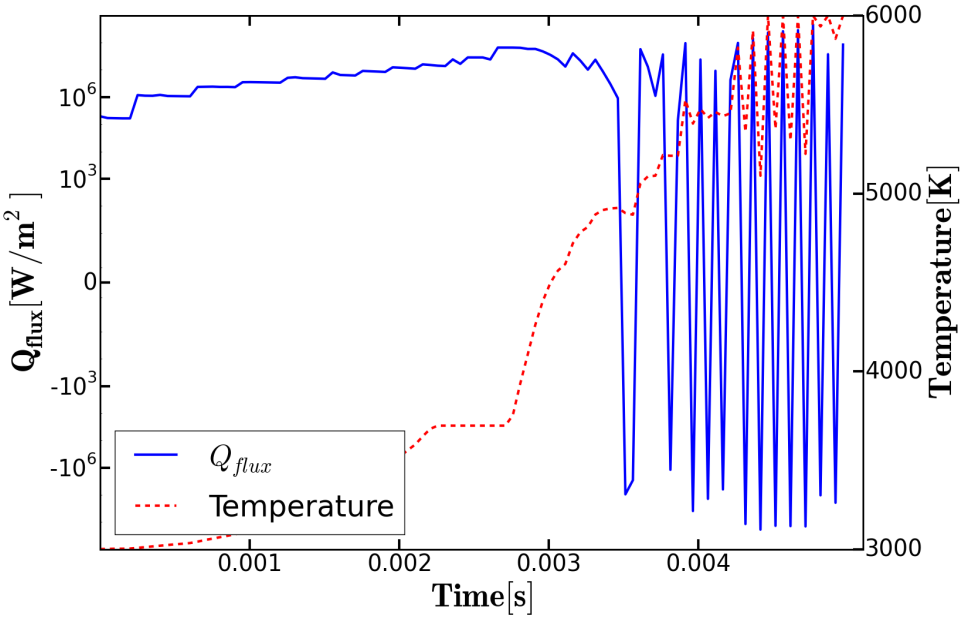



Figure 3.19: Heat flux to the $R_d = 10\mu m$ grain and temperature evolution

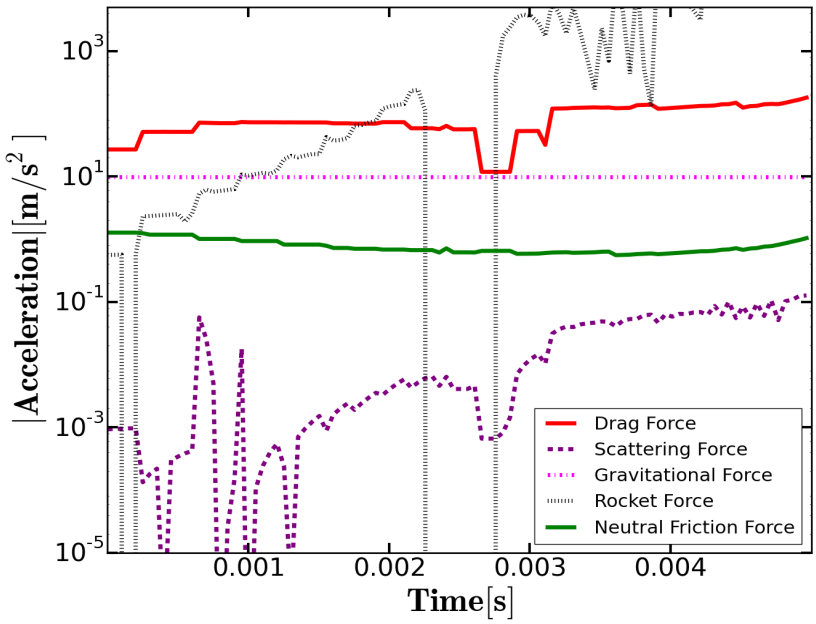



Figure 3.20: Acceleration of the dust grain depending on the force contribution

-perimental grain should probably be closer to the $R_d = 10\mu m$ case than to the other two. Therefore, a complete investigation of the intrinsic properties for this case is effectuated in Figures **3.19**, **3.20** and **3.21** where the temperature evolution, the acceleration and the radius rate of change can be

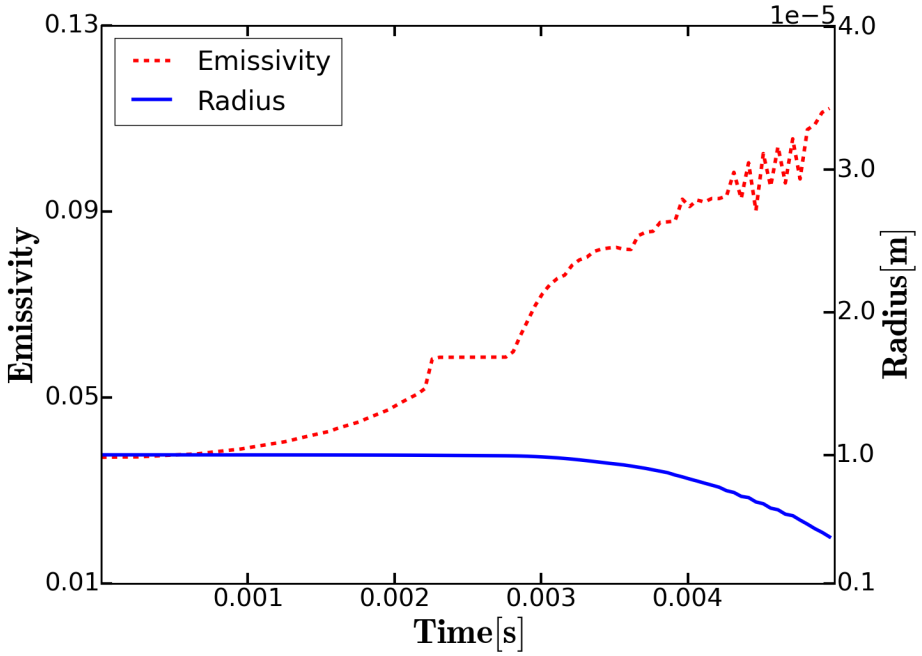



Figure 3.21: Emissivity and Radius evolution of the $R_d = 10\mu m$ dust particle over time.

observed. It is clear that since the particle approaches the core in its path, the temperature of the dust grain greatly rises and this leads to a fast grain ablation. Furthermore, the rocket force due to these intense temperature differences much dominates over the ion drag force except for those times when a phase transition is observed, $t \sim 0.002 - 0.003s$ (plateau in Figure **3.19**) in which case, there is no temperature increase in the grain due to melting. Toroidal motion is well represented by DUCAD simulations, however there are some discrepacies with repect to the poloidal projection. As the dust grain approaches the core, thermal gradients become more important and strong ablation effects can lead to important deviations from the dust homogeneous path due to a non uniform rocket force arising from a non-spherical particle, which could explain the observed behaviour. A wider range of simulation could be carried in order to fit a closer value of the dust initial size but since computational time greatly increases with dust particle size (due to the large number of integrals that must be evaluated in order to calculate the secondary electron emission) the spectrum has been kept in modest numbers such as two or three trajectories per experimental data set.

# Chapter 4

# Summary and future perspectives

A new code, DUCAD, DUst Characterization And Dynamics, has been presented following the same line of though as previous models for dust grains in fusion plasmas. A complete description of the theoretical foundations that include dust grain charging, heating, mass evolution and dynamics has been given and a further review is expected in future works in order to improve the accuracy of the model by including additional dust behaviour and dust wall-interactions. Some sample trajectories registered in Asdex Upgrade tokamak have been studied and it has been concluded that the simulated dust grains show satisfactory agreement for the most part of the cases when an homogeneous trajectory exists, for which a size stimation has also been possible. Furthermore, the study of dust intrinsic characteristics such as dust temperature evolution, grain ablation and dust thermal emissivity, key elements for characterizing particles seen by fast cameras, has been carried. While being true that a more extensive benchmarking is definetely needed to prove the code correct, the first layout and steps have shown a promissing direction.

# Appendix A

# Some considerations about Bessel functions

## A.1 Bessel functions of the first and second kind. Hankel functions.

Bessel functions are two linearly independent solutions of the second order linear homogeneous Bessel differential equation:

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - \alpha^2)y = 0 \quad \forall\, \alpha \in \mathbb{C} \tag{A.1}$$

Where $\alpha$ is the order of the differential equation. By applying Frobenius method, it is possible to find an expression as an infinite series. We can construct a solution for this equation valid near the origin; the form for such a solution is a series of ascending powers of x [**46**]:

$$y = \sum_{m=0}^{\infty} c_m z^{k+m} \tag{A.2}$$

Where k and $c_m$ need to be determined. This leads to a solution of the shape:

$$J_\alpha(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\Gamma(m+\alpha+1)} \left(\frac{x}{2}\right)^{2m+\alpha} \tag{A.3}$$

Where $\Gamma(x)$ is Euler's gamma function. $J_\alpha(x)$ is called the Bessel function of the first kind. Having found the first solution we may construct a second linearly independent one based on the fact that:

$$J_\alpha(x) = (-1)^\alpha J_{-\alpha}(x) \quad \forall\, \alpha \in \mathbb{Z} \tag{A.4}$$

Thus if $\alpha \notin \mathbb{Z}$ this relation does not hold and both $J_\alpha(x)$ and $J_{-\alpha}(x)$ are mutually linearly independent, allowing a second solution to be casted in:

$$\frac{J_\alpha(x) - (-1)^\alpha J_{-\alpha}(x)}{\alpha - int(\alpha)} \tag{A.5}$$

By evaluating the limit $\alpha \to int(\alpha)$ we can find the complete solution when $\alpha \in \mathbb{Z}$. Adopting the notation used by Weber, we may write:

$$Y_\alpha(x) = \frac{J_\alpha(x)cos(\alpha\pi) - J_{-\alpha}(x)}{sin(\alpha\pi)} \quad if\, \alpha \notin \mathbb{Z} \tag{A.6}$$

$$Y_\alpha(x) = \lim_{\alpha \to int(\alpha)} \frac{J_\alpha(x)cos(\alpha\pi) - J_{-\alpha}(x)}{sin(\alpha\pi)} \quad if\, \alpha \in \mathbb{Z} \tag{A.7}$$

This is called the Bessel function of second kind. As it is well known, if two solutions of a differential equation of second order are linearly independent, then any linear combination of them produces another solution of the differential equation. This allows us to define Hankel functions as:

$$H^1_\alpha(x) = J_\alpha(x) + iY_\alpha(x) \tag{A.8}$$

$$H^2_\alpha(x) = J_\alpha(x) - iY_\alpha(x) \tag{A.9}$$

## A.2 Spherical Bessel functions. Riccati-Bessel functions

A similar situation and perhaps closer to a physicist point of view, arises when solving the radial component of the scalar wave equation in spherical coordinates. The differential equation has the shape [**26**]:

$$\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) + \left[k^2r^2 - n(n+1)\right]R = 0 \tag{A.10}$$

Where k is the wave vector and n is a separation constant. As it turns out, this differential equation has a solution closely related to the ones derived in the previous section, namely the spherical Bessel functions:

$$j_\alpha(x) = \sqrt{\frac{\pi}{2x}}J_{\alpha+1/2}(x) \tag{A.11}$$

$$y_\alpha(x) = \sqrt{\frac{\pi}{2x}}Y_{\alpha+1/2}(x) \tag{A.12}$$

Where $j_\alpha(x)$ is the spherical Bessel function of the first kind and $y_\alpha(x)$ is the spherical Bessel function of the second kind. Furthermore, as previously, it is possible to construct another solution as a linear combination of both functions. These are called the spherical Hankel functions:

$$h^1_\alpha(x) = j_\alpha(x) + iy_\alpha(x) \tag{A.13}$$

$$h^2_\alpha(x) = j_\alpha(x) - iy_\alpha(x) \tag{A.14}$$

Finally, there is another family of functions called the Riccati-Bessel functions that can be constructed from the spherical Bessel functions such that:

$$\psi_\alpha(x) = x j_\alpha(x) \tag{A.15}$$

$$C_\alpha(x) = -x y_\alpha(x) \tag{A.16}$$

$$\xi_\alpha(x) = x h^1_\alpha(x) \tag{A.17}$$

$$\zeta_\alpha(x) = x h^1_\alpha(x) \tag{A.18}$$

# Appendix B

# DUCAD Code

## B.1 Main Routine

```
# -*- coding: utf-8 -*-
#--Main Loop for Dust Particle Transport Using EMC3EIRENE data--#
import numpy as np
import matplotlib as plt
from pylab import *
from cmath import *
from mpl_toolkits.mplot3d import Axes3D

###----Load data from EMC3EIRENE----###
execfile('emc3_reader.py');
execfile('load_data.py');

###----Parameters----###
execfile('Constants.py')              #Loads the table of constants
execfile('Dustparameters.py')         #Imports Initial Dust parameters
name_save=str(Rd)                     #Name for the .txt files
tot=100                               #Number of iterations
dt=5.01666666666651e-05               #Time step in seconds
time=np.arange(0.,tot*dt,dt)          #Time array
Z=1.                                  #Charge number of ions
vti=np.array([0.,0.,0.])              #Array for ion thermal velocity
vta=np.array([0.,0.,0.])              #Array for neutral velocity
vte=np.array([0.,0.,0.])              #Array for electron velocity
xp=np.zeros(tot)                      #Array for storing particle 'x'
yp=np.zeros(tot)                      #Array for storing particle 'y'
zp=np.zeros(tot)                      #Array for storing particle 'z'
Q_flux=np.zeros(tot)                  #Array for the power flux
Temp_dust=np.zeros(tot)               #Array for the dust temperature
Rd_array=np.zeros(tot)                #Array for the dust mass
Z0_array=np.zeros(tot)                #Array for the dust charge
Ftot_array=np.zeros(tot)              #Array for the total force
emissivity_array=np.zeros(tot)        #Array for the emissivity
Vd_array=np.zeros(tot)                #Array for storing the velocity
Fd_array=np.zeros(tot)                #Array for the Drag force
Fs_array=np.zeros(tot)                #Array for the Scattering force
Fg_array=np.zeros(tot)                #Array for the Gravitational
```

```
Fe_array=np.zeros(tot)              #Array for the Electric force
Fr_array=np.zeros(tot)              #Array for the Rocket force
Fa_array=np.zeros(tot)              #Array for the Neutral force
Continue_variable=1.                #Dummy variable

###----Loop for particle evolution calculation in time----###
for n in range(tot):
    print n
    if Continue_variable==0.:
        break
    ##--Plasma parameters in the cell the dust particle is--##
    execfile('Interpolation3.py');
    ne=run['A'].zone[j].field['ne'][ir,ip,it]
    M=run['A'].zone[j].field['M'][ir,ip,it]
    Ti=(Kb2/Kb1)*run['A'].zone[j].field['Ti'][ir,ip,it]
    Te=(Kb2/Kb1)*run['A'].zone[j].field['Te'][ir,ip,it]
    nH=run['A'].zone[j].field['nH'][ir,ip,it]
    ni=ne
    ##--Plasma values--##
    vti_mod=np.sqrt((2.*Ti)/mdi)
    vti[:]=vti_mod*ub_u[:]
    ui=M*vti
    vta_mod=np.sqrt((2.*Ti)/md)
    vta[:]=vta_mod*ub_u[:]
    ua=M*vta
    vte_mod=np.sqrt((2.*Te)/me)
    vte[:]=vte_mod*ub_u[:]
    ue=M*vte

    ##--Charging calculation--##
    try:
        execfile('Charging3.py')
    except SystemExit:
        print 'The charge number is=',Z0
    pot=Z0*qe/(4.*pi*eps0*Rd)

    ##--Heating calculation--##
    execfile('Heating7.py')

    ##--Mass calculation--##
    execfile('Mass.py')

    ##--Forces calculation--##
    execfile('Forces.py')

    ##--Plotting module--##
    execfile('Plotting.py')

###----Saving data into a .txt file----####
save_list=[xp,yp,zp,Q_flux,Temp_dust,Rd_array,Z0_array,
Ftot_array,emissivity_array,Vd_array,Fd_array,Fs_array,
Fg_array,Fe_array,Fr_array,Fa_array]
np.savetxt(name_save, save_list[:],
header='Array data for Dust Grain')
```

## B.2 Interpolation Routine

```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib as plt
from pylab import *

###--Interpolation routine--###

##--From cylindrical to Cartesian coordinates--##
xcc=np.zeros((14,399,16))                #3D array for Core
ycc=np.zeros((14,399,16))                #3D array for Core
xcs=np.zeros((51,610,16))                #3D array for SOL
ycs=np.zeros((51,610,16))                #3D array for SOL
xcp=np.zeros((22,211,16))                #3D array for PFR
ycp=np.zeros((22,211,16))                #3D array for PFR
if n==0:                                 #First time Ri is in cil.
    xd=Ri[0]*np.cos(Ri[1]*(pi/180.))     #X coordinate of dust
    yd=Ri[0]*np.sin(Ri[1]*(pi/180.))     #Y coordinate of dust
    zd=Ri[2]                             #Z coordinate of dust
else:                                    #n>0 Ri is in cartesians
    xd=Ri[0]                             #X coordinate of dust
    yd=Ri[1]                             #Y coordinate of dust
    zd=Ri[2]                             #Z coordinate of dust
xp[n]=xd                                 #Stores the 'x' coordinate
yp[n]=yd                                 #Stores the 'y' coordinate
zp[n]=zd                                 #Stores the 'z' coordinate
numberlist1=[0,3,6,9,12,15,18,21]        #j values for the Core
numberlist2=[1,4,7,10,13,16,19,22]       #j values for the SOL
numberlist3=[2,5,8,11,14,17,20,23]       #j values for the PFR
g=np.zeros(24)
for j in range(24):
    if j in numberlist1:
        for i in range(16):
            xcc[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
            np.cos(run['A'].zone[j].phic[i]*(pi/180.))
            ycc[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
            np.sin(run['A'].zone[j].phic[i]*(pi/180.))
            zc=run['A'].zone[j].zc[:,:,:]
            m=np.sqrt((xcc-xd)**2 + (ycc-yd)**2 + (zc-zd)**2)
            g[j]=m.min()
    if j in numberlist2:
        for i in range(16):
            xcs[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
            np.cos(run['A'].zone[j].phic[i]*(pi/180.))
            ycs[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
            np.sin(run['A'].zone[j].phic[i]*(pi/180.))
            zc=run['A'].zone[j].zc[:,:,:]
            m=np.sqrt((xcs-xd)**2 + (ycs-yd)**2 + (zc-zd)**2)
            g[j]=m.min()
    if j in numberlist3:
        for i in range(16):
            xcp[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
            np.cos(run['A'].zone[j].phic[i]*(pi/180.))
```

```
            ycp[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
            np.sin(run['A'].zone[j].phic[i]*(pi/180.))
            zc=run['A'].zone[j].zc[:,:,:]
            m=np.sqrt((xcp-xd)**2 + (ycp-yd)**2 + (zc-zd)**2)
            g[j]=m.min()

##--Interpolation--##
print np.where(g==g.min())
a1=np.where(g==g.min())        #Converts tuple into integer
b1=a1[0]
j=b1[0]

##--What is its cell?--##
if j in numberlist1:
    for i in range(16):
        xcc[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
        np.cos(run['A'].zone[j].phic[i]*(pi/180.))
        ycc[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
        np.sin(run['A'].zone[j].phic[i]*(pi/180.))
        zc=run['A'].zone[j].zc[:,:,:]
        m=np.sqrt((xcc-xd)**2 + (ycc-yd)**2 + (zc-zd)**2)
if j in numberlist2:
    for i in range(16):
        xcs[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
        np.cos(run['A'].zone[j].phic[i]*(pi/180.))
        ycs[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
        np.sin(run['A'].zone[j].phic[i]*(pi/180.))
        zc=run['A'].zone[j].zc[:,:,:]
        m=np.sqrt((xcs-xd)**2 + (ycs-yd)**2 + (zc-zd)**2)
if j in numberlist3:
    for i in range(16):
        xcp[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
        np.cos(run['A'].zone[j].phic[i]*(pi/180.))
        ycp[:,:,i]=run['A'].zone[j].Rc[:,:,i]*
        np.sin(run['A'].zone[j].phic[i]*(pi/180.))
        zc=run['A'].zone[j].zc[:,:,:]
        m=np.sqrt((xcp-xd)**2 + (ycp-yd)**2 + (zc-zd)**2)

print np.where(m==m.min())
a2=np.where(m==m.min())
b2=a2[0]
ir=b2[0]    #Radial Coordinate in which the particle is
b3=a2[1]
ip=b3[0]    #Poloidal Coordinate in which the particle is
b4=a2[2]
it=b4[0]    #Toroidal Coordinate in which the particle is

##--Position vector in cartesian coordinates--##
Ri[0]=xd
Ri[1]=yd
Ri[2]=zd

###--Calculation of the B-field vector--###
ub_u=np.array([0.,0.,0.])
```

```
Rc1=run['A'].zone[j].Rc[ir,ip,it]
zc1=run['A'].zone[j].zc[ir,ip,it]
phic1=run['A'].zone[j].phic[it]*(pi/180.)
xc1=Rc1*np.cos(phic1)
yc1=Rc1*np.sin(phic1)

#-Clockwise plasma flow-#
if it==15:
    Rc2=run['A'].zone[j+3].Rc[ir,ip,0]
    zc2=run['A'].zone[j+3].zc[ir,ip,0]
    phic2=run['A'].zone[j+3].phic[0]*(pi/180.)
else:
    Rc2=run['A'].zone[j].Rc[ir,ip,it-1]
    zc2=run['A'].zone[j].zc[ir,ip,it-1]
    phic2=run['A'].zone[j].phic[it-1]*(pi/180.)

xc2=Rc2*np.cos(phic2)
yc2=Rc2*np.sin(phic2)

ub=np.array([xc2-xc1,yc2-yc1,zc2-zc1])
ub_mod=np.sqrt((xc2-xc1)**2+(zc2-zc1)**2+(yc2-yc1)**2)
ub_u[:]=ub[:]/(ub_mod)

#-AntiClockwise plasma flow-#
if it==0:
    Rc2=run['A'].zone[j-3].Rc[ir,ip,15]
    zc2=run['A'].zone[j-3].zc[ir,ip,15]
    phic2=run['A'].zone[j-3].phic[15]*(pi/180.)
else:
    Rc2=run['A'].zone[j].Rc[ir,ip,it-1]
    zc2=run['A'].zone[j].zc[ir,ip,it-1]
    phic2=run['A'].zone[j].phic[it-1]*(pi/180.)

xc2=Rc2*np.cos(phic2)
yc2=Rc2*np.sin(phic2)

ub=np.array([xc1-xc2,yc1-yc2,zc1-zc2])
ub_mod=np.sqrt((xc2-xc1)**2+(zc2-zc1)**2+(yc2-yc1)**2)
ub_u[:]=ub[:]/(ub_mod)  #Unitary B-field vector in cartesians

###--Boundaries of the plasma parameters--###
namelist=['ne','nH','Te','Ti','M']
for dummy_name in range(size(namelist)):
    param=namelist[dummy_name]
    irmin=run['A'].zone[j].firstir[param]
    irmax=run['A'].zone[j].lastir[param]
    ipmin=run['A'].zone[j].firstip[param]
    ipmax=run['A'].zone[j].lastip[param]
    if ir<irmin or ir>irmax:
        print 'The particle is out of the plasma region'
        break
    if ip<ipmin or ip>ipmax:
        print 'The particle is out of the plasma region'
        break
```

## B.3 Charging routine

```
####——Calculation of the Charging of the Dust Particle——####
import numpy as np
from pylab import *
from cmath import *
import sys

###——Parameters——###
Z0_the=3.*Te*4.*pi*eps0*Rd/(qe**2)          #Theoretical value of Z0
Z0tot=int(Z0_the)                           #Adaptative charge number
Z0=np.arange(-Z0tot,Z0tot)                  #Range of search for Z0
if Z0tot==0:                                #Checks no ionization
    Z0=0.                                   #Assigns a 0 value
    print 'No ionization of the grain'      #Prints a warning message
    sys.exit()                              #Stops the script
Itot=np.zeros(2*Z0tot)                      #Array for the tot current
Ii=np.zeros(2*Z0tot)                        #Array for Ii
Ie=np.zeros(2*Z0tot)                        #Array for Ie
Ith=np.zeros(2*Z0tot)                       #Array for Ith
Isee=np.zeros(2*Z0tot)                      #Array for Isee
pot=Z0*qe/(4.*pi*eps0*Rd)                   #Dust potential
print 'The theoretical value w/o electron emission is=',-Z0_the

###——Extra parameters needed for analytical formulas——###
uitilde=np.sqrt((ui[0]-Vd[0])**2 + (ui[1]-Vd[1])**2 +
(ui[2]-Vd[2])**2)/vti_mod
uiptilde=uitilde+np.sqrt((Z*qe*pot[Z0tot+1:])/Ti)
uimtilde=uitilde-np.sqrt((Z*qe*pot[Z0tot+1:])/Ti)

###——Currents to/from the grain——###
##—Plasma ion currents to the grain—##
#if pot<=0:
Ii[0:Z0tot+1]=pi*(Rd**2)*Z*qe*ni*vti_mod*((uitilde +
(1./(2.*uitilde))*(1.-(2.*Z*qe*pot[0:Z0tot+1]/Ti)))
*math.erf(uitilde) + (1./(np.sqrt(pi)))*np.exp(-uitilde**2))
#if pot>0:
erf1=np.zeros(Z0tot-1)
erf2=np.zeros(Z0tot-1)
for i in range(0,Z0tot-1):
    x1=uimtilde[i]
    x2=uiptilde[i]
    erf1[i]=math.erf(x1)
    erf2[i]=math.erf(x2)
Ii[Z0tot+1:]=0.5*pi*(Rd**2)*Z*qe*ni*vti_mod*((uitilde +
(1./(2.*uitilde))*(1.-(2.*Z*qe*pot[Z0tot+1:]/Ti)))*(erf2 + erf1)
+ (1./(np.sqrt(pi)))*(1./uitilde)*(uiptilde*np.exp(-uimtilde**2)
+ uimtilde*np.exp(-uiptilde**2)))

##—Plasma electron currents to the grain—##
#if pot<=0:
Ie[0:Z0tot+1]=-2.*np.sqrt(pi)*(Rd**2)*
qe*ne*vte_mod*np.exp(qe*pot[0:Z0tot+1]/Te)
#if pot>0:
```

```
Ie[Z0tot+1:]=-2.*np.sqrt(pi)*(Rd**2)*
qe*ne*vte_mod*(1.+ qe*pot[Z0tot+1:]/Te)

##--Thermoionic emission from the dust grain--##
Psb=0.8              #Surface Barrier penetration factor
Wf=4.91              #Tungsten work function
Wf=Wf*Kb2/Kb1        #Tungsten work function in [J]
#if pot<=0:
Ith[0:Z0tot+1]=Psb*4.*pi*(Rd**2)*4.*pi*(Td**2)*me*(qe/(h**3))*
np.exp((-Wf+qe*np.sqrt(-qe*pot[0:Z0tot+1]/Rd))/Td)
#if pot>0:
Ith[Z0tot+1:]=Psb*4.*pi*(Rd**2)*4.*pi*(Td**2)*me*(qe/(h**3))*
(1.+qe*pot[Z0tot+1:]/Td)*np.exp((-Wf-qe*pot[Z0tot+1:])/Td)

##--Secondary electron emission of the dust grain--##
k=1.38
n_size=100
dummy_variable1=np.zeros(n_size)
dummy_variable2=np.zeros(n_size)
for x in range(1,n_size):
    dummy_variable1[0]=1.e12
    y=x/10.
    dummy_variable1[x]=abs(np.exp(y)*(1.-1./k)-1.+1./k-y)
rm1=np.where(dummy_variable1==dummy_variable1.min())
rm2=rm1[0]
rm=rm2[0]/10.

deltam=0.93                    #Maximum SEE yield
Em=(600./Kb1)*Kb2              #Energy of incident electrons
betha_tita=0.8                 #SEE Exponent

#if pot<=0:
xdomain_min=0.
xdomain_max=10000.
xdomain_step=0.011
int_array=np.zeros(xdomain_max/xdomain_step)
x_domain=np.arange(xdomain_min,xdomain_max,xdomain_step)
x_int=(x_domain[1:]+
x_domain[0:((xdomain_max-xdomain_min)/xdomain_step)])/2.
int_array[:]=x_int[:]*np.exp(-x_int[:])*
(deltam/(1.-np.exp(-rm)))*((x_int[:]*Te/Em)**(1.-k))*
(1.-np.exp(-rm*(((x_int[:]*Te)/Em)**(k))))
int_see=xdomain_step*sum(int_array)
Isee[0:Z0tot+1]=-((2.)/(2.-betha_tita))*Ie[0:Z0tot+1]*int_see
#if pot>0
z=-qe*pot[Z0tot+1:]/Te
sigma=-z*Te/Wf
int_see=np.zeros(size(z))
number_see=1.111111e3
int_array=np.zeros(number_see)
for dummy_see in range(0,Z0tot-1):
    xdomain_min=-z[dummy_see]
    xdomain_max=1500.*xdomain_min
    xdomain_step=(xdomain_max-xdomain_min)/number_see
```

```
    x_domain=np.arange(xdomain_min,xdomain_max,xdomain_step)
    x_int=(x_domain[1:]+
    x_domain[0:((xdomain_max-xdomain_min)/xdomain_step)])/2.
    int_array[:]=x_int[:]*np.exp(-x_int[:])*
    (deltam/(1.-np.exp(-rm)))*((x_int[:]*Te/Em)**(1.-k))*
    (1.-np.exp(-rm*(((x_int[:]*Te)/Em)**(k))))
    int_see[dummy_see]=xdomain_step*sum(int_array)
Isee[Z0tot+1:]=-((2.)/(2.-betha_tita))*
((1.+3.*sigma)/(1.+sigma)**3)*((np.exp(-z))/(1.-z))*Ie[Z0tot+1:]*
int_see[:]
Isee[Z0tot+1:Z0tot+10]=Isee[Z0tot]
for dummy_see2 in range(size(Isee)):
    if math.isnan(Isee[dummy_see2])==True:
        Isee[dummy_see2]=0.

###--Total current to the dust grain--###
Itot=abs(Ii+Ie+Isee+Ith)

###--Find the Z0 thah minimizes Itot (floating potential)--###
Z2=np.where(Itot==Itot.min()) #Look for the minimum value
Z1=Z2[0]
Z0=Z1[0]                      #Z0 is the one that minimizes sum
Z0=Z0/1.
Z0=Z0-Z0tot
Z0_array[n]=Z0

print 'The charge number is=',Z0
```

## B.4 Heating Routine

```
####--Calculation of the heating of the dust grains--####
import numpy as np
from pylab import *
from cmath import *
from scipy import special

##--Relation between enthalpy and temperature--##
P=ne*Te+ni*Ti
print 'Input temperature=',Td/Kb2, 'K'
Td_minus=Td

if n==0:
    Td=Td/Kb2
    Temp_dust[0]=Td
    if 298<Td<3695:                    #Considering BCC_A2
        a=-7646.311
        c=-24.1
        d2=-1.936e-3
        d3=2.07e-7
        d4=-5.33e-11
        dn1=44500.
        A=9.5168e-6
        a1_p=5.51e-9
```

```
        Hpres=A*P*(1.-a1_p*(Td**2)/2.)    #Pressure term
        Htot=a-c*Td-(d2*(Td**2)+2.*d3*(Td**3)+3.*d4*(Td**4)
        -2.*dn1*(Td**-1))+Hpres           #Total molar entalphy
        Td=Td*Kb2
    if Td==3695:        #Temperature of transition
        A=9.5168e-6
        a1_p=5.51e-9

        Hpres=A*P*(1.-a1_p*(Td**2)/2.)
        deltah_trans=52313.69             #Enthalpy of transition
        Htot=116780.2642252032+Hpres      #Molar enthalpy at 3695K
        Td=Td*Kb2
    if 3695<Td<=6000:  #Liquid phase above 3695K
        a=-30436.051
        b=375.175
        c=-54.0
        A=9.5168e-6
        a1_p=5.51e-9

        Hpres=A*P*(1.-a1_p*(Td**2)/2.)
        Htot=a-c*Td+Hpres                 #Total molar entalphy
        Td=Td*Kb2

print 'The initial Enthalpy is',Htot, '[J/mol]'

#-Molar enthalpy-#
nw=Md/mw                #Number of tungsten atoms
nm=nw/Na                #Number of W moles
Htot=Htot*nm            #Total enthalpy from molar enthalpy

##--Extra parameters needed for analytical formulas--##
uiptilde=uitilde+np.sqrt((Z*qe*pot)/Ti)
uimtilde=uitilde-np.sqrt((Z*qe*pot)/Ti)

##--Dust heating power by the kinetic energy of ions--##
if pot<=0:
    Pki=pi*(Rd**2)*ni*vti_mod*Ti*((1./(2.*np.sqrt(pi)))*
    (5.+(2.*uitilde**2)-((2.*Z*qe*pot)/(Ti)))*np.exp(-uitilde**2)
    +(1./(4.*uitilde))*(3.+(12.*uitilde**2)+(4.*uitilde**4)-
    (((2.*Z*qe*pot)/(Ti))*(1.+(2.*uitilde**2))))*math.erf(uitilde)
        )
else:
    Pki=pi*(Rd**2)*ni*vti_mod*Ti*((1./(4.*np.sqrt(pi)))*
    ((5.+(2.*uitilde**2)-(((3.+2.*uitilde**2)/(uitilde))*
    np.sqrt((Z*qe*pot)/(Ti))))*np.exp(-uiptilde**2)
    +(5.+(2.*uitilde**2)+(((3.+2.*uitilde**2)/(uitilde))*
    np.sqrt((Z*qe*pot)/(Ti))))*np.exp(-uimtilde**2))
    +(1./(8.*uitilde))*(3.+(12.*uitilde**2)+(4.*uitilde**4)
    -(((2.*Z*qe*pot)/(Ti))*(1.+(2.*uitilde**2)))*
    (math.erf(uiptilde)+ math.erf(uimtilde))) )

##--Dust heating power by the kinetic energy of electrons--##
if pot<=0:
    Pke=2.*np.sqrt(pi)*(Rd**2)*ne*vte_mod*Te*
```

```
    (2.-((qe*pot)/(Te)))*np.exp((qe*pot)/(Te))
else:
    Pke=2.*np.sqrt(pi)*(Rd**2)*ne*vte_mod*
    Te*(2.+((qe*pot)/(Te)))

##--Dust heating power provided by thermionic emission--##
sig=qe*pot/Td
if pot<=0:
    Pte=4.*pi*(Rd**2)*(-Ith[Z0+Z0tot-1]/(qe*4.*pi*(Rd**2)))*
    (Wf+2.*Td-qe*np.sqrt((-qe*pot)/(Rd)))
else:
    Pte=4.*pi*(Rd**2)*(-Ith[Z0+Z0tot-1]/(qe*4.*pi*(Rd**2)))*
    (Wf + ((2.+2.*sig+sig**2)/(1.+sig))*Td)

##--Dust heating power provided by evaporation--##
Td=Td/Kb2

if 298<=Td<=2500:
    Pv=np.exp(5.006 + 2.945 - 44094.*(Td**(-1.))
    + 1.3677*np.log(Td)) #Vapor pressure in [Pa]
if 2500<Td<=6000:
    Pv=np.exp(-45385./Td+7.871)
    Pv=Pv*101325.          #Vapor pressure in [Pa]

if 298<=Td<=2500:      #Solid phase
    Td=Td*Kb2
    Pvap=-Ad*np.sqrt((mw)/(2.*pi*Td))*Pv*
    ((Htot+(deltah_sub*nm))/Md)
if 2500<Td<3695:       #Solid phase
    Td=Td*Kb2
    Pvap=-Ad*np.sqrt((mw)/(2.*pi*Td))*Pv*
    ((Htot+(deltah_sub*nm))/Md)
if Td==3695:           #Melting phase
    Td=Td*Kb2
    Pvap=-Ad*np.sqrt((mw)/(2.*pi*Td))*Pv*
    ((Htot+(deltah_sub*nm))/Md)
if 3695<Td<=6000:      #Liquid phase above 3695K
    Td=Td*Kb2
    Pvap=-Ad*np.sqrt((mw)/(2.*pi*Td))*Pv*
    ((Htot+(deltah_vap*nm))/Md)
if Td>3100:
    print 'Vapor pressure is not valid in this range'

##--Dust heating due to secondary electron emission--##
if pot<=0:
    Psee=3.*Ad*Wf*(-Isee[Z0+Z0tot-1]/(qe*Ad))
else:
    z=-qe*pot/Te
    sigma=-z*Te/Wf
    Psee=3.*Ad*Wf*((1.+sigma)*(1.+2.*sigma)/(1.+3.*sigma))*
    (-Isee[Z0+Z0tot-1]/(qe*Ad))

##--Dust heating due to thermal emission--# (Mie theory)
Td=Td/Kb2
```

```
#-Thermal DC Conductivity-#
if 90<Td<750:       #res [Ohm*m]*10^8
    res=-1.06871+(2.06884e-2)*Td+(1.27971e-6)*Td**2
    +(8.53101e-9)*Td**3-(5.14195e-12)*Td**4
if 750<=Td<=3660: #res [Ohm*m]*10^8
    res=-1.72573+(2.14350e-2)*Td+(5.74811e-6)*Td**2
    -(1.13698e-9)*Td**3+(1.1167e-13)*Td**4
if 3660<Td<=6000:
    res=(Td-3660.)/42.+131.          #Resistivity by a linear fit
Td=Td*Kb2
res=res*1.e-8                        #Correction to the resistivity
con_DC=1./res                        #DC conductivity of W
wp=2.*pi*(5.17e4)*100.*c_light       #Plasma frequency of W
f=1./0.49                            #Correction factor
con_OPT=con_DC/f                     #Optical conductivity
eps_inf=1.                           #High frequency dielectric
w_tau=((wp**2)*eps0)/con_OPT         #Electron collision frequency

#-Integral evaluation-#
lam_cut=4.*pi*c_light/wp             #Cutoff wavelength
lam_step=0.01e-6
lam_max=200.e-6

lam=np.arange(lam_cut,lam_max,lam_step)
lamu=(lam[1:]+lam[0:((lam_max-lam_cut)/lam_step)])/2.
w=2.*pi*c_light/lamu
eps_1=eps_inf-((wp**2)/(w**2+w_tau**2)) #Real part of permitivity
eps_2=((w_tau*wp**2)/(w**3+w*w_tau**2)) #Ima. part of permitivity
eps_mod=np.sqrt(eps_1**2+eps_2**2)      #Permitivity modulus

n0=1.                                #Ind of refrac of plasma
n_mat=np.sqrt((eps_mod+eps_1)/2.)    #Real part
kappa=np.sqrt((eps_mod-eps_1)/2.)    #Imaginary part

m_rel=(n_mat+kappa*1.j)/n0           #Relative refractive index
x_size=2.*pi*Rd/lamu                 #Size parameter
k_wave=2.*pi/lamu                    #Wave vector

Q_abs=np.zeros((lam_max-lam_cut)/lam_step)
Q_ext=np.zeros((lam_max-lam_cut)/lam_step)

for i in range(0,int((lam_max-lam_cut)/lam_step)):

    nu_max= int(x_size[i] + 4.*x_size[i]**(1./3.) +2)
    nu_2=int(max(nu_max,abs(m_rel[i]*x_size[i]))+15)
    jn_x=np.zeros(nu_max+2)
    yn_x=np.zeros(nu_max+2)
    sum_c=np.zeros(nu_max)
    sum_ext=np.zeros(nu_max)
    a_nu=np.zeros(nu_max,dtype=complex)
    b_nu=np.zeros(nu_max,dtype=complex)
    nu=np.arange(0,nu_max+1,1)
```

```
        jn_x[0]=np.cos(x_size[i])      #j_-1(x)
        jn_x[1]=np.sin(x_size[i])      #j_0(x)
        for k in range(2,nu_max+2):
            jn_x[k]=((2.*nu[k-2]+1.)/x_size[i])*jn_x[k-1] - jn_x[k-2]

        yn_x[0]=-np.sin(x_size[i])     #y_-1(x)
        yn_x[1]=np.cos(x_size[i])      #y_0(x)
        for k in range(2,nu_max+2):
            yn_x[k]=((2.*nu[k-2]+1.)/x_size[i])*yn_x[k-1] - yn_x[k-2]

        psi=jn_x-complex(0,1)*yn_x     #Riccati-Bessel of third order

        D_mx=np.zeros(nu_2+1,dtype=complex)
        D_mx[nu_2]=0.
        D_mxr=D_mx[::-1]   #Inverse array
        nup=np.arange(1,nu_2+1)
        nup_r=nup[::-1]
        D_mxr[1:]=(nup_r[0:nu_2]/(m_rel[i]*x_size[i]))-1./
        (D_mxr[0:nu_2]+(nup_r[0:nu_2]/(x_size[i]*m_rel[i])))
        D_mx=D_mxr[::-1]   #Original array

        a_nu[0:]=((D_mx[1:nu_max+1]/m_rel[i]
        + nu[1:]/x_size[i])*jn_x[2:] - jn_x[1:nu_max+1])/
        ((D_mx[1:nu_max+1]/m_rel[i]
        + nu[1:]/x_size[i])*psi[2:] - psi[1:nu_max+1])
        b_nu[0:]=((m_rel[i]*D_mx[1:nu_max+1]
        + nu[1:]/x_size[i])*jn_x[2:] - jn_x[1:nu_max+1])/
        ((m_rel[i]*D_mx[1:nu_max+1]
        + nu[1:]/x_size[i])*psi[2:] - psi[1:nu_max+1])

        sum_c[0:]=(2.*nu[1:]+1.)*(np.real(a_nu[0:]+b_nu[0:])
        -(np.abs(a_nu[0:])**2+np.abs(b_nu[0:])**2))
        sum_ext[0:]=(2.*nu[1:]+1.)*np.real(a_nu[0:]+b_nu[0:])
        Q_abs[i]=(2./(x_size[i]**2))*(sum(sum_c))
        Q_ext[i]=(2./(x_size[i]**2))*(sum(sum_ext))

P_lam=((2.*h*c_light**2)/(lamu**5))*
(1./(np.exp((h*c_light)/(lamu*Td))-1.))
f_lam=Q_abs*P_lam
int_rad=lam_step*(sum(f_lam))
int_P=lam_step*sum(P_lam)                    #Normalization

#-Irad-#
Irad=pi*int_rad                              #Intesity of radiation
Td=Td/Kb2
ems=Irad/(sig_SB*Td**4)                      #Emissivity coefficient
emissivity_array[n]=ems                      #Stores the emissivity
print 'Emissivity coefficient is=',ems
Twall=300.                                   #Wall temperature [K]
Prad=-Ad*sig_SB*ems*(Td**4-Twall**4)         #Power of radiation
Td=Td*Kb2

##--Ion surface neutralization--## (Migraine)
Eisn=(7*Kb2)/Kb1    #(Tungsten)
```

```
Pisn=Eisn*Ii[Z0+Z0tot-1]/(Z*qe)

##--Finite diferences squeme for the new enthalpy calculation--##
Ptot=Pki+Pke+Pte+Pvap+Psee+Prad+Pisn      #Total power flux
Q_flux[n]=Ptot/Ad                         #Array for the power flux
Htot=Ptot*dt+Htot                         #Finite differences

if Htot<0: #Checks if the enthalpy is negative-> Time step too big
    print('The time step is too big for a correct
    computation of the Heating balance')
    Continue_variable=0.

Htot=Htot/nm                              #New molar enthalpy
A=9.5168e-6
a1_p=5.51e-9
Hpres=A*P*(1.-a1_p*(Td**2)/2.)
Hntotmol_p=Htot-Hpres
##--Temperature interpolation from new enthalpy--##
Htot3694=116780.21051525505           #Molar enthalpy at 3694.999K
Htot3695=169094.003                   #Molar enthalpy at 3695.001K
Htot6000=293563.949                   #Molar enthalpy at 6000.000K


if 0<Hntotmol_p<=Htot3694:            #Considering structure BCC_A2
    Tda=np.arange(299,3694,1)
    a=-7646.311
    c=-24.1
    d2=-1.936e-3
    d3=2.07e-7
    d4=-5.33e-11
    dn1=44500.
    A=9.5168e-6
    a1=5.51e-9

    Htota=a-c*Tda-(d2*(Tda**2)+2.*d3*(Tda**3)+3.*d4*(Tda**4)
    -2.*dn1*(Tda**-1))                #Total entalphy
    Hrel=abs(Htota-Hntotmol_p)
    H1=np.where(Hrel==Hrel.min())
    H2=H1[0]
    Td=H2[0]+299                      #New dust temperature
    Td=Td/1.

if Htot3694<Hntotmol_p<=Htot3695:     #Temperature of transition
    Td=3695.

if Htot3695<Hntotmol_p<=Htot6000:     #Liquid phase above 3695K
    Tda=np.arange(3696,6000,1)
    a=-30436.051
    b=375.175
    c=-54.0
    A=9.5168e-6
    a1=5.51e-9

    Htota=a-c*Tda                     #Total entalphy
```

```
    Hrel=abs(Htota-Hntotmol_p)
    H1=np.where(Hrel==Hrel.min())
    H2=H1[0]
    Td=H2[0]+3696                          #New dust temperature
    Td=Td/1.
if Hntotmol_p>Htot6000:
    Td=6000.
    Htot=Htot6000 #Prevents enthalpy to go > values than H(6000K)
    print 'No enthalpy data available above 6000K'

print 'The output temperature is', Td
Temp_dust[n]=Td        #Stores the new temperature in the array
Td=Td*Kb2              #Temperature from [K] to [J]
Td_plus=Td
```

## B.5 Mass Calculation Routine

```
###--Calculation of the mass equation for the dust particle--###
import numpy as np
from pylab import *
from cmath import *

#--Hertz-Knudsen formula: Maximum atomic flux leaving the grain--#

Td=Td/Kb2
Ad=4.*pi*(Rd**2)                       #Area of the dust grain
Ad=Ad+2.*Ad*coeff_lin*(Td-300.)        #Area of the dust grain with
                                       #thermal expansion correction

if 298<=Td<=2500:
    Pv=np.exp(5.006 + 2.945 - 44094.*(Td**(-1.))
    + 1.3677*np.log(Td)) #Vapor pressure in [Pa]
if 2500<Td<=6000:
    Pv=np.exp(-45385./Td+7.871)
    Pv=Pv*101325.          #Vapor pressure in [Pa]

Td=Td*Kb2
Md=Md - Ad*dt*np.sqrt((mw)/(2.*pi*Td))*Pv     #Knudsen formula
if Md<0:  #Checks if the dust grain has completely evaporated
    print 'The dust grain has evaporated'
Rd=((3.*Md)/(4.*pi*rho))**(1./3.)              #New radius
Rd_array[n]=Rd
```

## B.6 Forces Routine

```
# -*- coding: utf-8 -*-
##--Calculation of the Forces that rule Dust Particle Dynamics--##
import numpy as np
from pylab import *
from cmath import *
from math import *

##----Parameters----##
```

```
Fd=np.array([0.,0.,0.])     #Array for absorption force
Fs=np.array([0.,0.,0.])     #Array for scattering force
Fa=np.array([0.,0.,0.])     #Array for neutral friction force
Fg=np.array([0.,0.,0.])     #Array for gravitational force
E=np.array([0.,0.,0.])      #Array for the electric field
E_c=np.array([0.,0.,0.])    #Array for the electric field
Fe=np.array([0.,0.,0.])     #Array for electric force
Fr=np.array([0.,0.,0.])     #Array for rocket force

##--Extra parameters needed for analytical formulas--##
uiptilde=uitilde+np.sqrt((Z*qe*pot)/Ti)
uimtilde=uitilde-np.sqrt((Z*qe*pot)/Ti)
wimtilde=(uitilde**2)-(Z*qe*pot/Ti)
wiptilde=(uitilde**2)+(Z*qe*pot/Ti)
deb_e=np.sqrt(eps0*Te/(ne*(qe**2)))        #Electron Debye Length
deb_i=np.sqrt(eps0*Ti/(ni*(qe**2)))        #Ion Debye Length
uatilde=np.sqrt((ua[0]-Vd[0])**2 + (ua[1]-Vd[1])**2
+ (ua[2]-Vd[2])**2)/vta_mod

##--Check OML condition--##
print 'deb_e/Rd=',deb_e/Rd          #OML approximation valid as long
                                    #as deb_e/Rd > 1

##--Ion drag force arising from absorption of ions--##
if pot<=0:
    Fd=pi*(Rd**2)*mdi*ni*vti_mod*(ui-Vd)*(1./(2.*(uitilde)**2))*
    ((1./np.sqrt(pi))*(1.+2.*wimtilde)*np.exp(-(uitilde)**2)
    + uitilde*(1.+2.*wimtilde - (1./(2.*(uitilde)**2))*
    (1.-2.*wiptilde))* math.erf(uitilde))
else:
    Fd=pi*(Rd**2)*mdi*ni*vti_mod*(ui-Vd)*(1./(4.*(uitilde)**2))*
    ((1./np.sqrt(pi))*((1.+2.*(uitilde)**2
    +((1.-2.*(uitilde)**2)/uitilde)*np.sqrt((qe*pot)/Ti))*
    np.exp(-(uiptilde)**2) + (1. + 2.*(uitilde)**2-
    ((1.-2.*(uitilde)**2)/uitilde)*np.sqrt((qe*pot)/Ti))*
    np.exp(-(uimtilde)**2)) +uitilde*(1.+2.*wimtilde-
    (1./(2.*(uitilde)**2))*(1.-2.*wiptilde))*
    (math.erf(uiptilde) + math.erf(uimtilde)))

##--Ion drag force arising from scattering of ions--##
m_rat=mdi/mp
v_eff=np.sqrt(2.*Ti/mdi+((M*vti_mod)**2)*
(1+((np.abs(M*vti_mod)/np.sqrt(2.*Z*Te/mdi))/
(0.6+0.05*np.log(m_rat/Z)+(deb_e/(5.*Rd))*
(np.sqrt(Ti/(Z*Te))-0.1)))**3))
b90=Rd*Z*qe*np.abs(pot)/(mdi*(v_eff)**2)
deb_s=np.sqrt((deb_e)**2/(1+(2*Z*Te/(mdi*(v_eff)**2)))+Rd**2)
G=(math.erf(uitilde) - ((2.*uitilde*np.exp(-(uitilde)**2))/
(np.sqrt(pi))))/(2.*(uitilde)**2)          #Chandrasekhar function
lam=np.log((b90+deb_s)/(b90+Rd))           #Coulomb logarithm
Fs=2.*pi*(Rd**2)*mdi*ni*vti_mod*(ui-Vd)*((Z*qe*pot/Ti)**2)*
(G/uitilde)*lam                            #Formal scattering force

#-Neutral drag force arising from neutral friction with the Dust-#
```

```
Fa=pi*(Rd**2)*md*nH*vta_mod*(ua-Vd)*((1./np.sqrt(pi))*
(1. + 1./(2.*uatilde**2))*np.exp(-uatilde**2) + (1./(uatilde))*
(1.+(uatilde**2)-(1./(4.*uatilde**2))*math.erf(uatilde)))

##--Gravitational force--##
Fg[2]=-(Md*gav)          #Gravitational force in 'z' direction

##--Rocket Force--##
Td_plus=Td_plus/Kb2
if 298<=Td_plus<=2500:
    Pv_plus=np.exp(5.006 + 2.945 - 44094.*(Td_plus**(-1.))
    + 1.3677*np.log(Td_plus))    #Vapor pressure in [Pa]
if 2500<Td_plus<=6000:
    Pv_plus=np.exp(-45385./Td_plus+7.871)
    Pv_plus=Pv_plus*101325.      #Vapor pressure in [Pa]

Td_minus=Td_minus/Kb2
if 298<=Td_minus<=2500:
    Pv_minus=np.exp(5.006 + 2.945 - 44094.*(Td_minus**(-1.))
    + 1.3677*np.log(Td_minus))   #Vapor pressure in [Pa]
if 2500<Td_minus<=6000:
    Pv_minus=np.exp(-45385./Td_minus+7.871)
    Pv_minus=Pv_minus*101325.    #Vapor pressure in [Pa]

Fr=3.*Md/(4.*np.sqrt(2.*pi)*rho)*((Pv_plus-Pv_minus)/Rd)*
np.sign(M)*ub_u


##--Electric force--##
#Potential along toroidal line of sight
numberlist_elec=[0,15]
numberlist_elec1=[0,398] #Poloidal indexes in the core
numberlist_elec2=[0,609] #Poloidal indexes in the SOL
numberlist_elec3=[0,210] #Poloidal indexes in the PFR
numberlist_elec4=[0,13]  #Radial indexes in the core
numberlist_elec5=[0,50]  #Radial indexes in the SOL
numberlist_elec6=[0,21]  #Radial indexes in the PVR
numberlist1=[0,3,6,9,12,15,18,21]   #j values for the Core Regions
numberlist2=[1,4,7,10,13,16,19,22]  #j values for the SOL Regions
numberlist3=[2,5,8,11,14,17,20,23]  #j values for the PFR
#Potential along toroidal line of sight
if it==15:
    pot_plasma_it1=run['A'].zone[j+3].field['phi'][ir,ip,0]
    Rc_it1=run['A'].zone[j+3].Rc[ir,ip,0]
    phic_it1=run['A'].zone[j+3].phic[0]
    zc_it1=run['A'].zone[j+3].zc[ir,ip,0]
    pot_plasma_it0=run['A'].zone[j].field['phi'][ir,ip,it-1]
    Rc_it0=run['A'].zone[j].Rc[ir,ip,it-1]
    phic_it0=run['A'].zone[j].phic[it-1]
    zc_it0=run['A'].zone[j].zc[ir,ip,it-1]
if it==0:
    pot_plasma_it1=run['A'].zone[j].field['phi'][ir,ip,it+1]
    Rc_it1=run['A'].zone[j].Rc[ir,ip,it+1]
    phic_it1=run['A'].zone[j].phic[it+1]
```

```
        zc_it1=run['A'].zone[j].zc[ir,ip,it+1]
        pot_plasma_it0=run['A'].zone[j-3].field['phi'][ir,ip,15]
        Rc_it0=run['A'].zone[j-3].Rc[ir,ip,15]
        phic_it0=run['A'].zone[j-3].phic[15]
        zc_it0=run['A'].zone[j-3].zc[ir,ip,15]
    if not it in numberlist_elec:
        pot_plasma_it1=run['A'].zone[j].field['phi'][ir,ip,it+1]
        Rc_it1=run['A'].zone[j].Rc[ir,ip,it+1]
        phic_it1=run['A'].zone[j].phic[it+1]
        zc_it1=run['A'].zone[j].zc[ir,ip,it+1]
        pot_plasma_it0=run['A'].zone[j].field['phi'][ir,ip,it-1]
        Rc_it0=run['A'].zone[j].Rc[ir,ip,it-1]
        phic_it0=run['A'].zone[j].phic[it-1]
        zc_it0=run['A'].zone[j].zc[ir,ip,it-1]
    xc_it1=Rc_it1*np.cos(phic_it1*(180./pi))
    xc_it0=Rc_it0*np.cos(phic_it1*(180./pi))
    yc_it1=Rc_it1*np.sin(phic_it1*(180./pi))
    yc_it0=Rc_it0*np.sin(phic_it1*(180./pi))
    mod_it=np.sqrt((xc_it1-xc_it0)**2 + (yc_it1-yc_it0)**2
    + (zc_it1-zc_it0)**2)
    #Potential along poloidal line of sight
    #-CORE-#
    if j in numberlist1:
        if ip==398:
            pot_plasma_ip1=run['A'].zone[j].field['phi'][ir,0,it]
            pot_plasma_ip0=run['A'].zone[j].field['phi'][ir,ip-1,it]
            Rc_ip1=run['A'].zone[j].Rc[ir,0,it]
            Rc_ip0=run['A'].zone[j].Rc[ir,ip-1,it]
            phic_ip1=run['A'].zone[j].phic[it]
            phic_ip0=run['A'].zone[j].phic[it]
            zc_ip1=run['A'].zone[j].zc[ir,0,it]
            zc_ip0=run['A'].zone[j].zc[ir,ip-1,it]
        if ip==0:
            pot_plasma_ip1=run['A'].zone[j].field['phi'][ir,ip+1,it]
            pot_plasma_ip0=run['A'].zone[j].field['phi'][ir,398,it]
            Rc_ip1=run['A'].zone[j].Rc[ir,ip+1,it]
            Rc_ip0=run['A'].zone[j].Rc[ir,398,it]
            phic_ip1=run['A'].zone[j].phic[it]
            phic_ip0=run['A'].zone[j].phic[it]
            zc_ip1=run['A'].zone[j].zc[ir,ip+1,it]
            zc_ip0=run['A'].zone[j].zc[ir,398,it]
        if not j in numberlist_elec1:
            pot_plasma_ip1=run['A'].zone[j].field['phi'][ir,ip+1,it]
            pot_plasma_ip0=run['A'].zone[j].field['phi'][ir,ip-1,it]
            Rc_ip1=run['A'].zone[j].Rc[ir,ip+1,it]
            Rc_ip0=run['A'].zone[j].Rc[ir,ip-1,it]
            phic_ip1=run['A'].zone[j].phic[it]
            phic_ip0=run['A'].zone[j].phic[it]
            zc_ip1=run['A'].zone[j].zc[ir,ip+1,it]
            zc_ip0=run['A'].zone[j].zc[ir,ip-1,it]
    #-SOL-#
    if j in numberlist2:
        Rcc_own=run['A'].zone[j].Rc[ir,ip,it]
        Zcc_own=run['A'].zone[j].zc[ir,ip,it]
```

```
if ip==609:
    jPFR_electric=j+1
    Rcc_elec=np.zeros((22,211,16))
    Zcc_elec=np.zeros((22,211,16))
    Rcc_elec[:,:,it]=run['A'].zone[jPFR_electric].Rc[:,:,it]
    Zcc_elec[:,:,it]=run['A'].zone[jPFR_electric].zc[:,:,it]
    m=np.sqrt((Rcc_elec-Rcc_own)**2 + (Zcc_elec-Zcc_own)**2)
    a2=np.where(m==m.min())
    b2=a2[0]
    irPFR_elec=b2[0]
    b3=a2[1]
    ipPFR_elec=b3[0]
    pot_plasma_ip1=run['A'].zone[jPFR_electric].
    field['phi'][irPFR_elec,ipPFR_elec,it]
    pot_plasma_ip0=run['A'].zone[j].field['phi'][ir,ip-1,it]
    Rc_ip1=run['A'].zone[jPFR_electric].
    Rc[irPFR_elec,ipPFR_elec,it]
    Rc_ip0=run['A'].zone[j].Rc[ir,ip-1,it]
    phic_ip1=run['A'].zone[jPFR_electric].phic[it]
    phic_ip0=run['A'].zone[j].phic[it]
    zc_ip1=run['A'].zone[jPFR_electric].
    zc[irPFR_elec,ipPFR_elec,it]
    zc_ip0=run['A'].zone[j].zc[ir,ip-1,it]
if ip==0:
    jPFR_electric=j+1
    Rcc_elec=np.zeros((22,211,16))
    Zcc_elec=np.zeros((22,211,16))
    Rcc_elec[:,:,it]=run['A'].zone[jPFR_electric].Rc[:,:,it]
    Zcc_elec[:,:,it]=run['A'].zone[jPFR_electric].zc[:,:,it]
    m=np.sqrt((Rcc_elec-Rcc_own)**2 + (Zcc_elec-Zcc_own)**2)
    a2=np.where(m==m.min())
    b2=a2[0]
    irPFR_elec=b2[0]
    b3=a2[1]
    ipPFR_elec=b3[0]
    pot_plasma_ip1=run['A'].zone[j].field['phi'][ir,ip+1,it]
    pot_plasma_ip0=run['A'].zone[jPFR_electric].
    field['phi'][irPFR_elec,ipPFR_elec,it]
    Rc_ip1=run['A'].zone[j].Rc[ir,ip+1,it]
    Rc_ip0=run['A'].zone[jPFR_electric].
    Rc[irPFR_elec,ipPFR_elec,it]
    phic_ip1=run['A'].zone[j].phic[it]
    phic_ip0=run['A'].zone[jPFR_electric].phic[it]
    zc_ip1=run['A'].zone[j].zc[ir,ip+1,it]
    zc_ip0=run['A'].zone[jPFR_electric].
    zc[irPFR_elec,ipPFR_elec,it]
if not j in numberlist_elec2:
    pot_plasma_ip1=run['A'].zone[j].field['phi'][ir,ip+1,it]
    pot_plasma_ip0=run['A'].zone[j].field['phi'][ir,ip-1,it]
    Rc_ip1=run['A'].zone[j].Rc[ir,ip+1,it]
    Rc_ip0=run['A'].zone[j].Rc[ir,ip-1,it]
    phic_ip1=run['A'].zone[j].phic[it]
    phic_ip0=run['A'].zone[j].phic[it]
    zc_ip1=run['A'].zone[j].zc[ir,ip+1,it]
```

```
            zc_ip0=run['A'].zone[j].zc[ir,ip-1,it]
#-PFR-#
if j in numberlist3:
    Rcc_own=run['A'].zone[j].Rc[ir,ip,it]
    Zcc_own=run['A'].zone[j].zc[ir,ip,it]
    if ip==210:
        jSOL_electric=j-1
        Rcc_elec=np.zeros((51,610,16))
        Zcc_elec=np.zeros((51,610,16))
        Rcc_elec[:,:,it]=run['A'].zone[jSOL_electric].Rc[:,:,it]
        Zcc_elec[:,:,it]=run['A'].zone[jSOL_electric].zc[:,:,it]
        m=np.sqrt((Rcc_elec-Rcc_own)**2 + (Zcc_elec-Zcc_own)**2)
        a2=np.where(m==m.min())
        b2=a2[0]
        irSOL_elec=b2[0]
        b3=a2[1]
        ipSOL_elec=b3[0]
        pot_plasma_ip1=run['A'].zone[j].field['phi'][ir,ip-1,it]
        pot_plasma_ip0=run['A'].zone[jSOL_electric].
        field['phi'][irSOL_elec,ipSOL_elec,it]
        Rc_ip1=run['A'].zone[j].Rc[ir,ip-1,it]
        Rc_ip0=run['A'].zone[jSOL_electric].
        Rc[irSOL_elec,ipSOL_elec,it]
        phic_ip1=run['A'].zone[j].phic[it]
        phic_ip0=run['A'].zone[jSOL_electric].phic[it]
        zc_ip1=run['A'].zone[j].zc[ir,ip-1,it]
        zc_ip0=run['A'].zone[jSOL_electric].
        zc[irSOL_elec,ipSOL_elec,it]
    if ip==0:
        jSOL_electric=j-1
        Rcc_elec=np.zeros((51,610,16))
        Zcc_elec=np.zeros((51,610,16))
        Rcc_elec[:,:,it]=run['A'].zone[jSOL_electric].Rc[:,:,it]
        Zcc_elec[:,:,it]=run['A'].zone[jSOL_electric].zc[:,:,it]
        m=np.sqrt((Rcc_elec-Rcc_own)**2 + (Zcc_elec-Zcc_own)**2)
        a2=np.where(m==m.min())
        b2=a2[0]
        irSOL_elec=b2[0]
        b3=a2[1]
        ipSOL_elec=b3[0]
        pot_plasma_ip1=run['A'].zone[jSOL_electric].
        field['phi'][irSOL_elec,ipSOL_elec,it]
        pot_plasma_ip0=run['A'].zone[j].field['phi'][ir,ip+1,it]
        Rc_ip1=run['A'].zone[jSOL_electric].
        Rc[irSOL_elec,ipSOL_elec,it]
        Rc_ip0=run['A'].zone[j].Rc[ir,ip+1,it]
        phic_ip1=run['A'].zone[jSOL_electric].phic[it]
        phic_ip0=run['A'].zone[j].phic[it]
        zc_ip1=run['A'].zone[jSOL_electric].
        zc[irSOL_elec,ipSOL_elec,it]
        zc_ip0=run['A'].zone[j].zc[ir,ip+1,it]
    if not j in numberlist_elec3:
        pot_plasma_ip1=run['A'].zone[j].field['phi'][ir,ip-1,it]
        pot_plasma_ip0=run['A'].zone[j].field['phi'][ir,ip+1,it]
```

```
        Rc_ip1=run['A'].zone[j].Rc[ir,ip-1,it]
        Rc_ip0=run['A'].zone[j].Rc[ir,ip+1,it]
        phic_ip1=run['A'].zone[j].phic[it]
        phic_ip0=run['A'].zone[j].phic[it]
        zc_ip1=run['A'].zone[j].zc[ir,ip-1,it]
        zc_ip0=run['A'].zone[j].zc[ir,ip+1,it]
xc_ip1=Rc_ip1*np.cos(phic_ip1*(180./pi))
xc_ip0=Rc_ip0*np.cos(phic_ip1*(180./pi))
yc_ip1=Rc_ip1*np.sin(phic_ip1*(180./pi))
yc_ip0=Rc_ip0*np.sin(phic_ip1*(180./pi))
mod_ip=np.sqrt((xc_ip1-xc_ip0)**2 + (yc_ip1-yc_ip0)**2
+ (zc_ip1-zc_ip0)**2)

#Potential along radial line of sight
#-CORE-#
if j in numberlist1:
    Rcc_own=run['A'].zone[j].Rc[ir,ip,it]
    Zcc_own=run['A'].zone[j].zc[ir,ip,it]
    if ir==13:
        jSOL_electric=j+1
        Rcc_elec=np.zeros((51,610,16))
        Zcc_elec=np.zeros((51,610,16))
        Rcc_elec[:,:,it]=run['A'].zone[jSOL_electric].Rc[:,:,it]
        Zcc_elec[:,:,it]=run['A'].zone[jSOL_electric].zc[:,:,it]
        m=np.sqrt((Rcc_elec-Rcc_own)**2 + (Zcc_elec-Zcc_own)**2)
        a2=np.where(m==m.min())
        b2=a2[0]
        irSOL_elec=b2[0]
        b3=a2[1]
        ipSOL_elec=b3[0]
        pot_plasma_ir1=run['A'].zone[jSOL_electric].
        field['phi'][irSOL_elec,ipSOL_elec,it]
        pot_plasma_ir0=run['A'].zone[j].field['phi'][ir-1,ip,it]
        Rc_ir1=run['A'].zone[jSOL_electric].
        Rc[irSOL_elec,ipSOL_elec,it]
        Rc_ir0=run['A'].zone[j].Rc[ir-1,ip,it]
        phic_ir1=run['A'].zone[jSOL_electric].phic[it]
        phic_ir0=run['A'].zone[j].phic[it]
        zc_ir1=run['A'].zone[jSOL_electric].
        zc[irSOL_elec,ipSOL_elec,it]
        zc_ir0=run['A'].zone[j].zc[ir-1,ip,it]
    if not j in numberlist_elec4:
        pot_plasma_ip1=run['A'].zone[j].field['phi'][ir+1,ip,it]
        pot_plasma_ip0=run['A'].zone[j].field['phi'][ir-1,ip,it]
        Rc_ir1=run['A'].zone[j].Rc[ir+1,ip,it]
        Rc_ir0=run['A'].zone[j].Rc[ir-1,ip,it]
        phic_ir1=run['A'].zone[j].phic[it]
        phic_ir0=run['A'].zone[j].phic[it]
        zc_ir1=run['A'].zone[j].zc[ir+1,ip,it]
        zc_ir0=run['A'].zone[j].zc[ir-1,ip,it]
#-SOL-#
if j in numberlist2:
    Rcc_own=run['A'].zone[j].Rc[ir,ip,it]
    Zcc_own=run['A'].zone[j].zc[ir,ip,it]
```

```
    if ir==0:
        jCORE_electric=j-1
        Rcc_elec=np.zeros((14,399,16))
        Zcc_elec=np.zeros((14,399,16))
        Rcc_elec[:,:,it]=run['A'].zone[jCORE_electric].Rc[:,:,it]
        Zcc_elec[:,:,it]=run['A'].zone[jCORE_electric].zc[:,:,it]
        m=np.sqrt((Rcc_elec-Rcc_own)**2 + (Zcc_elec-Zcc_own)**2)
        a2=np.where(m==m.min())
        b2=a2[0]
        irCORE_elec=b2[0]
        b3=a2[1]
        ipCORE_elec=b3[0]
        pot_plasma_ir1=run['A'].zone[jCORE_electric].
        field['phi'][irCORE_elec,ipCORE_elec,it]
        pot_plasma_ir0=run['A'].zone[j].field['phi'][ir-1,ip,it]
        Rc_ir1=run['A'].zone[jCORE_electric].
        Rc[irCORE_elec,ipCORE_elec,it]
        Rc_ir0=run['A'].zone[j].Rc[ir-1,ip,it]
        phic_ir1=run['A'].zone[jCORE_electric].phic[it]
        phic_ir0=run['A'].zone[j].phic[it]
        zc_ir1=run['A'].zone[jCORE_electric].
        zc[irCORE_elec,ipCORE_elec,it]
        zc_ir0=run['A'].zone[j].zc[ir-1,ip,it]
    if not j in numberlist_elec5:
        pot_plasma_ip1=run['A'].zone[j].field['phi'][ir+1,ip,it]
        pot_plasma_ip0=run['A'].zone[j].field['phi'][ir-1,ip,it]
        Rc_ir1=run['A'].zone[j].Rc[ir+1,ip,it]
        Rc_ir0=run['A'].zone[j].Rc[ir-1,ip,it]
        phic_ir1=run['A'].zone[j].phic[it]
        phic_ir0=run['A'].zone[j].phic[it]
        zc_ir1=run['A'].zone[j].zc[ir+1,ip,it]
        zc_ir0=run['A'].zone[j].zc[ir-1,ip,it]
#-PFR-#
if j in numberlist3:
    pot_plasma_ir1=run['A'].zone[j].field['phi'][ir+1,ip,it]
    pot_plasma_ir0=run['A'].zone[j].field['phi'][ir-1,ip,it]
    Rc_ir1=run['A'].zone[j].Rc[ir+1,ip,it]
    Rc_ir0=run['A'].zone[j].Rc[ir-1,ip,it]
    phic_ir1=run['A'].zone[j].phic[it]
    phic_ir0=run['A'].zone[j].phic[it]
    zc_ir1=run['A'].zone[j].zc[ir+1,ip,it]
    zc_ir0=run['A'].zone[j].zc[ir-1,ip,it]



xc_ir1=Rc_ir1*np.cos(phic_ir1*(180./pi))
xc_ir0=Rc_ir0*np.cos(phic_ir1*(180./pi))
yc_ir1=Rc_ir1*np.sin(phic_ir1*(180./pi))
yc_ir0=Rc_ir0*np.sin(phic_ir1*(180./pi))
mod_ir=np.sqrt((xc_ir1-xc_ir0)**2 + (yc_ir1-yc_ir0)**2
+ (zc_ir1-zc_ir0)**2)
#Gradient calculation in toroidal coordinates
if np.abs(pot_plasma_ir1)>np.abs(pot_plasma_ir0):
    E[0]=-(pot_plasma_ir1-pot_plasma_ir0)/mod_ir
```

```
else:
    E[0]=-(pot_plasma_ir0-pot_plasma_ir1)/mod_ir
if np.abs(pot_plasma_ip1)>np.abs(pot_plasma_ip0):
    E[1]=-(pot_plasma_ip1-pot_plasma_ip0)/mod_ip
else:
    E[1]=-(pot_plasma_ip0-pot_plasma_ip1)/mod_ip
if np.abs(pot_plasma_it1)>np.abs(pot_plasma_it0):
    E[2]=-(pot_plasma_it1-pot_plasma_it0)/mod_ir
else:
    E[2]=-(pot_plasma_it0-pot_plasma_it1)/mod_ir
#Change of coordinate system to cartesians
a_radius=1.6
E_c[0]=(a_radius+E[0]*np.cos(E[1]))*np.cos(E[2])
E_c[1]=(a_radius+E[0]*np.cos(E[1]))*np.sin(E[2])
E_c[2]=E[0]*np.sin(E[1])
#Force evaluation
Fe=Z0*qe*E_c
Fe_mod=np.sqrt((Fe[0])**2+(Fe[1])**2+(Fe[2])**2)
if math.isnan(Fe_mod)==True:
    Fe=np.array([0.,0.,0.])
if Fe_mod>1:
    Fe=np.array([0.,0.,0.])

Ftot=Fd+Fs+Fa+Fe+Fr+Fg
Vd=Vd+(dt/Md)*Ftot
Ri1=Ri
Ri=Ri+(Vd*dt)    #Finite diferences for particle position [x,y,z]
Ri2=Ri

Vd_array[n]=np.sqrt((Vd[0])**2+(Vd[1])**2+(Vd[2])**2)
Fd_array[n]=np.sqrt((Fd[0])**2+(Fd[1])**2+(Fd[2])**2)
Fs_array[n]=np.sqrt((Fs[0])**2+(Fs[1])**2+(Fs[2])**2)
Fg_array[n]=np.sqrt((Fg[0])**2+(Fg[1])**2+(Fg[2])**2)
Fe_array[n]=np.sqrt((Fe[0])**2+(Fe[1])**2+(Fe[2])**2)
Fr_array[n]=np.sqrt((Fr[0])**2+(Fr[1])**2+(Fr[2])**2)
Fa_array[n]=np.sqrt((Fa[0])**2+(Fa[1])**2+(Fa[2])**2)
Ftot_array[n]=np.sqrt((Ftot[0])**2+(Ftot[1])**2+(Ftot[2])**2)
```

Note that some lines might have been split for displaying purposes which does not assure a complete functioning of the code if directly copied from this document. If the reader wishes to check/use the code in detail, is referred to contact the author.

# Bibliography


[1] U. Samm, M. Tokar', and B. Unterberg, Journal of Nuclear Materials **241-243**, 827 (1997).

[2] B. Bray, W. West, and D. Rudakov, Journal of Nuclear Materials **390-391**, 96 (2009).

[3] M. Sertoli *et al.*, Journal of Nuclear Materials **463**, 837 (2015).

[4] S. Iwashita *et al.*, Fusion Engineering and Design **88**, 28 (2013).

[5] K. Saito *et al.*, Journal of Nuclear Materials **363**, 1323 (2007).

[6] A. Ekedahl *et al.*, Journal of Nuclear Materials **390**, 806 (2009).

[7] J. Coenen *et al.*, Journal of Nuclear Materials **438**, S27 (2013).

[8] L. C. Cadwallader, *Dust Combustion Safety Issues For Fusion Applications* (Idaho National Engineering and Environmental Laboratory, Idaho, **2003**).

[9] Y. Shimomura, Journal of Nuclear Materials **363-365**, 467 (2007), plasma-Surface Interactions-17.

[10] M. Bacharis, M. Coppins, W. Fundamenski, and J. E. Allen, Plasma Physics and Controlled Fusion **54**, 085010 (2012).

[11] Y. Tanaka *et al.*, Journal of Nuclear Materials **415**, S1106 (2011), proceedings of the 19th International Conference on Plasma-Surface Interactions in Controlled Fusion.

[12] L. Vignitchouk, P. Tolias, and S. Ratynskaia, Plasma Physics and Controlled Fusion **56**, 095005 (2014).

[13] J. Sharpe, D. Petti, and H.-W. Bartels, Fusion Engineering and Design **63 - 64**, 153 (2002).

[14] O. Ishihara and N. Sato, Plasma Science, IEEE Transactions on **29**, 179 (2001).

[15] V. N. Tsytovich, N. Sato, and G. E. Morfill, New Journal of Physics **5**, 43 (2003).

[16] H. M. Mott-Smith and I. Langmuir, Phys. Rev. **28**, 727 (1926).

[17] J. E. Allen, Physica Scripta **45**, 497 (1992).

[18] R. D. Smirnov *et al.*, Plasma Physics and Controlled Fusion **49**, 347 (2007).

[19] S. I. Krasheninnikov, R. D. Smirnov, and D. L. Rudakov, Plasma Physics and Controlled Fusion **53**, 083001 (2011).

[20] E. C. Whipple, Reports on Progress in Physics **44**, 1197 (1981).

[21] S. Dushman, Rev. Mod. Phys. **2**, 381 (1930).

[22] C. Herring and M. H. Nichols, Rev. Mod. Phys. **21**, 185 (1949).

[23] P. Tolias, Plasma Physics and Controlled Fusion **56**, 123002 (2014).

[24] S. A. Schwarz, Journal of Applied Physics **68**, 2382 (1990).

[25] A. Dinsdale, Calphad **15**, 317 (1991).

[26] C. F. Boheren and D. R. Huffman, *Absorption and Scattering of Light by Small Particles* (John Wiley & Sons, New York, **1983**).

[27] M. A. Ordal *et al.*, Appl. Opt. **22**, 1099 (1983).

[28] M. A. Ordal *et al.*, Appl. Opt. **24**, 4493 (1985).

[29] Y. Tanaka, R. D. Smirnov, A. Y. Pigarov, and M. Rosenberg, Physics of Plasmas **15**, 1 (2008).

[30] P. Desai, T. Chu, H. James, and C. Ho, JPCRD **13**, 1069 (1984).

[31] G. Shah, Kodaikanal Observatory Bulletins Series A **2**, 42 (1977).

[32] M. Rosenberg, R. D. Smirnov, and A. Y. Pigarov, Journal of Physics D: Applied Physics **41**, 015202 (2008).

[33] H. D. Hagstrum, Phys. Rev. **96**, 336 (1954).

[34] E. R. Plante and A. B. Sessoms, Journal of research of the Notional Bureau of Standards **77A**, 237 (1972).

[35] D. R. L. (ed), *CRC Handbook of Chemistry and Physics* (CRC Press, Boca Raton, **2005**).

[36] I. H. Hutchinson, Plasma Physics and Controlled Fusion **48**, 185 (2006).

[37] K. Krieger *et al.*, Physica Scripta **2011**, 014067 (2011).

[38] K. Krieger *et al.*, Journal of Nuclear Materials **415**, S297 (2011), proceedings of the 19th International Conference on Plasma-Surface Interactions in Controlled Fusion.

[39] S. I. Krasheninnikov, Y. Tomita, R. D. Smirnov, and R. K. Janev, Physics of Plasmas **11**, 3141 (2004).

[40] C. Davisson and L. H. Germer, Phys. Rev. **20**, 300 (1922).

[41] Y. Feng *et al.*, Contributions to Plasma Physics **44**, 57 (2004).

[42] T. Lunt *et al.*, Journal of Nuclear Materials **463**, 744 (2015).

[43] R. D. Smirnov *et al.*, Plasma Physics and Controlled Fusion **51**, 055017 (2009).

[44] S. Fiedler *et al.*, Journal of Nuclear Materials **266**, 1279 (1999).

[45] S. I. Krasheninnikov, A. Y. Pigarov, R. D. Smirnov, and T. K. Soboleva, Contributions to Plasma Physics **50**, 410 (2010).

[46] G. N. Watson, *Theory of Bessel Functions* (Cambridge University Press, London, **1922**).

# Acknowledgements

I would like to thank the Erasmus Mundus Fusion EP consortium for giving me the opportunity to join this master programm and develop a professional career in fusion nuclear physics.

I would also like to thank my supervisors, Dr. Brochard and Mr. Shalpegin for their guidance during these four months and kind patience.

Last but not least I would like to thank my family for their unconditional and infinite support.

# Declaration in lieu of oath

Herewith I declare in lieu of oath that I have prepared this thesis exclusively with the help of my scientific teachers and the means quoted by them.

Nancy, the 06/07/2015

Guillermo Suárez López

# Copyright Agreement

I hereby grant the FUSION-EP consortium the non-exclusive right to publish this work.

I declare that this work is free of copyright claims of third parties.

Nancy, the 06/07/2015

Guillermo Suárez López