Plasma parameters from the Rosetta LAP instrument

Claes Weyde
Swedish Institute of Space Physics, Uppsala

November 27, 2006

Abstract

The focus of this thesis is the extraction of plasma parameters using the Langmuir probes mounted on the Rosetta spacecraft. These probes, developed by the Swedish Institute of Space Physics, Uppsala, measures properties of the plasma environment such as electron temperature and ion density. Using Matlab, routines are developed to analyse the data.

We investigate the electron current produced by the photoelectric effect on the spacecraft and determine a model to use in describing it mathematically. We notice a remnant current, whose origin is unaccounted for. Two hypotheses are given; a leak-current from the probe to the spacecraft or influence from the spacecraft body on the plasma being measured by the probes.

We use the analysis routines on the first Earth fly-by performed by Rosetta and compare it to independent sources, the Advanced Composition Explorer (ACE) and anther instrument on Rosetta, the Mutual Impedance Probe (MIP). The physical parameters extracted (with the exclusion of the electron temperature) by the routines seem to be consistent with theory and with independent sources.

Sammanfattning

Rosetta är ett ESA-finansierat rymdprojekt vars mål är att utforska fysiken hos kometen 67 P/Churyumov-Gerasimenko, bla. genom att placera en landare på dess yta. Ombord på rymdfarkosten Rosetta finns två s.k. Langmuir prober. Dessa är utvecklade av institutet för rymdfyisk i Uppsala och är i princip små väderstationer som är anpassade för att mäta olika parametrar hos ett plasma, parametrar såsom jondensitet och elektrontemperatur. Detta examensarbete fokuserar på att från de data som Langmuir proberna på Rosetta samlar, extrahera de olika plasmaparametrarna. Matlab rutiner har utvecklats i syfte att sköta denna analys automatiskt.

Vi undersöker den elektronström som produceras av den fotoelektriska effekten och bestämmer den model som enklast och bäst beskriver strömmarna matematiskt. Vi upptäcker en ström som inte förväntas. Två hypoteser ges; en läckström från proberna till rymdfarkosten och/eller rymdfarkostkroppens potentials påverkan på det omgivande plasmat.

Vi använder slutligen våra analysrutiner på den första jordförbiflygningen som Rosetta genomförde i mars 2005 och jämför våra resultat med oberoende källor, dels en obereonde rymdfarkost, the Advanced Composition Explorer (ACE), och dels ett annat instrument på Rosetta, the Mutual Impedence Probe (MIP). De fysikaliska parametrar vi erhåller med analysrutinerna, exklusive elektrontemperaturen, verkar vara konsistenta med teori och oberoende källor.

Contents

1 Introduction **9**
1.1 The Rosetta mission . . . 9
1.2 Instrumentation - What is measured? . . . 10
1.3 The plasma surrounding Earth . . . 11
1.3.1 Rosettas trajectory near Earth . . . 13

2 Plasma physics **15**
2.1 Plasma - the fourth state of matter . . . 15
2.2 Collective behaviour . . . 15
2.3 Debye shielding and quasineutrality . . . 16
2.4 Particle motion . . . 17

3 Probe theory **19**
3.1 The Langmuir Probe . . . 19
3.2 Orbital motion limited theory . . . 20
3.3 Bias-, probe- and spacecraft potentials . . . 21
3.4 Probe currents . . . 22
3.4.1 Electron and ion currents . . . 23
3.4.2 Ion Current for supersonic flow . . . 24
3.4.3 Photoelectron current . . . 24
3.4.4 Putting it all together . . . 25

4 Using the probes in space **27**
4.1 Modes of operation . . . 27

4.2 Bias sweeps - the CV curve 28
4.3 Complications 29
4.3.1 $\mathbf{v} \times \mathbf{B}$ effects 29
4.3.2 Wake effects 30
4.3.3 Magnetic effects 31

5 Investigating the photocurrent 33
5.1 Introduction 33
5.2 Slope of the sweeps 34
5.3 Models 37
5.4 Performing the investigation 37
5.4.1 Comparing the models 38
5.5 The LAP dance 41
5.6 Conclusion 45

6 Automatic fit routine 47
6.1 Parameterisation 47
6.2 Algorithm 48
6.2.1 Data handling 49
6.2.2 The spacecraft potential 49
6.2.3 Walkthrough 51
6.3 Functions 55
6.3.1 extraction.m 56
6.3.2 fitting2.m 57
6.3.3 preliminaries.m 59
6.3.4 fit_data2.m 61
6.3.5 model.m 64

7 Results and discussion 65
7.1 The first flyby of Earth 65
7.2 V_{sc} and the plasma density 66
7.2.1 Spacecraft potential dependence on density 66
7.2.2 Comparison to ACE 66

7.2.3 Comparison V_{sc} to n from LAP 69
7.3 The spacecraft potentials 70
7.4 Comparing the data - MIP 72
7.5 Conclusion 73
7.5.1 Outlook 73
7.6 Acknowledgements 74

A User's guide to Grafical 77
A.1 Appearance and plots 78
A.1.1 Loading data 78
A.1.2 The fitting panel 80
A.1.3 The position and velocity panel 83
A.1.4 Save Data 84
A.1.5 Set spacecraft potential manually 87

B The software routines 89
B.1 Main program 89
B.1.1 grafical.m 89
B.1.2 Rosetta_info.m 107
B.2 Handling the data 108
B.2.1 get_datagraf.m 108
B.2.2 adjust_data.m 111
B.2.3 remove_dat.m 112
B.2.4 search_dat.m 113
B.2.5 save_data.m 114
B.3 Setting the position and velocity 117
B.3.1 set_r_v.m 117
B.3.2 get_traj.m 120
B.4 Fitting routines 126
B.4.1 extraction.m 126
B.4.2 fitting2.m 127
B.4.3 preliminaries.m 131

B.4.4 zero_cross.m 132
B.4.5 moving_average.m 133
B.4.6 find_scpot.m 134
B.4.7 d2.m 135
B.4.8 fit_single_e.m 136
B.4.9 determine_ecl.m 137
B.4.10 fit_data2.m 138
B.4.11 model.m 150
B.4.12 compute_c.m 155
B.4.13 weighting.m 156
B.4.14 choose_fit.m 157
B.4.15 errorest.m 158
B.4.16 compare_fits.m 158
B.4.17 par2phys2.m 160
B.4.18 get_I_parts.m 162

Chapter 1

Introduction

This master thesis is written with the purpose of looking at the data collected by the Rosetta spacecraft as it moves through the solar wind and especially when it approaches Earth, since this is a region where plasma parameters will change a lot. The idea is to see what really can be measured by the Langmuir probes on Rosetta and to write a software which will extract plasma parameters. Furthermore, this software should be used on the first Earth flyby to determine values of the plasma density and temperature. This extraction can then be used as a tool to evaluate the software as well as the performance of the probes.

Here we will introduce the Rosetta space mission as well as briefly look at the plasma close to Earth, in order to get an idea of what kind of parameters to expect.

1.1 The Rosetta mission

The Rosetta mission is a space mission by the *the European Space Agency* (ESA). Initially the goal of the mission was to rendezvous with Comet 46 P/Wirtanen, but due to postponement of the launch a new target had to be set. The choice fell upon Comet 67 P/Churyumov-Gerasimenko. The mission is an ambitious one; Rosetta will not only follow the comet (orbiting around it) for more than a year, it will even place a lander on its surface, providing the means of studying the nucleus of the comet in situ, something that has never been done before.

Launched the 2nd of March 2004 from Kourou, French Guiana, the Rosetta spacecraft (s/c) is currently speeding through space on way to the comet. The trip will take 10 years. An Earth flyby has already been performed (in March 2005) to be followed by two more and one flyby of Mars. The purpose of them is to provide the energy for the spacecraft to be able to attain the comet's velocity and to finally rendezvous in 2014. The flybys are great opportunities to test the instruments on board the s/c in at least semi-known environments, to see if everything is working properly and to investigate these environments further.

1.2 Instrumentation - What is measured?

Rosetta is carrying an exhaustive set of instruments, 16 instruments on the orbiter and another ten on the lander. Both remote and direct sensing techniques are used. Examples of the former are cameras and radio equipment. The *Langmuir Probe* (LAP) is an example of the latter and is the main emphasis of this report.

The goal of the mission is to get a better understanding of the origin of comets and the solar system. Since comets are among the oldest bjoects traversing space information gathered may help scientists understand the origin of our solar system. The LAP will investigate the plasma environment with regards to plasma density, temperature, flow speed etc. This will give information regarding the comets interaction with the solar wind. LAP is part of a group of instruments called the *Rosetta Plasma Consortium* (RPC) (Table 1.1), which all measure different aspects of the plasma close to the s/c. Used together, they will provide a good picture of the plasma environment surrounding 67 P/Churyumov-Gerasimenko.

Acronym	Name	Measures
LAP	Dual Langmuir Probe Instrument	Plasma density, temperature, flow speed, electric field < 8 kHz, etc.
MIP	Mutual Impedance Probe	Plasma density, temperature, flow speed, electric field 7 kHz – 3.5 MHz, etc.
MAG	Fluxgate Magnetometer	Magnetic field < 25 Hz
IES	Ion and Electron Sensor	Ion and electron spectra (energy and direction)
ICA	Ion Composition Analyzer	Ion spectra (energy, angle and mass)
PIU	Plasma Interface Unit	Coordination of the RPC sensors

Table 1.1: The Rosetta Plasma Consortium

Rosetta carries two Langmuir probes[1] built at the *Swedish Institute of Space Physics, Uppsala division* (IRF-U) in Uppsala. The probes are mounted on the tips of two deployable booms (Figure 1.1). Their shape is spherical with a diameter of 5 centimetres. The sensor is made of titanium and has a surface coating of titanium-nitride, giving it a golden colour.

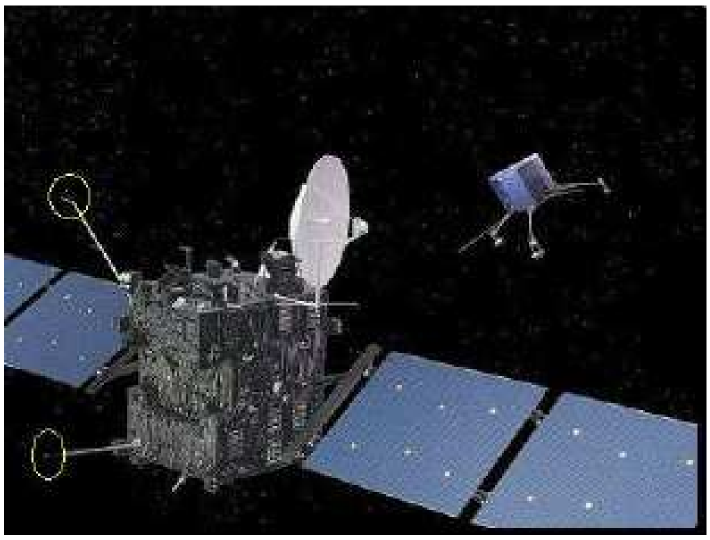


Figure 1.1: Artists impression of the Rosetta orbiter and lander. The LAP probes are encircled. Figure from ESA website[2]

1.3 The plasma surrounding Earth

Most of the data analysed in this thesis were gathered during the first Earth fly-by for reasons that will become apparent (see Chapter 7). For this reason it is important to have a good understanding of the plasma environment surrounding Earth. This section provides a brief overview of this vast topic[3].

The surroundings of Earth are very much dependent on the interplay between the solar wind[1] and the geomagnetic field. This interaction creates the *magnetosphere*, a region in which the magnetic field of Earth dominate the plasma dynamics. The boundary of the magnetosphere is known as the *magnetopause*. If we travel upstream of the magnetosphere, past the magnetopause, we will encounter a region called the *magnetosheath*. The magnetosheath separates the supersonic solar wind from the magnetosphere. The outer boundary of the magnetosheath, beyond which the solar wind flows freely is called the *bow shock*. This is the shock created when the supersonic solar wind encounters a subsonic obstacle.

[1]Basically ions and electrons travelling outward in the solar system at ∼500 km/s, emanating from the sun.

Closer to Earth there is a region of cold, dense plasma, stretching to about 3-6 Earth radii (R_E).[2] This region is known as the *plasmasphere*. The main ion species of the plasmasphere are protons (i.e. hydrogen ions), but it also contains small amounts of helium ions (He^+, 2-20%) and oxygen ions (O^+, 1-5%). The density of the plasma ranges from 10^4 cm^{-3} at the base of the plasmasphere to 10^2-10^3 cm^{-3} closer to the edge of the plasmasphere.

The plasmasphere is essentially an upward extension of the *ionosphere*, a region in the atmosphere where radiation (photons) and energetic particles ionise the otherwise neutral atmosphere. At the outer rim of the plasmasphere there is a sharp edge, where densities drop by a factor of 10-100 in a relatively short distance. This edge is called the *plasmapause*.

The plasmasphere co-rotates with the Earth. This means that the plasma in this region will have a drift velocity of about 0.6 km/s at Rosetta's point of closest approach. This is an effect that could become important when analysing the data, depending on the velocity of Rosetta as it travels through the plasmasphere (see Chapter 4.3). The magnetic field close to Earth can be modelled as a dipole field.

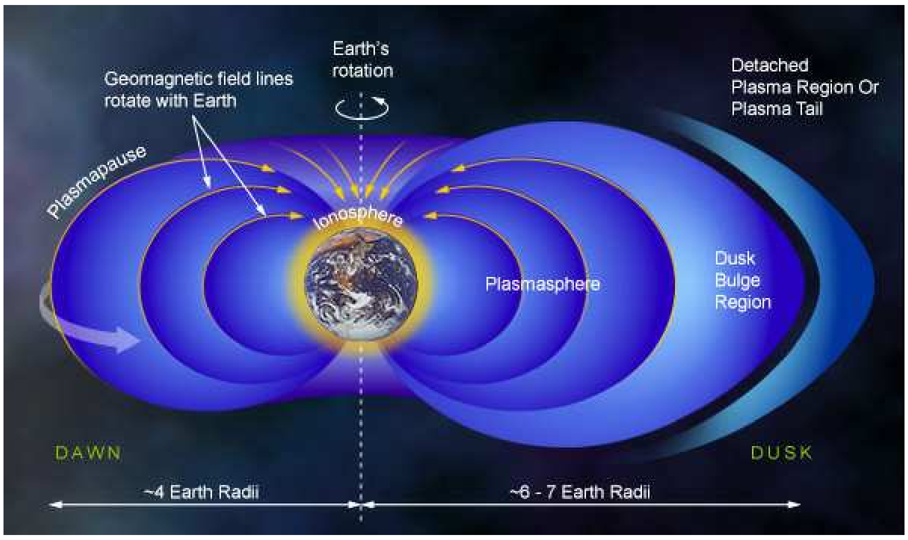


Figure 1.2: The Earth's plasmasphere[4].

Partly overlapping the plasmasphere one can also find the *radiation belts*, also

[2]One earth radius = 6371.2 km.

known as the *van Allen Belts.* These are regions where high energy (MeV) particles are trapped, and could possibly constitute a problem for the electronics on board Rosetta. After investigation[5], however, it was deemed safe to let the instruments be turned on and data was collected. This was fortunate as it provides data that can be analysed and then compared to known densities and temperatures around Earth.

1.3.1 Rosettas trajectory near Earth

Rosetta approached Earth from nightside, entered the magnetosphere through the geomagnetic tail, crossing the Van Allen belts. It went into the plasmasphere and out through the magnetopause and the bow shock on the morningside of the Earth. The closest approach was on the dayside of Earth at an altitude of about 2000 km above the planetary surface.

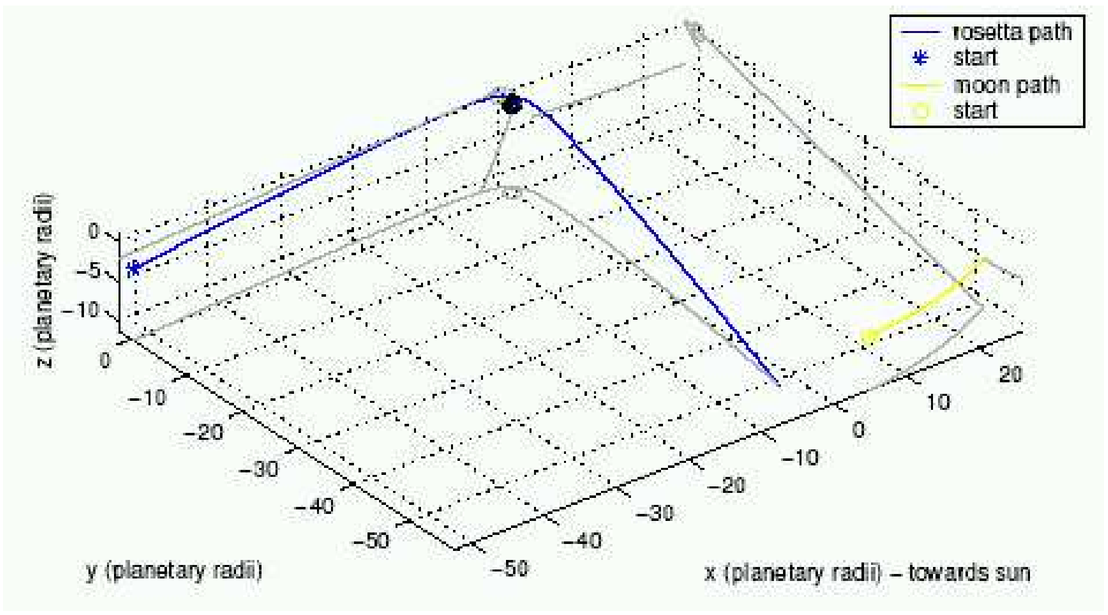


Figure 1.3: The trajectory of Rosetta for the first Earth flyby. Figure from Billvik 2005[5]

At altitudes closest to Earth one would expect a plasma density of about 10^3 cm^{-3} and electron temperatures between 0.1 and 0.5 eV.

Chapter 2

Plasma physics

2.1 Plasma - the fourth state of matter

Basically a plasma is a collection of electrons, ions and neutrals, i.e. a gas where some of, or all, the atoms are dissociated into electrons and ions. There may still be some disagreement amongst physicists as to whether a plasma really is a distinct state of matter or just a special type of gas, but here we will treat it as a fourth phase in which matter can exist. It is most similar to the gas phase in that it does not have a definite form or volume but there are still distinct differences between the two, notably in the way particles interact (see section 2.2 below), and it is not enough for a gas to be ionised in order for it to be a plasma (all gases have some degree of ionisation), so let us be more precise and define[6]:

a plasma is a quasineutral gas of charged and neutral particles which exhibits collective behaviour.

The words "quasineutral" and "collective behaviour" may need some clarification.

2.2 Collective behaviour

In an ordinary neutral gas, particles move undisturbed until they collide with other particles, hence the particle motion is controlled by collisions (gravitation is such a weak force that it is often negligible). In a plasma, however, the situation is different. Here the particles are *charged*, and as they move around, electric fields and currents will be generated. Because of this, particles in a plasma will influence each other at a distance. When saying that the particle motion exhibits *collective behaviour*, this is understood to mean that the motion of the particle is influenced by the total fields from all particles, not by individual particles close by.

If the plasma is very tenuous, the long-range electromagnetic forces dominate over the forces due to collisions, and we can speak of a “collisionless” plasma. This will be important when looking at particle motion, see section 2.4.

2.3 Debye shielding and quasineutrality

A most fundamental characteristic of the behaviour of any plasma is what is known as *Debye shielding* [6]. This is an innate ability of the plasma to shield out electric potentials in it (for example a biased probe). If a body is put into the plasma with a different potential than the plasma potential it will either attract ions or electrons depending on its potential. This will create a sheath around the body of particles with opposite charge to that of the body. This sheath will reduce the potential of the body as “seen” by particles outside of the sheath (much like electrons reduce the charge of the nucleus as seen by other charges far away from the atom). This phenomenon is known as *Debye shielding* and the characteristic thickness of the sheath is known as the *Debye length.* Consider the electrostatic potential generated by a single point particle with charge q,

$$V(r) = \frac{q}{4\pi\epsilon_0 r}, \tag{2.1}$$

where r is the distance to the particle and ϵ_0 is the vacuum permittivity. It can be shown [6] that the effective potential seen when in a plasma will be reduced to

$$V(r) = \frac{q}{4\pi\epsilon_0 r} e^{-\frac{r}{\lambda_D}}, \tag{2.2}$$

where λ_{D} is the Debye length, given by

$$\lambda_D = \sqrt{\frac{\epsilon_0 k_B T_e}{n_e e^2}} \tag{2.3}$$

where T_{e} is the electron temperature and n is the density of the electrons.

Now, if the dimensions of a system is much larger than λ_D the local potentials will be effectively shielded out. This means that in the plasma outside of the sheath the density of the ions and the electrons will be almost equal, the plasma is said to be *quasineutral*, i.e.

$$n_i \approx n_e. \tag{2.4}$$

Equation (2.4) will be very important in our parametrisation for the analysis of the data, see Appendix 6.

2.4 Particle motion

The particles constituting the plasma move in random thermal motion and under influences of electromagnetic forces. The plasma applicable to Rosetta is a very tenuous space plasma and it can be considered collissionless, thereby simplifying the theory considerably. Since there are so many particles interacting with each other a statistical approach is warranted. Even though there are no collisions a Maxwellian distribution of the particle motion is used. This distribution has worked well in the past when considering space plasmas and is a good choice on a probability basis. The random thermal velocity of the particles is defined as

$$v_j = \sqrt{\frac{k_B T_j}{m_j}} \tag{2.5}$$

where k_B is Boltzmanns constant and T_j and m_j are the temperature and mass of the particle species respectively. In addition to the random motion of the particles, they will generally move with a drift speed with respect to Rosetta, due to motion of the spacecraft as well as plasma flow.

In addition to the random motion and the drift, if the plasma is magnetised, the particles will gyrate around the magnetic field lines. If the magnetic field is uniform and static the particle equation of motion will become

$$m\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = q\mathbf{v} \times \mathbf{B} \tag{2.6}$$

where B is the magnetic field. Solving this, we see that the solution describes a simple harmonic oscillator, with the so called *cyclotron frequency*, defined as

$$\omega_c \equiv \frac{|q|B}{m} \ . \tag{2.7}$$

This gyration will have a radius defined by

$$r_L \equiv \frac{v}{\omega_c} = \frac{mv}{|q|B}, \tag{2.8}$$

where r_L is known as the *Larmor radius* and is a way to describe the circular orbit that the particles move in, around the magnetic field lines. Now, if in addition to the magnetic field, there was also an electric field through which the particles move, the equation of motion would become

$$m\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right), \tag{2.9}$$

Solving this, it is found that the electric field introduces a drift motion, called the *electric field drift*, given by

$$\mathbf{v}_E \equiv \frac{\mathbf{E} \times \mathbf{B}}{B^2} \ . \tag{2.10}$$

In addition to the cases mentioned above, the magnetic and electric fields can be non-uniform and time-varying. This will complicate things further and introduce many different kind of motions. For our purposes equation (2.5) for the random motion will be sufficient.

Chapter 3

Probe theory

In this Chapter we will introduce the Langmuir probe and its theory. In order to simplify our future work the Orbital Motion Limited (OML) theory will be described and we will consider under what circumstances it is applicable. Finally the different constituent currents to the probe will be discussed.

3.1 The Langmuir Probe

The Langmuir probe was invented by Nobel laureate Irving Langmuir (1881–1957). The device can be used to determine various properties of a plasma such as electron temperature and ion density. The probe can come in many shapes and sizes, for example plates, spheres or cylinders and one or several probes can be used together. The probe itself, however, is nothing more than an electrode which can be put into the plasma and fed by a power supply. Any kind of plasma can be investigated using the Langmuir probe but here we will focus on a probe mounted on a spacecraft, measuring space plasmas which are typically very tenuous (as low as a couple of particles per cm^3) compared to the type of plasmas encountered in the laboratory.

The principle behind the probe is very simple. Consider an electrode immersed in a plasma. By using a power supply it can be given different potentials with respect to the plasma. If the electrode potential is positive electrons will be attracted to the probe and if the potential is negative it will attract ions (repelling electrons). The s/c body constitutes the return electrode, completing the circuit. Now consider a positive potential; electrons will be attracted to the probe and a current will flow which can be measured. Under ideal conditions (section 3.2) the current I_p will be proportional to the number density of the electrons.

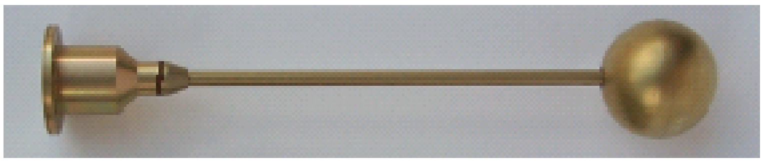


Figure 3.1: One of the two Langmuir probes on Rosetta

3.2 Orbital motion limited theory

The basic principles of the Langmuir probe, as briefly described above, may be very simple, but the theoretical description of its response is very complicated, and in order to do any kind of work we need to use approximations to the theory.

The problem of modelling the current to a probe in a plasma is a complicated one. However, there exist simplified expressions in the limits with regard to the density characteristics of the plasma. We have two limits; *sheath limited* (SL) or *orbital motion limited* (OML). In the SL case there is a very dense plasma and the Debye length will be much smaller than the probe radius

$$r_p \gg \lambda_D \quad \text{(SL)} \tag{3.1}$$

while in the OML case the plasma is very tenuous and the Debye length will be much larger than the probe radius

$$r_p \ll \lambda_D \quad \text{(OML)} \tag{3.2}$$

In Table 3.1, typical Debye lengths for the different regions which Rosetta will move through are listed. When comparing the Debye length in these regions to the 2.5 cm radii of the Langmuir probes on Rosetta it is clear that the OML theory can safely be used when in the solar wind and the magnetosphere. When passing through the plasmapasphere the Debye length comes close to the probe radius and there may possibly be sheath effects influencing the currents in this region. The simplifications and advantages of using the OML theory, however, are so great, that we will use the OML theory in all these cases knowing that the results in the plasmasphere may have an error due to the small Debye length in that region.

Plasma region	n_e [cm^{-3}]	T_e [eV]	λ_D [m]
Solar wind	5	10	10
Magnetosphere	10	500	50
Plasmasphere	10^4	1	0,07

Table 3.1: The Debye length for different plasmas

The OML theory applies to a low density, collisionless plasma absent of any magnetic fields. Since the plasma is so tenuous here, there is only a weak screening effect. What this means is that in the OML case the measured current will be proportional to the density since the particle motion does not affect the potential. Even if the plasma should at some instance not correspond to the OML-regime, it will still provide an upper bound for the current collected[7]. The spacecraft itself will also be surrounded by a sheath and this could potentially be a problem for the current collection of the probes. This is one of the reasons why they are mounted on booms, protruding a couple of meters from the spacecraft.

3.3 Bias-, probe- and spacecraft potentials

In order to make any sense of the data obtained it is crucial to know the *probe potential*, V_p, with respect to the plasma. The problem is that the probe potential will most often *not* be the same as the *bias potential*, $\mathrm{V_b}$, applied to the probe. This problem stems from the fact that the spacecraft will very seldom have a potential equal to zero when in space. This means that when applying a bias potential, the probe potential with respect to the plasma will become

$$V_p = V_b + V_{sc} \tag{3.3}$$

where V_sc is the potential of the spacecraft (with respect to the plasma).

In order to understand what potential the spacecraft will have, one need to understand which effects contribute to charge it. There are of course many different contributions to the charging and only a few can be discussed. The ones that are most important, for most of the time[1], are photoelectron emission and the mass difference between the ions and the electrons.

The spacecraft, being an isolated body in a plasma, would eventually become negatively charged because of the large difference in mass between the ions and

[1]There are effects which could be more important in some regions, but Rosetta will most likely spend quite a small amount of time in these regions, thus they will not be included here

the electrons[2]. If the temperatures are roughly equal, the mass difference result in a large difference in the velocity of the species (see equation (2.5)), the electrons will move much faster than the ions and even though the plasma as a whole is neutral, more electrons than ions will collide with the spacecraft, thus negatively charging it.

Now, the spacecraft may also be sunlit and then then the photoelectric effect will come into play (see Section 3.4.3). When emitted photoelectrons leave the spacecraft it will be charged positively. What charge the s/c eventually ends ups with depends on which effect is more dominant, and this will vary from region to region. Table 3.2 shows what $\mathrm{V_{sc}}$ to expect in some typical environments, taking these two effects into account.

Plasma region	Sign of V_{sc}
Solar wind	positive
Magnetosphere	positive
Plasmasphere	negative

Table 3.2: The spacecraft potential for different type of plasmas

Though an accurate estimate of the spacecraft potential requires a detailed model of the spacecraft-plasma interaction, typical values are a few volts negative in the plasmasphere, up to 10 V positive in the solar wind, and perhaps up to several tens of eV in the more tenuous magnetospheric regions. That being said, the sign of V_{sc}, given in table 3.2, is uncertain since a lot of different effects not discussed may influence the potential. For a numerical study of the charging of the Rosetta orbiter, refer to Roussel and Berthelier[8].

Knowing this it is clear that the spacecraft potential must be determined in order to know the real probe potential with respect to the plasma. Fortunately, this is possible to do from the bias sweeps. In section 6.2.2 we discuss how this problem is solved here.

3.4 Probe currents

There are several different currents, with different origins, contributing to the total current collected by the probes on Rosetta. To understand and interpret the LAP measurements it is important to have a good understanding of these different

[2]For example: $\frac{\mathrm{m_p}}{\mathrm{m_e}} \approx 1836$

constituent currents and how they are connected to the voltages, i.e. an expression for the electron and ion currents as functions of the probe voltage is needed and this is what we will discuss in the next section.

3.4.1 Electron and ion currents

Consider the orbital motion limited regime (section 3.2). Disregarding any magnetic fields in the vicinity, the theory developed by Langmuir and Mott-Smith in 1926 can be used[9].

The current to a body at plasma potential due to the random movement of the particles is

$$I_{j0} = A_p q_j n_j \sqrt{\frac{k_B T_j}{2\pi m_j}} \tag{3.4}$$

where A_p is the probe area and q_j, n_j, T_j and m_j refers to the j:th particle species charge, density, temperature and mass respectively.

The temperature in the expression above is in Kelvin (K). It is customary in plasma physics to give temperature in units of eV[3]. The conversion factor can be determined from 1 eV = $1.6{\cdot}10^{-19}$ J. With $k_\mathrm{B} = 1.38{\cdot}10^{-23}$ J/K we get

$$1 \text{ eV} \Leftrightarrow 11600^\circ \text{ K}. \tag{3.5}$$

The electrons and ions may have energies of several keV, but this high temperature does not mean a lot of heat since the density is often very low in a plasma and the total amount of heat transferred by collisions is not great. For convenience, in the coming set of equations, the temperature will be given in electron volts.

Depending on whether the probe potential is positive or negative the current will respond differently. A positive probe potential will give a linear response for the amount of collected electrons (more being collected the stronger the potential) and a negative one will give a exponential decay for the electrons.

$$I_e = \begin{cases} I_{e0}\left(1 + \frac{V_p}{T_e}\right), & V_p > 0 \\ I_{e0} e^{\frac{V_p}{T_e}}, & V_p < 0 \end{cases} \tag{3.6}$$

where T_e is the electron temperature (in eV). An expression of the same form, but with a sign change (since ions have opposite sign to the electrons) will hold for the

[3] $\mathrm{E_{av}} = \mathrm{K_b T}$

ions

$$I_i = \begin{cases} -I_{i0} e^{-\frac{V_p}{T_i}}, & V_p > 0 \\ -I_{i0}\left(1 - \frac{V_p}{T_i}\right), & V_p < 0 \end{cases} \tag{3.7}$$

where T$_i$ is the ion temperature. By convention, currents are counted positive when flowing from the probe to the plasma.

3.4.2 Ion Current for supersonic flow

The expression for the ion currents above is not quite applicable in all cases since the ions will pass the s/c with a supersonic velocity. In the solar wind the drift velocity is much higher than the s/c speed and the ions will move much faster than Rosetta, while in the magnetosphere of the Earth, Rosetta will move much faster than the ions. Either way, except for certain places in the magnetosphere where the ions are subsonic, the random thermal motion of the ions will be negligible compared to their average velocity with respect to Rosetta. Magnetic field effects on the ions will also be negligible (see Section 4.3.3) and we can approximate the ion current as the ram flux of the ions hitting Rosetta, i.e. the flux of ions hitting the probe due to the motion of the spacecraft. The ion current for supersonic flow will then become[7]

$$I_{i,ram} = -n_i e \pi r_p^2 \left(1 + \frac{2eV_p}{m_i v_{\mathrm{sc}}^2}\right) v_{\mathrm{sc}} \tag{3.8}$$

where r_p is the probe radius, m_i the ion mass and v_{sc} is the spacecraft velocity w.r.t. the plasma.

3.4.3 Photoelectron current

Whenever the probe is exposed to sunlight the photoelectric effect must be taken into consideration. A photon hitting the probe will give all its energy to an electron. Depending on whether this energy is great enough the electron may be emitted from the spacecraft. These photoelectrons will have a varying energy but most will lie in the region of a couple of eV. If the probe potential is below the energy of the emitted electron, the electron will be able to leave the probe and counteract the current collected. Thus an expression for this photoelectron current is also needed.

In a tenuous plasma the photoelectron current will be the dominant contribution to the probe current. The photoelectron current depends on the sunlit area of the probe, the irradiance from the sun and the surface properties, thus making it a complicated effect to take into account. Here we adopt the following form of the current, suitable for our needs[10]

$$I_{ph} = \begin{cases} -I_{ph,0}\left(1 + \frac{V_p}{T_{ph}}\right) e^{-\frac{V_p}{T_{ph}}}, & V_p > 0 \\ -I_{ph,0}, & V_p < 0 \end{cases} \tag{3.9}$$

where solar irradiation and surface properties of the probe determines the constants $I_{ph,0}$ and T_{ph}.

Photoelectrons emitted from the probe is not the only problem. The radiation from the sun will also emit photoelectrons from the spacecraft body, booms and solar panel. These electrons will contribute to a denser cloud of particles around the spacecraft and the probes will measure a higher current. If we assume that the cloud is fairly homogenous it would be reasonable to assume that the probes collect the photoelectrons much like they collected the electrons from the plasma (eq. (3.6)), the only difference being that now it is the bias potential (i.e. potential of the probe with respect to the s/c), not the probe potential in the expressions.

$$I_{sc} = \begin{cases} I_{s0}(1 + \frac{V_b}{T_s}), & V_b > 0 \\ I_{s0}e^{\frac{V_b}{T_s}}, & V_b < 0 \end{cases} \tag{3.10}$$

Refer to section 5.3 for a more detailed discussion of the mathematical model used for the photoelectron current in this work.

3.4.4 Putting it all together

The current that is measured will now be a sum of four parts, the electron current, the ion current, the emitted photoelectron current and the photoelectron current from the s/c to the probe (eqs. (3.6), (3.7)/(3.8), (3.9) and (3.10))

$$I = I_e + I_i + I_{ph} + I_{sc} \tag{3.11}$$

All these must be taken into account when analysing the data from the probe. It is important to understand that these expressions are only simplified models of the real current. There exist more complete models, but they are also more complicated and hence less useful.

Chapter 4

Using the probes in space

Here the different ways in which one can use the Langmuir probes on Rosetta are presented. We discuss the bias sweep in more detail and introduce the concept of the CV curve. The last section deals with some of the complications encountered when travelling through space and we determine whether these effects are large or if they can be neglected.

4.1 Modes of operation

There are several modes in which the probes can be used. Using only a single probe, three different modes can be utilised [IRF-U website[1]] (the parameters that each method can provide are also given)

- Bias sweeps: n_{e}, T_{e}, V_{i}, V_{sc}
- Continuous current measurement: $\frac{\partial n}{n}$, $\frac{n}{\sqrt{T_e}}$
- Continuous voltage measurement: V_{sc}

Refer to section 4.2 for a thorough breakdown of the bias sweep.

Doing a continuous current measurement means letting the probe have a fixed bias potential and measuring the collected current. In this way the current will vary proportional to the density of the plasma, but also depend on the plasma temperature and the spacecraft potential.

[1]http://www.space.irfu.se/rosetta/science.html

By continuous voltage measurement the probe is set at a fix bias current and by measuring its potential w.r.t. the s/c body a value of the spacecraft potential V_{sc} can be obtained.

Rosetta is equipped with two probes, adding to the possible information that can be gathered. By letting both probes have the same bias potential local fluctuations in density and temperature can be measured. The electric field between these two probes can be measured as well as the flow velocity of the plasma along the probe separation line (a kind of time-of-flight measurement).

4.2 Bias sweeps - the CV curve

A single probe can do a sweep from negative to positive bias potential thereby obtaining the current to the probe as a function of the potential. This kind of sweep will thus provide a so called current-voltage (CV) curve which can be used to obtain various parameters such as density and temperature. The CV-curve for a spherical probe can look something like figure 4.1. Positive current is defined as electrons reaching the probe (i.e. electrons travelling toward the probe).

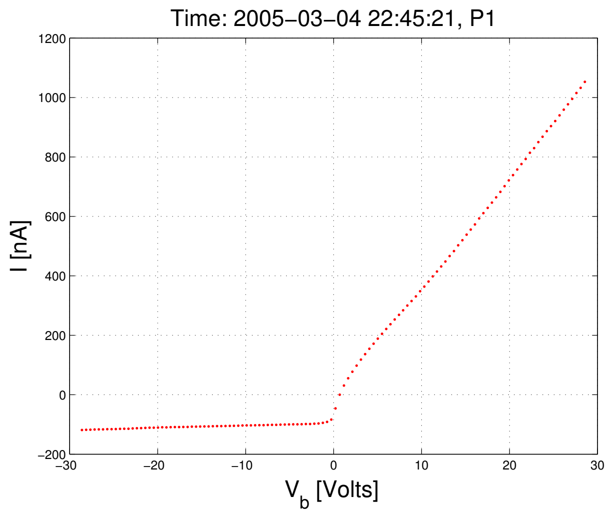


Figure 4.1: An actual probe sweep from Rosetta for a sunlit probe in the plasmasphere.

If we have a large negative bias, below $-V_{\mathrm{sc}}$ (see eq. 3.3), ions will be attracted to the probe while electrons are repelled. This region, on the negative x-axis in Figure 4.1 is called the *ion collection region*. If we increase the potential we will at some point have a situation where the current from the electrons is as big as the current from the ions (and thus the total current will cancel). This is called the floating potential. Increasing the bias potential even further will result in more electrons than ions reaching the probe. Thus the electron current will start to dominate, this is called the *electron collection region* and this is where the curve is linear on the positive potential side in Figure 4.1. There is quite a large difference in the absolute currents generated in the ion and electron collection regions respectively. This is because electrons have a much smaller mass than the ions and hence will move much faster.

4.3 Complications

The theory of section 3.4, being a great simplification, cannot perfectly model the plasma in which Rosetta is moving. There are many effects that will influence the measurements. This section is devoted to some of these. The ones considered here are:

- $\mathbf{v} \times \mathbf{B}$ effects
- Wake effects
- Magnetic effects

4.3.1 $\mathbf{v} \times \mathbf{B}$ effects

When Rosetta travels through space it will encounter magnetic fields, particularly when close to a planet such as Earth or Mars. General effects of magnetic fields are discussed briefly in section 4.3.3. There will, however, also be secondary effects of these B-fields. A charge moving through a magnetic field $\mathbf{B}$ with velocity $\mathbf{v}$ will experience an induced electric field $\mathbf{E}$ according to the formula

$$\mathbf{E} = \mathbf{v} \times \mathbf{B} \tag{4.1}$$

In free space, not close to any planets it is the speed of the solar wind that is important (since the solar wind has a speed around 400-500 km/s while Rosetta is moving much slower). Let's assume the velocity of the solar wind is perpendicular to the magnetic field (which may or may not be the case depending on how far out from the sun Rosetta is, however, taking the extreme case gives an upper bound on the effect), having a velocity of 400 km/s. The interplanetary magnetic field

(IMF) at 1 AU has a strength of about 10 nT.[2] The induced electric field will thus be

$$400 \cdot 10^3 \cdot 10^9 = 0.004 \text{ V/m}$$

Since the probes are located on booms about 1 m in length[3] the induced potential will be

$$0.004 \text{ V/m} \cdot 1 \text{ m} = 0.004 \text{ V} = 4 \text{ mV}$$

This is such a small effect that it can be ignored. When travelling inside the magnetopause, however, the surroundings change. It is now the Earth's magnetic field and Rosettas own speed which must be taken into account. If we treat the geomagnetic field as a perfect dipole it will have a form of

$$B_E(L) = B_0 L^{-3} \tag{4.2}$$

where B_0 is the equatorial field at 1 Earth radii. The closest to Earth that Rosetta comes is at 1960 km above the planetary surface (Section 1.3.1). At this altitude Rosetta will "feel" the strongest magnetic field. At closest approach Rosetta moved with a velocity relative to Earth of about 10 km/s [ESA website[4]]. Assuming that the plasma co-rotates with Earth it will have a velocity of about

$$\frac{2\pi \cdot (6371.2 + 1960)}{24 \cdot 3600} \text{ km/s} = 0.60586 \text{ km/s}$$

small enough to be neglected (this should be added to Rosettas velocity relative to the Earth, but it will not make any difference in the end). Calculating a geomagnetic field of

$$31 \cdot 10^{-6} \cdot (\frac{1960}{6378,16} + 1)^{-3} = 1.3875 \cdot 10^{-5} \text{ T}$$

the induced potential over a 1 m boom will become

$$1.3875 \cdot 10^{-5} \cdot 10 \cdot 10^3 = 0.13946 \text{ V}$$

This is more of a contribution than in interplanetary space but it is still small.

4.3.2 Wake effects

As Rosetta moves through space, a wake will be created behind the s/c, somewhat like that behind a boat on the sea. The random motion of the ions need quite some

[2] 1 astronomical unit = the mean Sun-Earth distance = $1.49597870 \cdot 10^{11}$ m
[3] the actual length of the booms are 2010 mm and 1395 mm, respectively
[4] http://www.esa.int

distance to fill the void created behind Rosetta, since their thermal velocity is much smaller than their drift velocity. The electrons, however, with their considerable thermal motion, will fill the void swiftly. This means that the wake is actually mostly an absence of ions. This is not entirely true if the Debye length would be in the order of the spacecraft size. Here we assume that there are almost no ions in the wake. This will mean that the plasma behind the s/c will be more tenuous with respect to ions than the actual plasma and now the probe will measure a lower current behind the s/c on the ion side. This effect can be seen in early data. When the probe is in the wake behind Rosetta almost no ions can be seen and therefore it is important to know Rosettas direction in space and the attitude of the probes when measuring.

4.3.3 Magnetic effects

The equations (3.6) and (3.7)/(3.8), for the electron and ion currents are only derived for unmagnetised plasmas. Rosetta will travel through plasmas which are magnetised. If the Larmor radius of the particles is large compared to the probe diameter there will be no problem, since on the scale of the probe the magnetic field will have a negligible influence on the movement of the particles. The Larmor radius, defined in Section 2.4, is given by

$$r_L = \frac{mv}{\mid q \mid B} = \frac{\sqrt{2mE_{kin}}}{\mid q \mid B} \tag{4.3}$$

It is seen that the Larmor radius is dependent on the mass as well as the velocity of the particles. This means that for a given energy, the electrons, having a smaller mass, will have a smaller Larmor radius than the ions. By calculating the electron Larmor radius and comparing it to the probe radius we can see whether the magnetic field's influence is negligible or not. Using values found inside the plasmasphere close to Earth[5] (this is where the magnetic field is strongest) it is seen that the Larmor radius is 10 cm, larger than the probes 2.5 cm radius. We therefore expect that the errors introduced by use of the unmagnetised probe theory will be small.

[5] $m_e = 10^{-30}$ kg, q = 10^{-19} C, B = 10 μT, v = 10 km/s

Chapter 5

Investigating the photocurrent

In which we perform an investigation of the photoelectron current emitted from the probes when in sunlight. Four different models describing the photoelectron current are compared and one model together with parameter values is chosen for our continued work. Finally we discuss some of the properties of the probes when in shadow and wake, based upon data from the so called LAP dance that Rosetta went through.

5.1 Introduction

The *photoelectric current* is important in sunlit plasmas. Since the theoretical current is a superposition of several different currents (see section 3.4), it is important to have a good model for the photoelectrons, in order to extract real physical parameters such as density and temperature. In Figure 5.1 below data from the *Ion Composition Analyzer* onboard Rosetta are presented[11]. These data were gathered during the earth flyby on 1 March 2005 and the figure show energy (on the y-axis) and density (by colour) of the observed ions as time progresses. Throughout the data a dark red “band” of ions with an energy of 1 keV can be seen. These are solar wind ions. There are several short intervals of low density (green or even white colour) in the data. For example a green interval can be seen at around 03:00 and a large white interval after 12:00. The low density at those times should mean that the current measured by the LAP is dominated by photoelectrons. Furthermore, at times the probes are located behind Rosetta, in the wake of the spacecraft, but still in sunlight. As was explained earlier (see section 4.3.2) the ion side of the sweep will be mostly photoelectrons under these circumstances and such sweeps will also provide valuable data for characterising the photoelectron current from the probes. It should be noted that the density

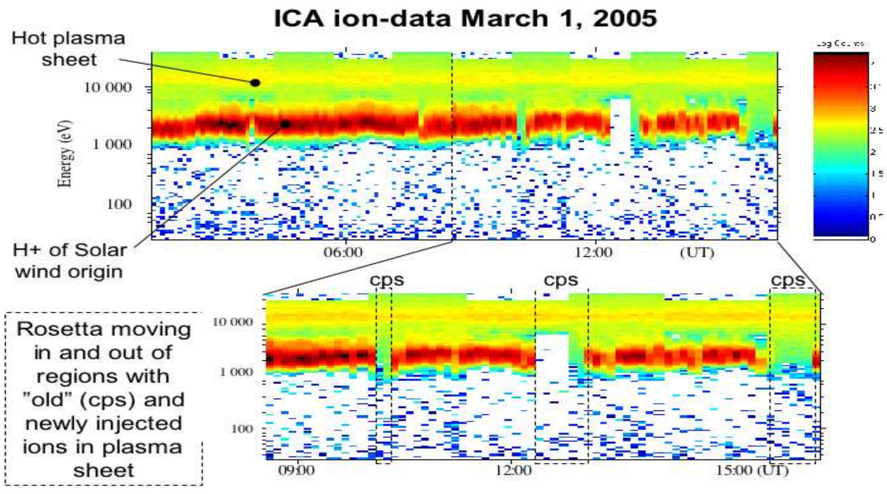


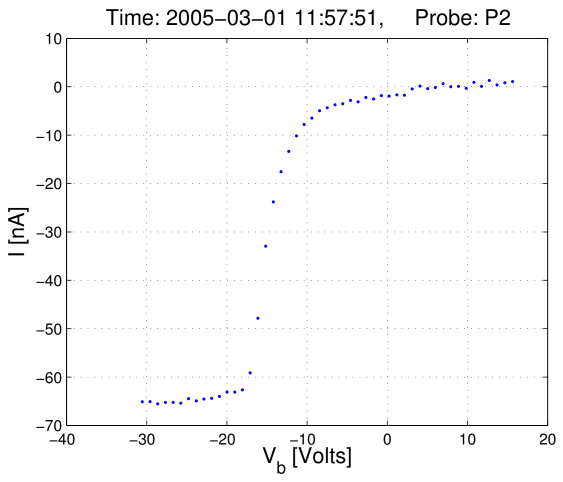
Figure 5.1: Data from the ICA-instrument onboard Rosetta (see Table 1.1)[11]. On the y-axis the energy of the gathered ions is given while colour designates density. The solar wind ions having an energy of 1 keV traverses the entire data. Notice the bands of low density in the 1 keV region at the approximate times 03:00, 08:00, 10:00, 12:00.

is quite low in the solar wind and photoemission will dominate at all times, but these intervals minimises other effects even further. By looking at data from these intervals, a qualitative model for the photoelectron current can be established.

The model used will be determined using data taken on 2005-03-01 at 11:55 to 12:30.

5.2 Slope of the sweeps

Figure 5.2, shows two sweeps in 1 March 2005, around 12:00. This is just before the first Earth flyby and Rosetta is in the geomagnetic tail of the Earth moving toward the planet. The plasma in the geomagnetic tail is very tenuous and there should be little contribution from plasma electrons.



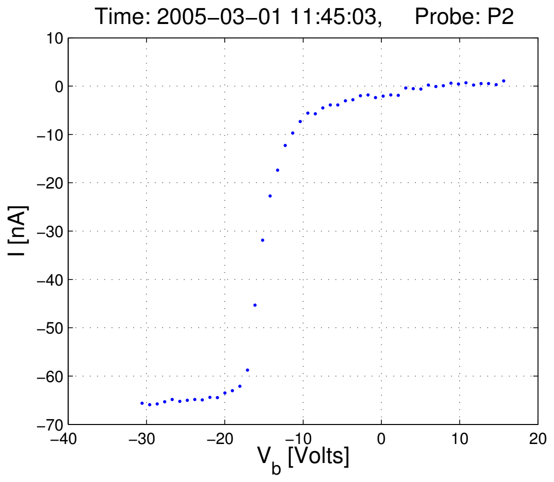


Figure 5.2: Two sweeps from probe 2 when Rosetta moves towards Earth, through the geomagnetic tail

Here probe 2 was in the s/c wake, but still sunlit providing a good opportunity to study the photoelectric current from the probe. Most of the current contribution come from photoelectrons emitted from the probes but there is still a small slope on both the ion and electron side. The reason for the slope on the ion side can not be due to ions from the plasma, since it is not dense enough to create such a slope, at least not with reasonable temperatures for the ions. Looking at Figure 5.2 we can calculate a slope of about

$$g = \frac{\mathrm{d}I}{\mathrm{d}V} = \frac{3\text{ nA}}{15\text{ V}} = 2 \cdot 10^{-10} = \frac{1}{5\text{ G}\Omega}$$

From equation (3.8), I_{i0} for the solar wind with a high density of ions can be calculated (the solar wind is actually denser than the geomagnetic tail in which we are, giving an upper bound on the current)

$$I_{i0}^{sw} = n_i e \pi r_p^2 v_{sc} = 30 \cdot 10^6 \cdot 1.6 \cdot 10^{-16} \cdot \pi \cdot 0.025^2 \cdot 400 \cdot 10^3 \text{ A } = 3.8\text{ nA}$$

Now an approximation of $\frac{\mathrm{d}I}{\mathrm{d}V}$ can be obtained by taking into account that $\frac{mv^2}{2} =$ 1 keV for solar wind ions. We get

$$\frac{\mathrm{d}I}{\mathrm{d}V} = \frac{3.8\text{ nA}}{1\text{ keV}} = 3.8 \cdot 10^{-12} = \frac{1}{0.2632\text{ T}\Omega}$$

Comparing this to the value $\frac{1}{5\text{ G}\Omega}$ we got from the plot, it is clear that the slope on the ion side cannot be due to plasma ions.

To perform the same calculation on the electrons we need to know what electron current to expect from the solar wind. Using eq. (3.4) we obtain

$$I_{e0}^{sw} = A_p q_e n_e \sqrt{\frac{k_B T_e}{2\pi m_e}} = 4\pi \cdot 0.025^2 \cdot 1.6 \cdot 10^{-19} \cdot 5 \cdot 10^6 \cdot \sqrt{\frac{5 \cdot 1.6 \cdot 10^{-19}}{2\pi \cdot 9.1 \cdot 10^{-31}}} = 2.35 \text{ nA}$$

This would give a slope of

$$g = \frac{\mathrm{d}I}{\mathrm{d}V} = \frac{2.35 \text{ nA}}{5V} \approx 0.5 \cdot 10^{-9} = \frac{1}{2 \text{ G}\Omega}$$

This value is in the same order as what we calculated from Figure 5.2 and plasma electrons cannot be ruled out as a source for the ion side slope. One explanation for this behaviour may be that the spacecraft potential actually is quite a bit higher (or lower if negative) than what is detected by the probes. The reason is the $\frac{1}{\mathrm{r}}$ decay of the Coloumb potential coupled with the fact that the booms on which the probes are located have a finite length, ending within the influence of the spacecraft body. What this means is that the plasma in which the probes are measuring is influenced by the spacecraft and its Coloumb potential. This will lead to a determination of the spacecraft potential being smaller than what it really is. This would mean that the part of the sweep refered to as the "ion side" may in fact still be the electron side since the spacecraft potential is higher (perhaps as much as twice the value determined).

However, electrons coming from the plasma is not the only possibility. The slope could also be due to photoelectrons coming from the spacecraft. Either way, had the current been only photoelectrons emitted from the probe we would not have a slope above zero, and because of the contribution of other currents (be it plasma electrons or spacecraft photoelectrons) on the electron side of the sweep that part of the data is less suited for our analysis.

Yet a third explanation is that this current could be due to a leakage current from the probe to the spacecraft. Normally the probe is supposed to be isolated so that no current can pass to the spacecraft (through the booms), but there is of course the possibility of for example contamination on the surface of the insulators separating conductive areas at different potential. Such contamination can give some conductivity, resulting in a small leak-current.

For the remainder of this report, the last explanation (i.e. the leak-current), has been assumed. This current was subtracted, simply by fitting the slope on the ion side. This way the slope on the ion side disappears, but on the electron side a small slope still remains, this is still unaccounted for but it is quite possible an effect of plasma- or spacecraft photoelectrons as discussed above.

5.3 Models

There exist several theories modelling the photoelectron current and here four models will be analysed and compared. The first one is also the simplest and has already been mentioned (equation (3.9)), for convenience it will be written here again

$$I_{ph} = \begin{cases} -I_{ph,0} e^{-\frac{V_p}{T_{ph}}}, & V_p > 0 \\ -I_{ph,0}, & V_p < 0 \ . \end{cases} \tag{5.1}$$

This is a very simple model with just an exponential decay in the current as the potential is increased (a high potential means that the photoelectrons will not have the energy to escape the probe, but will be pulled back in). This describes Boltzmann distributed emission from a plane surface. The second model considered is discussed by Grard[12], and introduces a second linear term in the positive probe potential, describing emission from a sphere instead of a plane. This will make the transition region between positive and negative potentials smooth, which may be more physical (though it may not follow the data any closer).

$$I_{ph} = \begin{cases} -I_{ph,0}\left(1 + \frac{V_p}{T_{ph}}\right) e^{-\frac{V_p}{T_{ph}}}, & V_p > 0 \\ -I_{ph,0}, & V_p < 0 \ . \end{cases} \tag{5.2}$$

The third model considered regards the photoelectron current as a superposition of two Boltzmann distributions with different temperatures[13]. The current can in that case be expressed as

$$I_{ph} = \begin{cases} -I^1_{ph,0} e^{-\frac{V_p}{T_{ph,1}}} - I^2_{ph,0} e^{-\frac{V_p}{T_{ph,2}}}, & V_p > 0 \\ -I^1_{ph,0} - I_{ph,0}, & V_p < 0 \ . \end{cases} \tag{5.3}$$

where the constants $\mathrm{I}^1_{ph,0}$, $\mathrm{I}^2_{ph,0}$, $\mathrm{T}_{ph,1}$ and $\mathrm{T}_{ph,2}$ refers to the two different currents. Finally model 2 and model 3 are put together and we get model 4

$$I_{ph} = \begin{cases} -I^1_{ph,0}\left(1 + \frac{V_p}{T_{ph,1}}\right) e^{-\frac{V_p}{T_{ph,1}}} - I^2_{ph,0}\left(1 + \frac{V_p}{T_{ph,2}}\right) e^{-\frac{V_p}{T_{ph,2}}}, & V_p > 0 \\ -I^1_{ph,0} - I_{ph,0}, & V_p < 0 \ . \end{cases} \tag{5.4}$$

5.4 Performing the investigation

The data is collected over a period of 30-40 minutes and there will therefore be fluctuations in the parameters values. The parameters derived in this investigation ($I_{\mathrm{ph},0}$ and T_{ph}) should not vary with the potential, but they do depend on external influences, notably the UV irradiation from the sun. Since Rosetta was at approximately the same distance from the sun during the interval of the data collection, these parameters should be more or less constant. The irradiation from the sun is not completly static even on this time scale and this will influence our result, but probably quite marginally. A possible error-source is how “clean” the data analysed are with regards to the surrounding plasma. We want the sweep to be as photo-dominated as possible. By only using intervals in the data that we know are dominated by photo-current these kind of effects should be minimised.

5.4.1 Comparing the models

The method used to compare the different models uses a semi-automatic approach. The software written fits the models to the data, given certain initial values, but the initial values can be chosen freely. The spacecraft potential can also be set manually.

There are mainly two problems with the fitting. First, the spacecraft potential is needed in order to use the models described in eqs. (5.3) and (5.4). The problems inherent with obtaining V_{sc} is described in 6.2.2, here it is enough to note that it will present a problem for the software at times. One solution is to set the potential manually (by eye) where the software fails.

The second problem is that the fitting routine used is the prebuilt MATLAB function *lsqcurvefit*. What it does is basically that it minimises the least-square error. By taking the norm of the residual we have a way of describing the error, and also a way of comparing the different models above. This is a non-linear fitting technique, suitable for non-linear models. The problem arises when considering the different parameters that can be used for the fitting. By trying different parameters, several local minima can be found. The fit will look nice, but the parameters need not be a good reflection of the actual state of the plasma. Here, values for the parameters can be set (either as initial guesses or as the final physical parameters) manually.

When comparing the different models it was soon realized that models 1 and 2 (i.e. the ones with only one population of electrons) were inferior to the other two. For high tolerances in the fit routine they worked well, but when reducing the tolerance, model 3 and 4 gave better results. This can actually be understood quite easily from the formulas themselves. It is always possible to get at least as good a fit with model 3 as with model 1, since model 1 is a *special case* of model 3. By discarding these models only model 3 and model 4 remain. To compare them two approaches were used:

- Automatic, with Vsc as free parameter but selected manually if its value seemed bad
- Manual, parameters chosen so that the fit looks good and the values are reasonable

The error (calculated as the root mean square difference of the fit to the actual data value, over the whole data interval) was calculated for each fit and compared between the models. As can be seen from figure 5.3 both models work well.

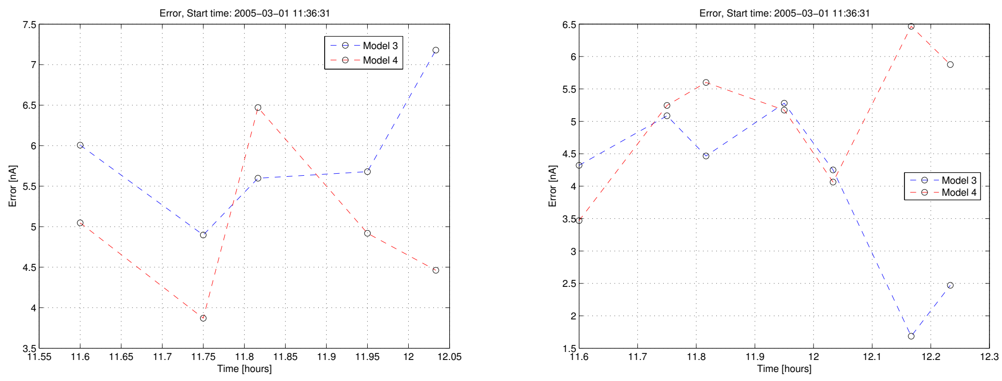


Figure 5.3: The total least-squares error divided by the number of data points for probe 1, models 3 and 4, left using the manual method and right using the automatic one. This is done for seven probe bias sweeps.

When looking at more dense plasmas the photoelectron current will be small in comparison to the ion and electron current. Furthermore, as it can change quite drastically over time it was decided that we should determine a fit that was good, and consider the values obtained for the temperature and the ratio between the currents in eqs. (5.1) - (5.4) as fixed. By doing this, the absolut value of the current can still be varied when fitting data later on, but we have a fix on the individual parameters that should be quite good[1]. This method will of course introduce an error but it should only influence the total fit marginally while at the same time reduce the number of independent variables for the fitting algorithms to work with by three.

When using fixed physical parameters (see Figure 5.4), the errors, once again,

[1]The absolute value should scale as $\frac{1}{r^2}$ with the distance from the sun

were not appreciably different. The fixed parameters used (See table 5.1) worked quite well when changing between data sets. Notice in Figure 5.4 how the error goes up as time progresses, indicating that we are leaving the photo-dominated region and other currents start to become important.

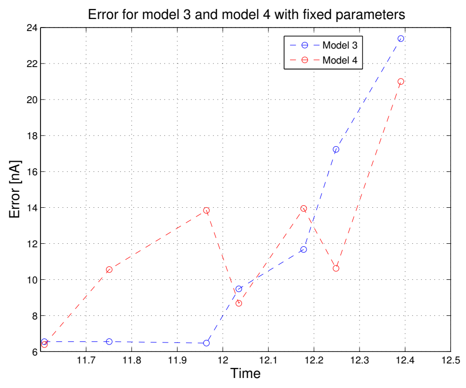


Figure 5.4: The total least square error divided by the number of data points for models 3 and 4 when using fixed parameters, probe 1. Seven probe bias sweeps are used.

Model	$I^1_{Ph,0}$ [nA]	$I^2_{Ph,0}$ [nA]	$T_{Ph,1}$ [eV]	$T_{Ph,2}$ [eV]	Mean error [nA]
Model 3	56	4.5	2.6	12	11.6
Model 4	55	6	1.2	7.1	12.2

Table 5.1: The fixed physical parameters used when comparing model 3 and 4,probe 1. These are the values used in Figure 5.4

Both models worked well, but due to the fact that the parameter values determined for model 4 are closer to those found in the literature[13], the decision fell upon model 4 (eq. 5.4). Henceforth, it will be used with the values given in table 5.1. Figure 5.5 shows a fit of a sweep in a photo-current dominated plasma using both models and the values in Table 5.1.

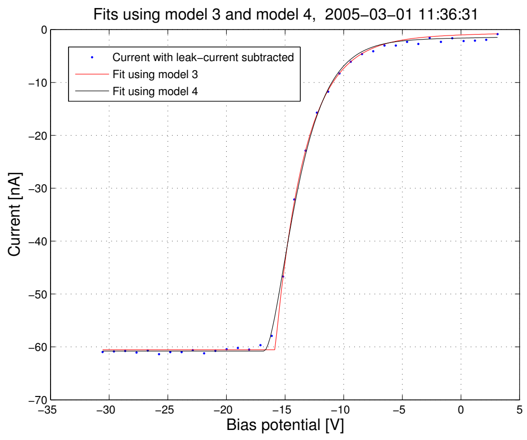


Figure 5.5: A fit of a sweep from when the probe was in sunlight, using model 4 and model 3, probe 1. Notice the smooth character of model 4 at the bend. The values used are given in Table 5.1.

It is interesting to note that the simpler models 1 and 2 were discarded early on. The photoelectron energy distribution thus appears to be more complex than a single Boltzmann distribution.

5.5 The LAP dance

The LAP dance, performed on October 10, 2004, is a time interval during which the Rosetta probes were alternatingly moving in and out of sunlight. This "dance" of the s/c through different probe illumination conditions provides an interval over which the probes response to total shadow as well as total sunlight can be measured and analysed. The LAP dance was performed when Rosetta was in the solar wind, i.e. in a very tenuous plasma (see table 6.1) and once again the measured current on the ion side of the sweep should basically be photoelectrons leaving the probe. This also means that when the probes are in shadow, no current should be measured. Figure 5.6 is a plot of the photo-electron current leaving the probes (actually it is a plot of the saturated ion side current, which in this case should be mainly photoelectrons). I_{ph0} was determined by first subtracting the leak-current (see Section 5.2) from the data and then fitting the ion saturated side linearly. The first 40% of the ion side was used for this fitting. This lines crossing of the y-axis should, to a good approximation, be I_{ph0}, and is what is used in Figure 5.6.

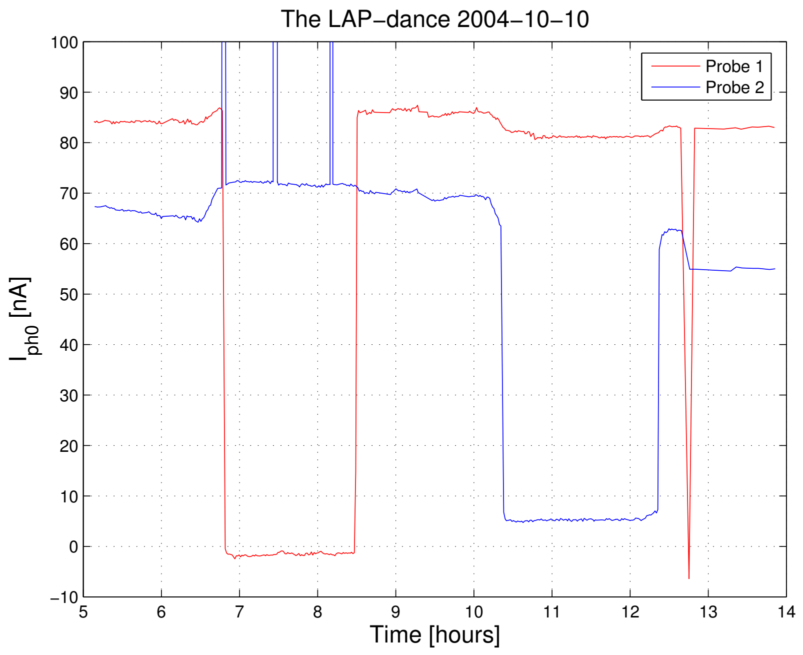


Figure 5.6: The current to the probes calculated from a linear fit of the ion side of the sweep. The three peaks in P2 data are not real.

Some qualitative conclusions can be drawn immediately. It is clear that the probes differ in their total photo-emission, probe 1 emitting about 85 nA while probe 2 only emits 60-70 nA. This difference could be due to some contamination on probe 2, perhaps from thruster exhaust. Another effect, not investigated here, is the fact that the probes will be illuminated differently. Depending on the orientation of the sun probe 2 may have had a greater part of its surface blocked by the shadow of its boom. Another possible explanation is that the probe surfaces may have slightly different photoemission properties.

Another interesting feature can be seen when either probe moves into darkness (between 6:50 and 8:30 for probe 1 and between 10:20 and 12:10 for probe 2). Instead of giving a zero value both still register a current. For probe 1 this current is slightly negative, meaning that the probe actually collects electrons (note that the sign convention in these plots is such that photoemission is counted positive), while probe 2 has a larger positive current, which could stem from some residual photoemission, possibly because of reflected light from the solar panels or some other surface on the spacecraft. It cannot be a collection of ions, since during the LAP dance, being in shadow from the spacecraft is equivalent to being in the wake of the spacecraft, as the solar wind propagates radially out from the sun.

The wake should be quite free of ions as explained earlier (see section 4.3.2).

In order to find out whether this remnant current was due to calibration of the instrument or some physical event a closer look at this current was performed. The linear fit of the ion side, defined as discussed above, was evaluated to give the currents at $V_\mathrm{b} = 0$ and $V_\mathrm{p} = 0$ respectively.

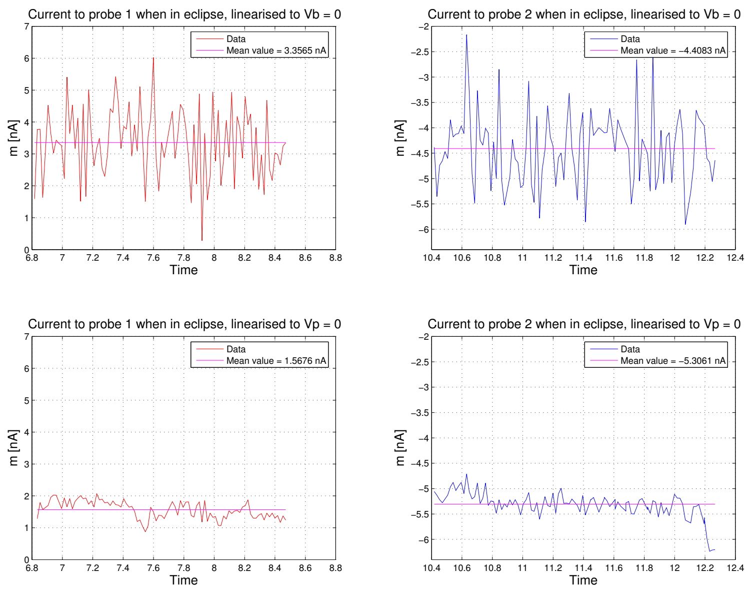


Figure 5.7: The crossing of the linear fit with the y-axis as a function of time.

The top two plots in Figure 5.7 show the evaluation for probe 1 (red) and probe 2 (blue) at $\mathrm{V}_\mathrm{p} = 0$ and the lower two are the evaluation at $\mathrm{V}_\mathrm{b} = 0$. The fact that the lower plots show *less* variation than the upper plots suggests that the discrepancy has a physical origin. If the current originated from a calibration error one would expect it to be better organised by the bias potential V_b, than by the actual probe-to-plasma potential V_p. Since this is not the case the conclusion is that the problem is a physical one, in the sense that the observed currents are due to particles carried to the probes, not due to any imperfections in the instrument electronics. The fact that the value of the current has opposite sign for the probes (1.5 nA for probe 1 and -5.3 nA for probe 2) further complicates the picture. This seem to point at different physical processes working in the two cases. For probe 2 one explanation could be reflection of UV light from the solar panels, thus exposing it to radiation which would mean a small photoemission even though it is supposed to be in shadow. To further explore this option the attitude of Rosetta was needed at the data points. Using routines built by Magnus Billvik[5] and applying them (somewhat modified) to the code created for analysing the data sweeps this information could be extracted. The result can be seen in Figure 5.5.

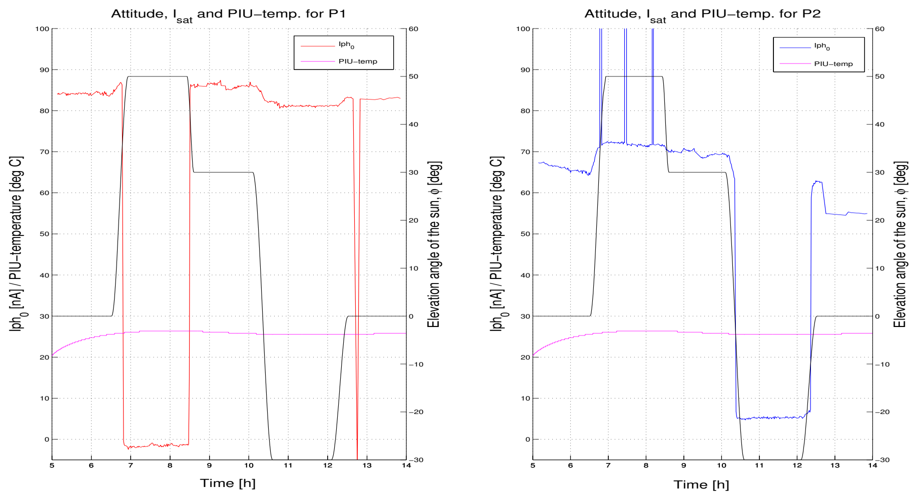


Figure 5.8: The attitude of Rosetta, ion saturation current of the probes and temperature of the PIU during the LAP-dance. The attitude is given as the elevation angle from the xy-plane of Rosetta of the radius vector pointing to the sun. The current closely follows the attitude change, notice the sudden rise in photo-emission when Rosetta starts to turn. The temperature of the PIU (teal line) does not follow the fast changes in the attitude.

Here the elevation angle from the xy-plane of Rosetta of the radius vector pointing to the sun is given as a function of time. There is no need to include a second angle (i.e. the azimuth) for the attitude because Rosetta is always pointing its solar panels at right angles to the sun in order to maximise the power output (see Figure 5.5). In addition to the attitude the temperature of the *Plasma Interface Unit*[14] (PIU) is also given. The PIU temperature sensor is mounted in the same elctronics box as the LAP electronics, and can be assumed to be a reasonable proxy for the LAP temperature. The reason to include the temperature in the plot is to look for any effects of varying instrument offsets due to varying temperature in the instrument box. Even though there is a small change in the temperature, possibly due to attitude changes, but more probably becuase of variation in internal heat generation, it is not on the same time scale as the fast changes in photo-emission and attitude and can thus be ruled out as a source for the observed variation of photoemission current. The sudden rise in the photo-emission around 6:30-6:40 for both probes can now be linked to a simultaneous change in attitude. What gives rise to this change in photo-emission is not clear. It could be sunlight reflection from the surface of the s/c, since the light will come in at an angle to the spacecraft, or perhaps some inhomogeneity on the probe surface, but this is just speculation and unfortunately this is where we need to stop. It is beyond the scope of this report to investigate these details further.

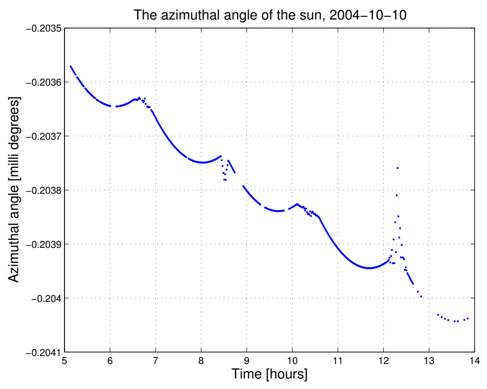


Figure 5.9: The azimuthal angle from the xz-plane to the vector pointing towards the sun during the LAP-dance. It is very close to zero and changes very little throughout the time interval, thus exposing as large an area of the solar panels to sunlight as possible.

5.6 Conclusion

Analysing the photoemission from the probe 2 it was found that the model that fit the data best was of the form

$$I_{ph} = \begin{cases} -I^1_{ph,0}\left(1+\frac{V_p}{T_{ph,1}}\right)e^{-\frac{V_p}{T_{ph,1}}} - I^2_{ph,0}\left(1+\frac{V_p}{T_{ph,2}}\right)e^{-\frac{V_p}{T_{ph,2}}}, & V_p > 0 \\ -I^1_{ph,0} - I_{ph,0}, & V_p < 0 \end{cases}$$

Parameters to the model were also determined and fixed to be as in Table 5.2.

Parameter	Value
$I^1_{Ph,0}$	55 nA
$I^2_{Ph,0}$	6 nA
$T_{Ph,1}$	1.2 eV
$T_{Ph,2}$	7.1 eV

Table 5.2: The physical parameters used for model 4

This model is what we are going to use in our further work, fixing the parameters but leaving the absolute value of the photoemission variable, by applying a dimensionless factor. The same form of the photoemission current, with the same values for $T_{ph,1}$, $T_{ph,2}$ and $\frac{I_{ph,1}}{I_{ph,2}}$ is assumed to hold also for probe 1.

We found a small (a couple of nA) unexplained current. This provided the basis for a hypothesis of a leak current from the probe to the spacecraft. This current, albeit small, may influence the analysis when in very tenuous, sunlit plasmas, where there should be small contribution from the ion current. In these cases it can be a good idea to subtract the slope of the ion side throughout the data. When in denser plasmas this effect is utterly negligible (where ion currents of μA are common) and can be disregarded. Another possibility is the influence of the spacecraft body on the plasma being measured by the probes, effectively lowering the spacecraft potential measured.

The analysis of the LAP dance led to some information regarding the characterstics of the probes, presented below

- **Absolute photoemission:** A difference in the emission of photoelectrons between the probes were found. Probe 1 has a photoemission of about 85 nA and probe 2 a photoemission of about 72 nA. These values were determined at 1.09 A.U. for conditions on 2004-10-10. The origin of the difference in photoemission magnitude is not known.
- **Remnant current:** Both probes registered a remnant current when in wake and shadow. The remnant current to probe 1 was positive (between 1.5 and 3.5 nA) and the remnant current to probe 2 was negative (between -4 and -5.5 nA). The origin of this current, while not concluded, seem to be physical and does not appear to have any connection to the temperature of the LAP electronics. The difference in sign point to different physical processes working in the two cases. One possibility for probe two could be reflection of UV light from the solar panels or s/c body.

Further, more detailed, work need to be carried out to understand the details of the above mentioned peculiarities. These effects found are however quite small and will not present too much of a problem for the continued work in this thesis and will thus be neglected henceforth.

Chapter 6

Automatic fit routine

Here the routines which are used for the automatic fitting of the data are presented and the parameterisation is explained. The algorithm used for fitting the data automatically is walked through step-by-step and some of the more non-transparent function files will be explained a little more in depth for anyone faced with task of working with these routines.

The first thing to note is that the automatic fitting routine is embedded into the graphical users interface "Grafical" and as such is not very convenient to alter. The basic functions, however, once given the proper indata, do not need the GUI and can thus be explained separately. In fact, the entire automatic fitting process is independent of the GUI in this aspect and in this chapter we will discuss it a bit more in-depth. Before we do that, however, the parameterisation of the models need some discussion. In what follows, functions, vectors (i.e. row or column matrices, not physical vector quantities) and matrices will be given in **bold** text.

6.1 Parameterisation

In order to determine a good fit for the data we need to determine the characteristics of the plasma in terms of its physical parameters. The problem is that it is very inconvenient to use the real physical parameters directly in the fit. Instead, a parameterisation based on the mathematical form of the model is implemented (e.g. straight line, exponential etc.). This approach makes it easier to determine some parameters directly from for example a simple linear fitting. These can then be used as initial guesses in a non-linear fitting for the parameters. The total current is given by

$$I = \begin{cases} -I_{e,0}\left(1+\frac{V_p}{T_e}\right) - I_{i,0}e^{-\frac{V_p}{T_i}} - I_{ph,01}\left(1+\frac{V_p}{T_{ph,1}}\right)e^{-\frac{V_p}{T_{ph,1}}} - I_{ph,02}\left(1+\frac{V_p}{T_{ph,2}}\right)e^{-\frac{V_p}{T_{ph,2}}}, & V_p > 0 \\ -I_{e,0}e^{\frac{V_p}{T_e}} - Ii,0\left(1-\frac{V_p}{T_i}\right) - I_{ph,01} - I_{ph,02}, & V_p < 0 \end{cases} \tag{6.1}$$

This lends itself to a simple parameterisation of the form

$$I = \begin{cases} ae^{\frac{b}{a}V_p} + c\left(1+dV_p\right) + f\left(I_{ph,01}\left(1+\frac{V_P}{T_{ph,1}}\right)e^{-\frac{V_P}{T_{ph,1}}} + I_{ph,02}\left(1+\frac{V_P}{T_{ph,2}}\right)e^{-\frac{V_P}{T_{ph,2}}}\right), & V_p > 0 \\ a + bV_p + ce^{dV_p} + f\left(I_{ph,01} + I_{ph,02}\right), & V_p < 0 \end{cases} \tag{6.2}$$

where we have chosen to keep the photoelectron parameters as they are, only varying the total photoelectron current by the use of the parameter **f**.

The values of the parameters $I_{\text{ph},01}$, $I_{\text{ph},02}$, $T_{\text{ph},1}$ and $T_{\text{ph},2}$ will be kept fixed, using the values determined in Chapter 5. The conversion between mathematical parameters and physical parameters are defined as

$$\mathbf{a} = -I_{i,0} \tag{6.3}$$

$$\mathbf{b} = \frac{I_{i,0}}{T_i} \tag{6.4}$$

$$\mathbf{c} = I_{e,0} \tag{6.5}$$

$$\mathbf{d} = \frac{1}{T_e} \tag{6.6}$$

$$\mathbf{g} = \frac{1}{T_{ph01}} \tag{6.7}$$

$$\mathbf{h} = \frac{1}{T_{ph02}} \tag{6.8}$$

Now **b** can be determined directly from the slope on the ion side. The reason why we did not parameterise **d** in the same way is because of the fact that the slope on the electron side is much less trustworthy in that it may contain several other current contributions (such as photoelectrons from the spacecraft which we have not included in the above parameterisation) besides plasma electrons.

6.2 Algorithm

In order to understand and interpret the fits one need to know how exactly, these are made. This section will look at all the algorithms used when fitting the data, step-by-step.

6.2.1 Data handling

The data from the Rosetta LAP instrument can be obtained through the client-server system ISDAT. Originally developed for the Cluster satellites, ISDAT reads the data from the various missions (such as Freja, Cluster, Viking, Rosetta etc.) and presents it in a uniform format. There are several different clients with which one can obtain the data from ISDAT. Here we use the ISDAT interface developed for Matlab, the function **isgetDataLite**[15].

The data obtained through ISDAT cannot be used directly by the fitting routines, but must first be adjusted somewhat. There are mainly three things that need to be taken care of:

- **Sorting:** ISDAT presents the data in chronological order. As the sweeps can be made in any of the directions down, up, up-down or down-up, the data need to be sorted in ascending order, from low bias potentials to high. This was done to facilitate the coding and it is a convenient and transparent way to have the data stored.
- **Mean of same potential values:** For every bias potential, four measurements of the current will be gathered. This is because of when the bias is changed it will take some time for the current to respond. For the sweeps we here used, there is little sign of any transient, so by performing several measurements and then taking a mean value one can be quite certain that the right current is obtained.
- **Cutting saturated values:** The LAP indstrument has two gain ranges, ± 10 μA or ± 200 μA. All measuremnets treated in this report were made with the ± 10 μA range, so all currents above this value will ne saturated. This will of course interfere with the fitting procedure so all data above that threshold must be cut in order for the fitting to function properly.

For this reason a function named **adjust_data.m** was created. Given the bias potential, current and time it performs the above mentioned adjustments to the data, returning the three inputs ready to be used by the fitting routines.

6.2.2 The spacecraft potential

As was mentioned in Section 3.3, obtaining the spacecraft potential, $\mathbf{V}_{\text{sc}}$, is crucial in order to perform the fitting of the data. It is important that a good initial value of $\mathbf{V}_{\text{sc}}$ is determined before the non-linear fitting techniques are used, since all the models are dependent on this potential. If an acceptable value has been determined it can be refined using **lsqcurvefit**, see Section 6.3.4.

A function, named **find_scpot.m**, has been developed in order to get a first determination of the spacecraft potential. It looks for the maximum in the second derivative of the data. In theory this method should work fine. The problem, however, is that often the data may be somewhat noisy. Taking the derivative numerically, unfortunately has the effect that it blows up noise, and since we are forced to take two derivatives the effect is doubled. Because of this fact, and the fact that all data points are looked through in an attempt to get hold of the maximum in the second derivative, the routine may find local maxima that are actually larger than the global maximum of the real $\mathbf{V}_{\mathrm{sc}}$. This will cause problems for the entire fitting. This effect is most important in very tenuous plasmas for a probe in eclipse, where the bend close to the spacecraft potential is weak. A couple of procedures are used to remedy this problem, the most important one being the filtering of the data using a Savitzky-Golay filter before differentiation. Another important procedure is the first non-linear fitting of the spacecraft potential performed in **fit_data2.m**, see Section 6.3.4 for more details on how this fit is performed. In the function **preliminaries.m** some more steps are taken to try and determine the spacecraft potential *before* the actual non-linear fitting is performed, see Section 6.3.3.

6.2.3 Walkthrough

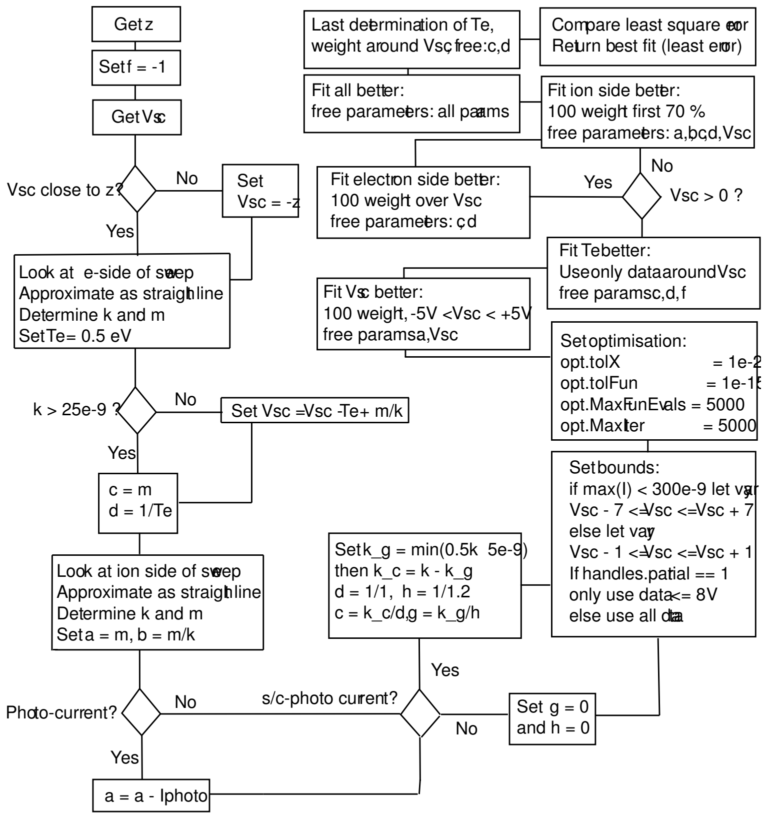


Figure 6.1: A flow chart of the algorithms used in the automatic fitting routine

The first function called is **gothrough.m** in which the coordinates and velocity of Rosetta is determined. This is done by using (somewhat altered) routines developed by Magnus Billvik[5]. The program also sets the coordinate system and units (these are chosen by the user, see Section A.1.3) correctly and saves all in a structure called PARAMETERS.

Now the function **fitting2.m** is called (from **gothrough.m**) and this is where the actual fitting starts (see Section 6.3.2 for more details on this).

- The photoemission current, temperature and functional form, is hardcoded in the function, using the values obtained when investigating the photocurrent exclusively, see chapter 5. Only the absolute magnitude of the photoemission current is fitted.

- The initial values of the other parameters are determined by calling a function created specifically for this purpose; **preliminaries.m**, see Section 6.3.3. The parameters determined in **preliminaries.m** are:

- The zero-crossing voltage of the current, **z**.
- The parameter used to change the magnitude of the photocurrent, **f**, is at this stage hardcoded to -1.
- A preliminary value of $\mathbf{V}_{\mathrm{sc}}$.
- The parameters **c** and **d** determined from the electron side of the sweep (being mathematical parameters for the plasma electrons). The slope of the electron side, **k** is also given.
- The ion side parameters, **a** and **b** are obtained.

- Depending on whether the user has chosen to set the spacecraft potential manually $\mathbf{V}_{\mathrm{sc}}$ is set to this manual value or else to the value determined in **preliminaries.m**.

- Depending on whether it should be a photocurrent or not (the user can have chosen not to have it or, alternatively, the automatic fitting may have chosen a combination that excludes photocurrents) **f** is either set to zero or the value determined in **preliminaries.m**. **a** is corrected accordingly.

- If there are two electron populations, **c** and **d** are altered and **g** and **h** are introduced, otherwise **g** and **h** are set to zero.

- Now, depending on whether the user has chosen to do one automatic fit using the model he/she has chosen or if he/she has chosen to try all different fits, this is done. Below, the procedure that the algorithm utilises when trying all models is described. Figure 6.2 is a schematic representation of the ordering of the models. The function performing the fit is called **fit_data2.m** and it is explained in-depth in Section 6.3.4.

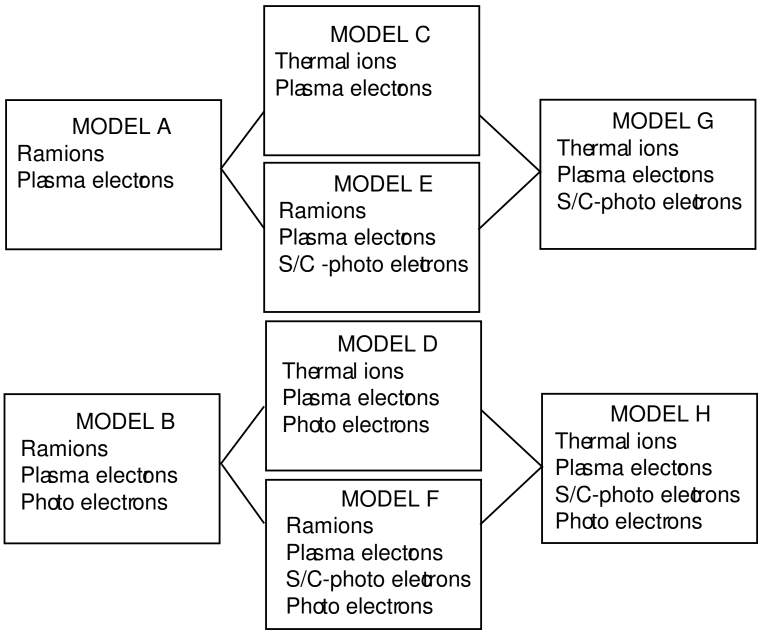


Figure 6.2: A schematic representation of the model order taken by the routine

- The first model that is fit, model A, is the one containing **ram ions** and **plasma electrons**.
- The second one, model B, is **ram ions**, **plasma-** and **probe photo-electrons**.
- The values determined for the parameters from models A and B are now used as initial values for models C-F.
- Model C, **thermal ions** and **plasma electrons** is fitted, using values from model A.
- Model D, **thermal ions**, **plasma electrons** and **probe photoelectrons**, using values from model B.
- Model E, **ram ions**, **plasma electrons** and **sc-photoelectrons**, using values from model A.
- Model F, **ram ions**, **plasma-**, **sc-** and **probe photoelectrons**, using values from model B.
- The values determined for the parameters from models C-F are now used for models G and H.
- Model G, **thermal ions**, **plasma electrons** and **sc-photoelectrons**, is fitted using values from both model C and model E.

  - Model H, **thermal ions**, **plasma-**, **sc-photo electrons** and **probe photoelectrons**, is fitted using values from both model D and model F.

- All these different fits, models A-H, are now compared using a function called **choose_fit.m**. The comparison is performed in one of two ways depending on what the input into the function is. If the function has an input with errors calculated for the models already, so called resnorms that are obtained from the MATLAB built-in function **lsqcurvefit**, obtained when fitting the models, **choose_fit.m** merely picks out the one with the least error and brands this fit as the best one. The resnorms are actually a sum of the square of the function value in every data-point minus the actual data value in that point. If there are no resnorms output from the function, the least square value is calculated by another function and the fit with the smallest least square is taken as the best fit. It is now returned and saved.

6.3 Functions

All in all, there are 18 functions written for the purpose of analysing the data automatically. The most important ones are described below.

Table 6.1: The routines developed for the automatic analysis of the sweep data

Filename	Description
extraction.m	the main program, governing the other functions
fitting2.m	governs the fitting process
preliminaries.m	extracts some preliminary (initial) values for the parameters
zero_cross.m	determines where the current crosses the y-axis (i.e. crosses zero)
moving_average.m	smoothes the data
find_scpot.m	determines a first spacecraft potential from the second derivative of the data
d2.m	using numerical formulas for the derivative, returns the second and third derivative of the data
fit_single_e.m	computes initial guesses of parameters coupled to the electrons
determine_ecl	determines if the probe is in eclipse or not
fit_data2.m	the function performing the fitting, using **lsqcurvefit**.
model.m	is called with parameters to calculate the current for the given value of the parameters
compute_c.m	is used to calculate the parameter c from the floating potential
weighting.m	weights the data around a user specified point by a user specified amount
choose_fit.m	given a set of different fits, chooses which one that fits the data best
errorest.m	calculates a least square error for the fit to the data
compare_fits.m	does essentially the same thing as choose_fit, but more specific for the extraction routine
par2phys2.m	converts the mathematic parameters used in the fit to physical parameters
getIparts.m	returns the different constituent currents of the fit

6.3.1 extraction.m

Form: [ppar,fit,currents,epop,fits,pars,parameters] = extraction(Vb,I,handles,phi,probe)

Input: $\mathbf{V_b}$ and **I** are the bias potential and the current of the sweep respectively. The bias potential need to be adjusted

handles is a structure containing information regarding spacecraft speed and what kind of currents should be used in the fitting. In grafical (see Appendix A) **handles** contain lots of information specific for the GUI that is not needed for the analysis, but the information that *must* be contained for the fit is:

handles.vsc	spacecraft *speed* in m/s, for example 10000.
handles.e_plasma	use plasma electrons? 1 = yes, 0 = no
handles.e_photo	use photo electrons emitted from the probe?
handles.e_sc	use photo electrons coming from the s/c?
handles.i_ram	use ram or thermal ions? 1 = ram, 0 = thermal
handles.fit_partial	Fit entire data set or just lower part? 1 = partial, 0 = all

phi is the elevation angle of the sun in radians.

probe designates what probe the data comes from, 1 or 2.

Output: **ppar** is the physical parameters of the plasma given as a vector: [V_{sc} f z I_{ph01} I_{ph02} T_{ph1} T_{ph2} n_{i} T_{i} m_{i} n_{e} T_{e} n_{e2}], where f is a factor for the photocurrent (though already multiplied in I_{ph01} I_{ph02}, z is the zero-crossing of the current and m_i is the effective mass of the ions. Unfortunately this version of the program does not support effective mass, and as such m_{i} is a placeholder variable.

fit is a matrix containing the potential and current values of the fit.

currents is a cell matrix containing the constituent currents of the fit in order I_{ph} I_{i} I_{e} I_{e2}

epop specifies if one or two electron populations were assumed in the fitting process, 1 or 2.

fits and **pars** are the fits and *mathematical* parameters of all the different models respectively. These are only used when the **try all fits** option is selected. All models are returned so that the user can go through their results, regardless of the choice of best model by the algorithm. In this case **parameters** is the mathematical parameters of the best fit.

6.3.2 fitting2.m

Form: [parameter,fit] = fitting2(Vb,I,handles,phi,probe)

Input: The inputs are exactly the same as for **extraction.m**. See Section 6.3.1.

Output: **parameters** is the mathematical parameters obtained in the fitting. They are given as a row vector in the form: [$\mathrm{I_{ph01}}$ $\mathrm{I_{ph02}}$ T_{ph1} $\mathrm{T_{ph2}}$ a b c d g h V_{sc} f z err] where err is the error of the fit in amperes.

fit is a matrix containing the potential and current values of the fit.

In **fitting2.m** the fitting procedure is governed. First it calls **preliminaries.m** (see Section 6.3.3) to get initial values for the parameters. These initial values are manipulated somewhat depending on the selections performed by the user. If the user has chosen to set the spacecraft potential manually, it will be used as initial value. Furthermore, depending on whether photoelectrons are used or not, **f** will be set to the value obtained from **preliminaries.m** or to zero. As a result, **a** is corrected to

$$\mathbf{a} = \mathbf{a} - \mathbf{f} \cdot I_{ph} \tag{6.9}$$

where I_{ph} is the total magnitude of the photocurrent, since **a** is, in a sense, the absolute value of the ion-current, and the ion side of the sweep should be a superposition of the ion-current and the photocurrent.

If the user has selected two electron populations, the initial value for both will be set as follows. The initial guess of the temperature for the spacecraft photoelectrons will be 1.2 eV (this is equal to the value obtained for the photoelectrons escaping the probe, see Section 5.6) and for the plasma electrons to 0.5 eV. Now the slope on the electron side will determine the value for $\mathrm{I_{e01}}$ and $\mathrm{I_{e02}}$ through the formulas

$$I_{e01} = -\mathbf{k_1} \cdot T_{e1} \tag{6.10}$$

$$I_{e02} = \mathbf{k_2} \cdot T_{e2} \tag{6.11}$$

where k_1 and k_2 are determined from **k**, the slope of the electron side, returned from **preliminaries.m**. k_2 is set as the minimum of 0.5·**k** and $5{\cdot}10^{-9}$ and then k_1 will be the remaining slope, i.e.

$$\mathbf{k_1} = \mathbf{k} - \mathbf{k_2} \tag{6.12}$$

What this means is basically that the second electron population, as a starting guess, can maximally be half of the electrons collected by the probe (the fitting procedure will then surely change this). Of course, in the program, we do not work with the physical parameters, but rather with the mathematical ones, and accordingly the algorithm will set:

$$\mathbf{h} = \frac{1}{1.2} \tag{6.13}$$

$$\mathbf{d} = \frac{1}{1} = 1 \tag{6.14}$$

$$\mathbf{c} = \mathbf{k_1}\frac{1}{h} \tag{6.15}$$

$$\mathbf{g} = \mathbf{k_2}\frac{1}{d} \tag{6.16}$$

When the preliminaries have been performed, **fitting2.m** continues with the fitting. If the user has selected the **try all** option, the program will try all different combinations of fits on the data. The order in which the different combinations are tested is important. The output from some fits are used as the input for others, eg. the output from the fit using one electron population is used as the input for the fit using two electron populations. See section 6.2 for more on the order of the fits. The different combinations are now compared and the best one is chosen as the fit to use and its parameters are returned by the function.

6.3.3 preliminaries.m

Form: [dat_out] = preliminaries(Vb,I)

Input: Only the bias potential, $\mathbf{V}_\mathrm{b}$, and the current, **I**, is needed for the input.

Output: **dat_out** is a row vector containing the initial values of some of the mathematical parameters: [a b c d Vsc f z k] where k is the slope of the electron side of the sweep.

preliminaries.m is a function that calculates the initial values of some of the parameters that are needed. It uses several functions to do this, but they will not be explained in depth. The workings of these functions will be explained here.

The first thing **preliminaries.m** does is to determine the zero crossing in the current, **z**. This is done through the function **zero_cross.m**. The algorithm is taken from Reine Gill[16] and simply put looks for the first value in the current above zero and the last value in the current below zero. Then it draws a line between these and finds the zero crossing on this line. If the data is noisy, the first current value above zero does not need to be a real *physical* value above, but rather due to a noisy data set, some points may randomly rise above zero. Becuase of this a procedure in which the point with the greatest possibility of being the *first* one above zero in current and the point with the greatest possibility of being the *last* one below zero in current are determined. The line between these two are then taken and the zero-crossing determined on this line.

The parameter that affects the photocurrent magnitude, **f** is hardcoded to -1.

Next a starting value on the spacecraft potential is found by the use of the second derivative. Refer to Section 6.2.2 for more details regarding the determination of the spacecraft potential. Depending on the value of $\mathbf{V}_\mathrm{sc}$ and **z**, $\mathbf{V}_\mathrm{sc}$ may be set to -**z**. This is becuase the spacecraft potential cannot be too far to the right of the zero crossing, i.e. not at too high $\mathbf{V}_\mathrm{b}$ values as compared to **z**.

Next a determination of the initial values to **c** and **d** is performed with the help of a function named **fit_single_e**. The name refers to the fact that this only determines parameters when there is a single electron population. The algorithm is a simple one, looking at the last 70% of the data above $\mathbf{V}_\mathrm{p} = 0$, to which a linear fit to the data is performed:

$$y = kx + m$$

and by using MATLAB's function **polyfit** **k** and **m** are easily obtained. The initial value of the temperature, T_{e0} is set to 0.5 eV. It was found too hard to set a consistent initial value of the temperature based on the data given. The reason being that the data-points were too sparse. The temperature can be obtained by looking at the data just below the bend (close to the spacecraft potential). The problem is that there is only a small interval in which to look, perhaps in the order of 0.2 V. With the data-point spacing being 0.125 V only one or two points may be good and this is far too small a data to draw any good conclusions from. Depending on whether the slope of the line approximated is greater than $25{\cdot}10^{-9}$ or not the determined spacecraft potential is adjusted accordingly

$$V_{\rm sc} = V_{\rm sc} - T_{e0} + \frac{m}{k} \tag{6.17}$$

Now, $\mathrm{I_{e0}}$ = m, and $\mathrm{T_{e0}}$ is returned. The program sets

$$\mathbf{c} = I_{\rm e0}$$

and

$$\mathbf{d} = \frac{1}{\mathrm{T_{e0}}}.$$

Next a simple determination of the ion side is performed in the same way (from a linear fit of the low part of the line this time), and initial values of **a** and **b** can be obtained.

All these values; **a**, **b**, **c**, **d**, **k** and $\mathbf{V_{sc}}$ are then returned in a vector to **fitting2.m**.

6.3.4 fit_data2.m

The function responsible for the actual fitting is **fit_data2.m**. Here we will go through this function, step by step, in order to try and clarify how it works.

Form [par_out,fits_out] = fit_data2(Vb,I,par,handles)

Input: $\mathbf{V_b}$ and **I** are the bias potential and current of the sweep respectively.

par is a vector containing initial guesses for the mathematical parameters that are used to fit the data, see section 6.1 for their exact definition. They must be given in the form [I_{ph01} I_{ph02} T_{ph1} T_{ph2} a b c d g h V_{sc} f z]. By using **preliminaries.m** initial guesses of a,b,c,d,V_{sc},f and z can be obtained.

handles is the same structure that need to be given when calling **extraction.m**. Refer to section 6.3.1 for how it works.

Output: **par_out** is a vector containing the *fitted* mathematical parameters, it is in the same form as par, except for the fact that there is a extra 14th parameter, the error of the fit included.

fits_out is a matrix containing the potential and current values of the fit.

The first thing done is setting bounds on the parameter values that the program alters in order to fit the models to the data. These bounds are common sense ones, for example **f** can never be greater than zero or because of positive density **a** must always remain negative. One thing worth mentioning is that V_{sc} takes on two different bounds depending on the data. If the maximum current is less than $300 \cdot 10^{-9}$ A, i.e. a tenuous plasma, the first determination of the spacecraft potential may be quite wrong, so V_{sc} is allowed to change between +7 and -7 V of its original value, while if the current is greater than $300 \cdot 10^{-9}$ A it is only allowed to vary by 1 V either way.

Depending on what the user has selected, either the entire data (if handles.fit_partial = 0) or just the part of the data below 8 V (if handles.fit_partial = 1) will be analysed.

Next the optimisation is set. With opt.TolX and opt.TolFun one can set what tolerance should be used, the smaller the value chosen, the better the fit. Unfortunately, lowering the tolerance will lead to a slower fitting process. The variables opt.MaxFunEvals and opt.MaxIter sets the number of function evaluations the program should perform.

After these preliminaries the actual fitting starts. **fit_data2.m** uses a built-in function called **lsqcurvefit**. **lsqcurvefit** uses a non-linear fitting technique, which require the initial guesses as well as the bounds on the values to be quite good (since there exist several local minimums in the function and it can find one of these). It is important to understand how **lsqcurvefit** works in order to understand the workings of the fitting procedure.

The vector **ph_par** and **params_in** are given to **lsqcurvefit** which, in its turn, will pass these vectors on to the model used, **model.m** in this case (see section 6.3.5 for more about how **model.m** works). The way **lsqcurvefit** works, is that you first give a function, through which **lsqcurvefit** will try different values of the parameters given in **params_in** (the first try is those values given and then **lsqcurvefit** will change the values to fit better). In each step it compares the fit obtained (**model.m** returns a current vector) to the real data, which is given as the fourth input in **lsqcurvefit**. **lowbound**, **highbound** and **opt** are the lower and upper bounds for the parameters and the optimization ranges structure set earlier. Everything that comes after **opt** in the calling of **lsqcurvefit** will without discrimination be passed on to the function given in the first input. In our case this mean that **ph_par**, **ref**, **vargin** and **handles** will be passed on to the function **model.m**. Refer to Section 6.3.5 for a breakdown of **params_in**, **vargin** and **ref**.

Before **lsqcurvefit.m** is called, however, the program determines what kind of model the user wants to try and fit. Depending on the model, different parameters will be varied and held fixed. The ones being varied are given in the vector **params_in** and the ones kept fixed in **vargin**.

The fit is now performed several times, holding different parameters fixed and weighting different intervals in each fit. The weighting is performed using the function **weighting.m**. To weight the data around a point one need to give the data (V and I), the point to weight, how large symmetrical interval around this point one wants (in volts) and finally how much weight (10 means to create 10 copies of each point in the interval, 100 to create 100 copies etc.).

Depending on the data, 5 or 6 different steps are performed.

- First the program tries to determine the spacecraft potential better. This is done by weighting the data around the initial guess of $\mathbf{V}_{sc}$ (± 5 V) by a factor of 100, and then varying the parameters **a**, **f** (if photocurrent is selected), and $\mathbf{V}_{\mathrm{sc}}$.

- The second step tries to find a better value for the electron temperature, T_e. This is done by *only* looking at the data points very close to $\mathbf{V}_{\mathrm{sc}}$ (using the three datapoints lying below -$\mathbf{V}_{\mathrm{sc}}$+1 V), letting **c** and **d** change.

- The third step is dependant on the sign of V_{sc} obtained. If it is positive the program tries to fit the electron side better, focusing on the parameters **c** and **d** (and possibly **g** and **h** though most often not) and weighting the electron side. The reason for only doing this when having a positive spacecraft potential is because of the fact that when two electron populations are present the spacecraft potential is almost always negative. Trying to fit the electron side better in this case and get sensible results for the two populations is almost impossible. The reason is that there are in principle two linear functions on the electron side and the program will have no idea how to separate them, so the values they are given will be random (the total fit, however, should be quite good). When V_{sc} is negative the program will immediately go to the fourth fitting stage.

- The fourth stage is focused on the ion side of the sweep, trying to get a good fit on the lower part of the data. Here **a**, **b**, **d**, $\mathbf{V}_{\mathrm{sc}}$, **f** (if applicable) and **z** are all free, but **b**, **z** and $\mathbf{V}_{\mathrm{sc}}$ will probably not change much as they should be quite good already, and are therefore held constant.

- Now the parameters of the best fit is taken as values going in to a fifth fit, where all parameters except **b** are varied. The data is weighted around V_{sc} quite heavily since this is an important region.

- Finally all the different fits are compared and the best one is chosen. This comparison is performed by the function **choose_fit.m**. The best fit will go through a final procedure to try and determine T_{e} better, since this is a very important variable (it will also influence the density). This means that **c** and **d** are allowed to change once again.

The last thing done is determining an error in the least square sense and then returning all parameters with errors from the function.

6.3.5 model.m

Form: I = model(params,Vb,ph_par,ref,vargin,handles)

Input: **params** are the mathematical parameters that can be varied by the function **lsqcurvefit.m** (see Section 6.3.4), given in a vector. Which parameters to place and how to place them in the vector depends on the choice of the variable **ref**. For example, if **ref** = 1, then the parameters in **param** should be [a Vsc z] and **vargin** should be [b d]. In this way **param** and **vargin** are closely tied. Refer to the help-text of **model.m** for the different choices of fixed and free parameters. The reason for this somewhat complicated way of using different references, is to be able to use the function with different parameters varied, in this way it is specially tailored for **lsqcurvefit**. On the other hand, when just using it to get a value on the current (for example when using **feval.m**, it does not matter which reference one chooses, as long as **params** and **vargin** are correctly set.

ph_par is always the photoelectron parameters and should be given as [I_{ph01} I_{ph02} T_{ph1} T_{ph2}].

handles is the exact same structure as in **extraction.m** and **fit_data2.m**, see section 6.3.1.

$\mathbf{V_b}$ is the bias potential, and the current will be calculated over this interval.

Output: **I**, the current calculated for the parameters given.

By calling **model.m** with a certain set of parameters, it will return a current calculated over the entire interval given by V_b. **model.m** uses the floating potential, to determine the parameter c from the other parameters. Next, depending on what currents the user has chosen to use in the fit, the current is calculated for the parameters and then returned in a vector.

Chapter 7

Results and discussion

Where we use our developed analysis tool on the first Earth flyby to evaluate the performance of the routines.

7.1 The first flyby of Earth

In the evening 4 March 2005 Rosetta blew past Earth at around 10 km/s. Before that, it had already spent several days in a plasma influenced by the presence of Earth. During a short period of time, Rosetta encountered a wide variety of different plasmas with a density ranging from a few, to several thousand particles per cubic centimetre. Thus, the Earth flyby provided the perfect opportunity to test the analysis routines and compare the results obtained with other instruments onboard Rosetta.

The Langmuir probes were supposed to collect data in intervals of a couple of minutes throughout the entire flyby, but unfortunately, because of an error in communications, there is a wide gap in the data collected, starting in the evening of the first of March and ending mid-day of the third of March. Despite this, there is plenty of data available, especially in the evening of the fourth of March, when Rosetta was closest to Earth, and exposed to a dense plasma-spheric plasma.

Using the models determined in Chapter 5 for the photoelectron current, Grafical, the application developed (see Appendix 6 and A) for this very reason, where put to the test.

The program was set to automatically fit the data for the period 1 March 2005 to 6 March 2005. The results follow.

7.2 V_{sc} and the plasma density

7.2.1 Spacecraft potential dependence on density

According to Pedersen{pedersen the plasma density should exhibit an exponential dependence on the spacecraft potential. This is easy to see from eq. 3.4, which shows that the density is proportional to the current. The equation for the equilibrium state of the probe, disregarding smaller current contributions such as ions etc. will be

$$I_{e0}\left(1+\frac{V_{sc}}{T_e}\right) = I_{ph0}e^{-\frac{V_{sc}}{T_{ph}}} \tag{7.1}$$

thus, the relation between the density and the spacecraft potential will be of the form

$$ln(n_e) = A - B\ ln\left(1+\frac{Vsc}{T_e}\right) - \frac{V_{sc}}{T_{ph}}. \tag{7.2}$$

We can see that the dependence is not strictly linear, but to a good approximation the linear part of the function will dominate the logarithmic part (i.e. the $\frac{V_{sc}}{T_{ph}}$ term will dominate the $ln\left(1+\frac{Vsc}{T_e}\right)$ term) and we can treat the relationship as linear. This means that to find a linear relationship between the logarithm of the density and the spacecraft potential would be a good sanity check for our routines.

It would also be good to perform the same analysis with independent data, thus comparing our data to theory as well as independent sources.

7.2.2 Comparison to ACE

To compare results we turned to data provided by the *Advanced Composition Explorer*, ACE[17]. ACE is a spacecraft maintained and developed by *the National Aeronautics and Space Administration* (NASA) which orbits about the L1 libration point, approximately 1.5 million km from Earth. It measures several parameters of the solar wind and its data can thus be used to compare with our results.

Because of the difference in position between Rosetta and ACE, Rosetta would measure the same plasma as ACE a couple of hours later. A time-shift was calculated for each data point and added to the data. In order to simplify the calculations any components of the solar wind velocity not in the GSE x-axis were omitted, since these amounted to at most 1/10 of the velocity component in the x-direction and could thus be considered negligible.

During Rosettas travel through the magnetosphere probe 2 was often in wake. Thus, to compare ACE-data and data from Rosetta measurements from probe 1 was needed. From about 22 o'clock at the fourth of March to about 23 o'clock at the fifth of March, probe 1 was not in sweep mode, but rather in its continuous voltage measurement mode (see section 4.1). The spacecraft potential that probe 1 measured in this mode could, with a better time resolution than if the probe had been in sweep mode, be plotted against the density given from ACE at the same time, see Figure 7.1. The linear relationship discussed in Section 7.2.1 can be seen, though it is quite diffuse.

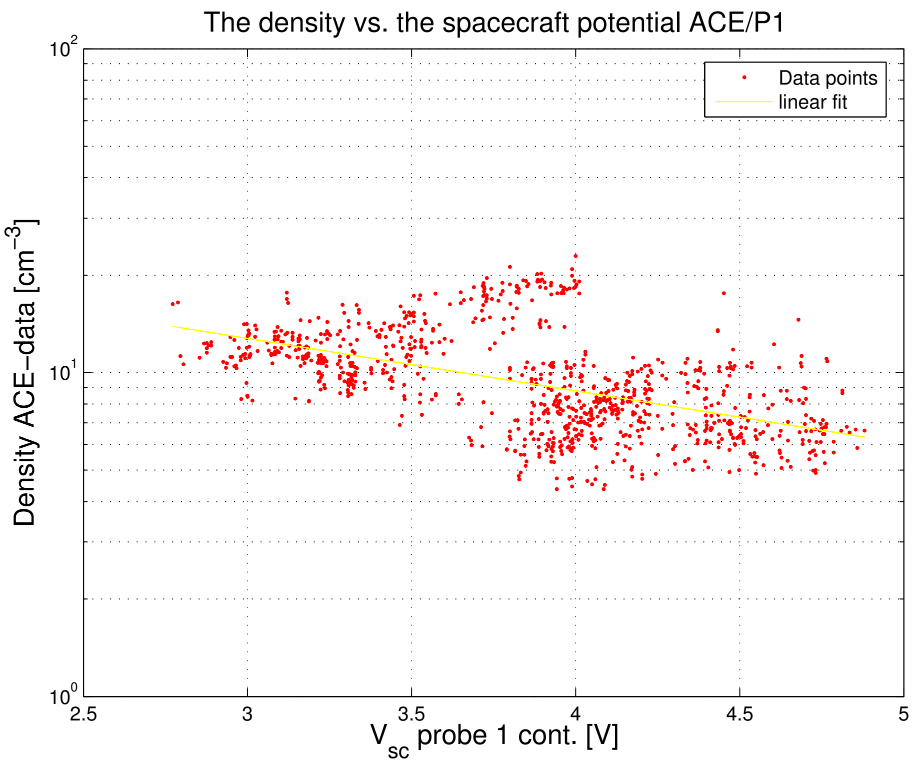


Figure 7.1: The plasma density obtained from ACE versus the spacecraft potential from probe 1.

By performing a linear fit we get a bid on our polynomial in equation (7.2).

$$log(n) = -0.1327 \cdot V_{sc} + 16.6969 \tag{7.3}$$

This result mean that for a given spacecraft potential, the density can be calculated from this simple equation.

After the completion of this work, an improved method by Haaland et. al [18] to calculate the time shift, has come to the authors knowledge. That method, which takes account of solar wind magnetic field density front directions, could possibly give a smaller spread to plots like Figure 7.1. The difference in time shift could amount to as much as a couple of hours.

Now the polynomial determined from the ACE-data is used with the potential obtained from the continuous voltage measurement performed by probe 1 during most of 5 March 2005. By using equation (7.2) the density can be determined. Figure 7.2 show a time series of the plasma density obtained from ACE, as well as the density from Rosetta.

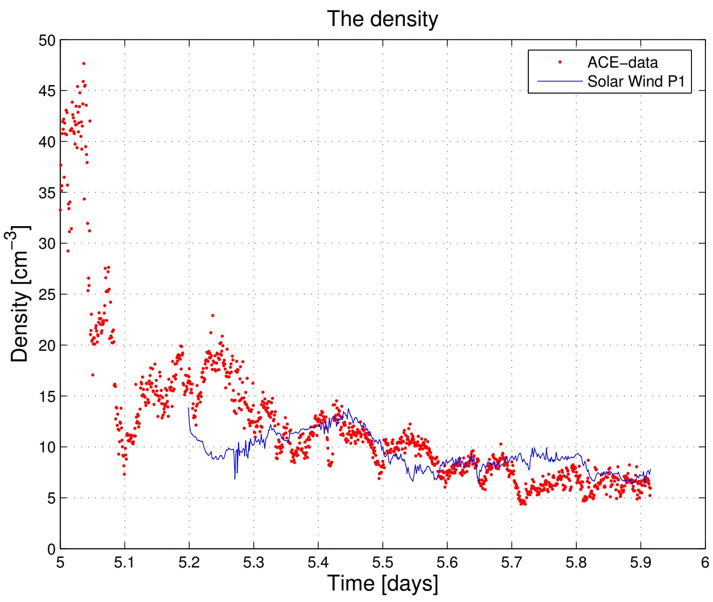


Figure 7.2: The density of the plasma over time from ACE (red) and from probe 1 (blue)

The overall correspondence is quite good, but there are of course differences in detail. This is to be expected because ACE is not in the exact same environment as Rosetta. Looking closely at Figure 7.2 it would seem like a shift of 2 hours in time would make the density from ACE and the density from the polynomial correspond better which could mean that the time shift is incorrectly calculated, as discussed above.

7.2.3 Comparison V_{sc} to n from LAP

Performing the same linear fit as in Section 7.2.2 but with density and s/c potential from LAP we get another bid on our polynomial in equation (7.2)

$$log(n) = -0.51405 \cdot V_{sc} + 19.202105 \tag{7.4}$$

These two equations may be good in different density regions. The equation from Rosetta, eq. (7.4) may be a good approximation at high densities while the one from ACE, eq. (7.3), may be a better fit at lower densities. This behaviour is also visible in Figure 7.3 below.

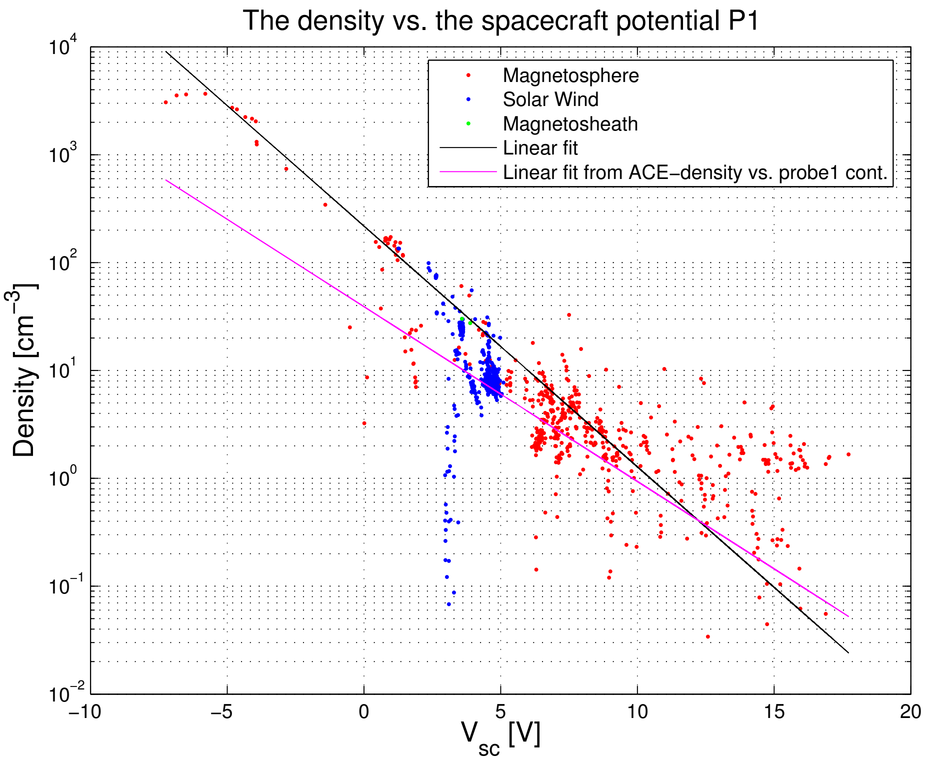


Figure 7.3: The logarithm of the density plotted against the spacecraft potential for probe 1. The different colours designate the different regions where the data points have been collected. Notice the linear relationship between the density and the potential. The black line is a linear fit to the data from Rosetta while the purple line is a linear fit the data taken from ACE.

Though it is clear that the density in Figure 7.3 exhibits the logarithmic relationship, it is remarkable that this relationship holds for negative spacecraft potentials as well. For negative spacecraft potentials the equilibrium equation 7.1 should not hold (see equations 3.6 and 3.9). That the line is so linear despite this could however be explained by considering that the solar panels are not outer part of the equipotential surfaces. So even though the spacecraft potential is negative, the solar panels (emitting most of the photoelectrons) are still be positive, making the photocurrent exhibit the same exponential behaviour which led to equation 7.2. Looking again at Figure 7.2 one can see that the data points start to bend to the right at high potentials which could indicate a second electron population.

7.3 The spacecraft potentials

Ideally the probes should measure the same spacecraft potential at the same time. Since the probes measure at different times (usually within a minute of each other or so) a simple function was written which finds the measurements of the probes that are closest in time to each other. If, for a given measurement of probe 1, no measurement of probe 2 within 4 minutes can be found, the data-point is thrown away. This way a set of data points could be determined and plotted against each other. Figure 7.4 is a plot of V_{sc} for probe 1 and probe 2.

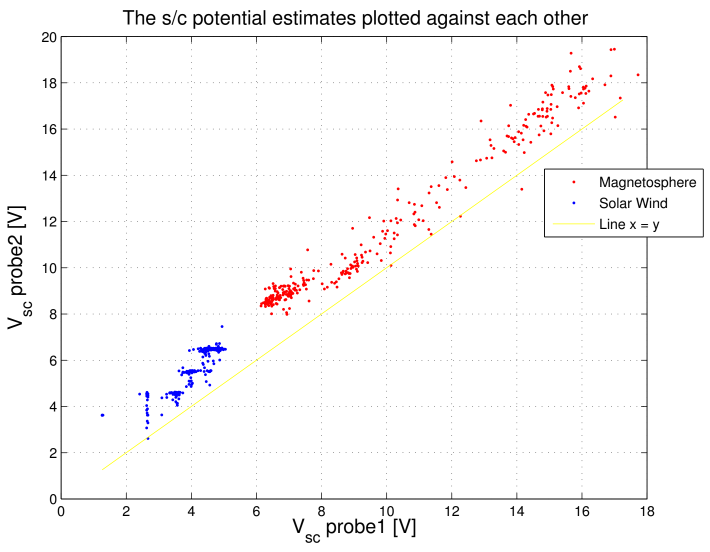


Figure 7.4: The spacecraft potentials plotted against each other. The linear relationship is seen to hold well. Notice the shift of 1-2 V between probe 1 and probe 2.

The differently coloured points come from different regions; the red points are measured in the magnetosphere while the blue ones are taken in the solar wind. The linear relationship holds really well but there is a shift in potential between the probes. Probe 1 measures about 1.4 V lower than probe 2 throughout. This can be explained by the fact that for most of this period probe 2 actually is in wake behind Rosetta. As was explained earlier (section 4.3.2), there are almost no ions in the wake while electrons may enter on the order of $-\frac{k_B T_e}{e}$. This means that the wake should be charged negatively to a potential approximately $-T_e$ since not many electrons have an energy greater than T_e so that charging stops at this level. This would hold if the Debye length is smaller than the wake. According to Table 6.1 of section 3.2 the Debye length in the solar wind is about 10 m, larger than the wake. To help us out we look at Poisson's equation. Regarding the shape of the wake as a cylinder with radius a, it will become

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0} = e\frac{n_e - n_i}{\epsilon_0} = \frac{en_0}{\epsilon_0}, \text{ r < a} \tag{7.5}$$

Solving this for a cylindrical geometry we get

$$\phi = \begin{cases} 0, & r > a \\ -\frac{k_B T_e}{e}\frac{a^2-r^2}{\lambda_d^2}, & r < a \end{cases} \tag{7.6}$$

Using a Debye length of about 7 m and electrons with energy 10 eV we get a shift of about 1 V, just as we can see from Figure 7.4.

7.4 Comparing the data - MIP

The *Mutual Impedance Probe*, MIP, on Rosetta is a good instrument for measuring plasma density in dense plasmas and thus useful for comparison. Unfortunately, when this thesis was written only the density for one point was available for comparison, but it can nevertheless show if the values determined by the routines are probable. Because of the uncertainty in the T_e-value we derive from the sweeps the density determined through MIP could be better than the density determined through LAP in the densest plasmas in the deep plasmasphere, and it is interesting to see what temperature we get by using the MIP estimate of the density and working backwards. This has been done. Table 7.1 show the density from MIP and LAP as well as the temperature from LAP at 22:15 on 4 March 2005.

	n [cm^{-3}]	T_e from routines [eV]	T_e using n from MIP [eV]
MIP	2964	N/A	N/A
LAP P1	3250	0.3105	0.3734
LAP P2	1853	0.2885	0.1127

Table 7.1: The density from MIP and LAP, as well as the temperature from LAP.

As can be seen the density determined from probe 2 is quite bad, but this was expected since probe 2 is in wake at the time. The value of the density determined from probe 1 is quite good, if on the high side. The temperature determined from LAP by the routines have not changed much from the initial value of 0.3 eV and can thus not be trusted (this is not surprising, as the voltage step of 0.25 V used in the sweeps is not sufficient for determining temperatures below around 0.5 V - there are simply too few data points on the exponential part of the sweep). The T_e values determined using the MIP density should be good and has a reasonable value. Nevertheless, the T_e value derived from LAP alone is within 25% of the combined MIP-LAP values and LAP and MIP agrees on density to within 10%.

7.5 Conclusion

The results obtained through the analysis routines seem to be consistent with other sources both on Rosetta (MIP) and on independent spacecrafts (ACE). In addition, the results show a consistency with theory, as can be seen from the linear relationship of the logarithm of the density vs. the spacecraft potential. There are unfortunately limitations as well. The determination of the electron temperature is too dependent on initial values given and as a result the temperature cannot be trusted. Accordingly, there will be an uncertainty in the absolute value given for the electron density, though the overall trend should be thrustworthy.

7.5.1 Outlook

Looking ahead, the next potential usage of the routines should be after the Mars flyby in february of 2007. Before they can be used to satisfaction, however, there are a couple of things which could be done in order to facilitate and improve the analysis substantially. Some possible work that can be done are itemised below.

- These limitations discussed above in Section 7.5 stem from the non-linear approach in the solution of the problem. The non-linear routines introduces an arbitrariness in the solution, in the sense that as soon as the required precision is met, the iterations stop irrespective of the actual parameter values determined. The practical way of solving these problems would include removing the routines using lsqcurvefit and re-writing them using theory and linear relationships. The advantage is that the programmer will have complete control of the analysis and the process will speed up considerably (a large part of the time consumption is located in the non-linear fitting), the main disadvantage being the considerable development time needed for such an approach (which is also the reason why it was not used in this thesis).
- Adding more models for the plasma, including several ion species where applicable, and taking effects such as shadowing of the booms etc. into consideration would make the program more general and better at handling special cases.
- Rewriting the program in an object-oriented programming language (e.g. C++), thus speeding up the analysis further.
- Improving ACE-Rosetta comparison by use of the improved time lag method by Haaland et al**??**.

These improvements mentioned above are by no means exhaustive, but it is a good starting list. The most important improvement, and perhaps the most time-consuming one, would be to refine the analysis. This should prove to be rewarding, making the results easier to understand and analyse.

Chapter 8

Acknowledgements

I would like to thank my supervisor Anders Eriksson at the Swedish Institute of Space Physics, Uppsala division (IRF-U), for helping me througout this endevour. Thank you for providing me with the project and giving tips and help whenever needed. Thank you for your constant enthusiasm and good spirits which really helped to spur me on when the job got though.

I would also like to thank Reine Gill (IRF-U) for providing me with his developed code and explaining software issues as well as physical problems for me.

Thanks goes out to Jan-Erik Wahlund (IRF-U) for showing me some of his analysis and software and giving tips on how to proceed.

Lastly I would like to thank the entire staff at IRF-U, for the nice company at the coffe-breaks and lunches and for providing a pleasant atmosphere to work in.

Bibliography

[1] A. I. Eriksson, R. Boström, R. Gill, L. Åhlén, S.-E. Jansson, J.-E. Wahlund, M. André, A. Mälkki, J. A. Holtet, B. Lybekk, A. Pedersen, L. G. Blomberg, and the LAP team. RPC-LAP: The Rosetta Langmuir probe instrument. *Space Sci. Rev.*, in press, 2006.

[2] European Space Agency. website, http://www.esa.int.

[3] M.G. Kivelson and C.R. Russel. *Introduction to Space Physics.* Cambridge University Press, 1997.

[4] website, http://www.windows.ucar.edu/tour/link=/glossary/plasmasphere.html.

[5] Magnus Billvik. The first Rosetta Earth flyby. Master's thesis, Uppsala University, Swedish Institute of Space Physics, 2005.

[6] Francis F. Chen. *Introduction to Plasma Physics and Controlled Fusion, Volume 1: Plasma Physics.* Plenum Press, first edition, 1975.

[7] Magnus Carlson. The Langmuir probes on Freja. Master's thesis, Uppsala University, Swedish Institute of Space Physics, 1994.

[8] J. Roussel and J. Berthelier. A study of the electrical charging of the Rosetta orbiter: 1. numerical model. *Journal of Geophysical Research*, 109:12 pp., 2004.

[9] H.M. Mott-Smith and I. Langmuir. The theory of collectors in gaseous discharges. *Physical Review*, 28:727–763, 1926.

[10] Anders I. Eriksson and Rolf Boström. Measurements of plasma density fluctuations and electric wave fields using spherical electrostatic probes. Technical Report 220, Swedish Institute of Space Physics, April 1995.

[11] provided by Rickard Lundin and Hans Nilsson, IRF Kiruna.

[12] Réjean J.L. Grard. Properties of the satellite photoelectron sheath derived from photoemission laboratory measurements. *Journal of Geophysical Research*, 78(16):2885–906, 1973.

[13] A. Pedersen. Solar wind and magnetosphere plasma diagnostics by spacecraft electrostatic potential measurements. *Annales Geophysicae*, 13(2):118–29, 1995.

[14] 2006. Carr et al.

[15] website, http://www.space.irfu.se/isdat/.

[16] R. Gill. Reused code.

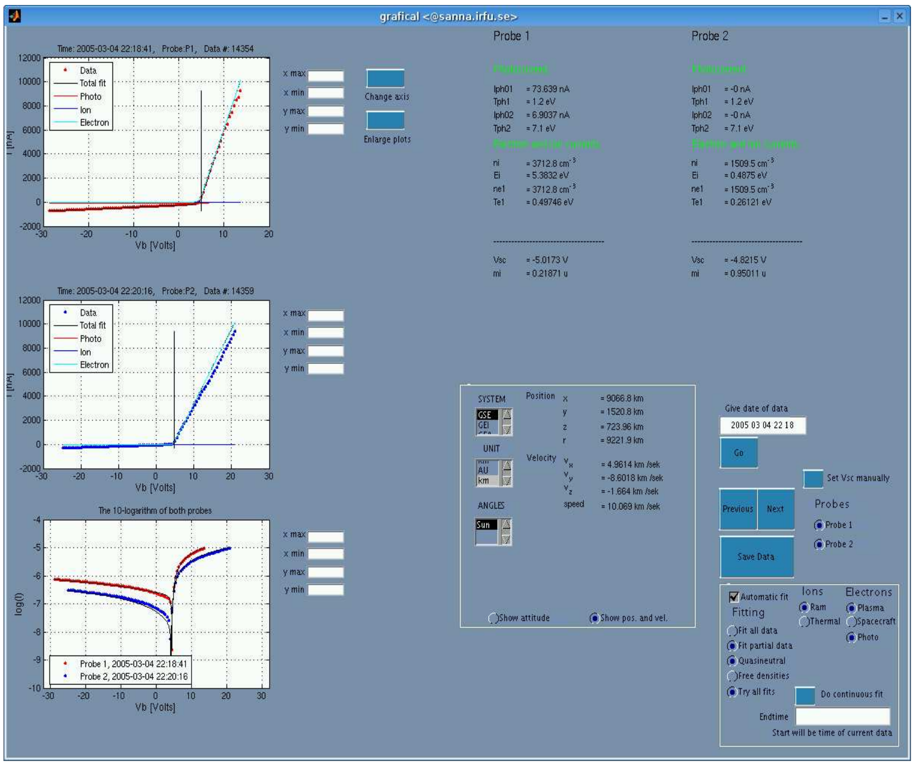
[17] website, http://www.srl.caltech.edu/ACE/.

[18] S. Haaland, G. Paschmann, and B. U. Ö. Sonnerup. Comment on "a new interpretation of weimer et al.'s solar wind propagation delay technique" by bargatze et al. *J. Geophys. Res.*, 111:A06102, doi:10.1029/2005JA011376, 2006.

Appendix A

User's guide to Grafical

*In which the graphical user interface (*GUI*), Grafical, is explained.*


grafical <@sanna.irfu.se>
Time: 2005-03-04 22:18:41, Probe:P1, Data #: 14354
Data
Total fit
Photo
Ion
Electron
I [nA]
Vb [Volts]
x max
x min
y max
y min
Change axis
Enlarge plots
Time: 2005-03-04 22:20:16, Probe:P2, Data #: 14359
The 10-logarithm of both probes
log(I)
Probe 1, 2005-03-04 22:18:41
Probe 2, 2005-03-04 22:20:16
SYSTEM
GSE
GEI
UNIT
AU
km
ANGLES
Sun
Go
Previous
Next
Save Data
Probes
Ions
Electrons

```
Probe 1                        Probe 2

Iph01   = 73.639 nA            Iph01   = -0 nA
Tph1    = 1.2 eV               Tph1    = 1.2 eV
Iph02   = 6.9037 nA            Iph02   = -0 nA
Tph2    = 7.1 eV               Tph2    = 7.1 eV

ni      = 3712.8 cm^-3         ni      = 1509.5 cm^-3
Ei      = 5.3832 eV            Ei      = 0.4875 eV
ne1     = 3712.8 cm^-3         ne1     = 1509.5 cm^-3
Te1     = 0.49746 eV           Te1     = 0.26121 eV

------------------------------ ------------------------------

Vsc     = -5.0173 V            Vsc     = -4.8215 V
mi      = 0.21871 u            mi      = 0.95011 u

Position  x      = 9066.8 km
          y      = 1520.8 km
          z      = 723.96 km
          r      = 9221.9 km
Velocity  vx     = 4.9614 km /sek
          vy     = -8.6018 km /sek
          vz     = -1.664 km /sek
          speed  = 10.069 km /sek

( ) Show attitude     (•) Show pos. and vel.

Give date of data: 2005 03 04 22 18
Set Vsc manually
Probes: (•) Probe 1  (•) Probe 2

[✓] Automatic fit
Fitting                 Ions          Electrons
( ) Fit all data        (•) Ram       (•) Plasma
(•) Fit partial data    ( ) Thermal   ( ) Spacecraft
(•) Quasineutral                      (•) Photo
( ) Free densities
(•) Try all fits        Do continuous fit
Endtime
Start will be time of current data
```

Figure A.1: A screen-capture of the GUI

A.1 Appearance and plots

When in MATLAB, the interface is started by simply entering "grafical", while in the directory containing the m-files of the application. The interface looks like Figure A.1. There are three plots, these are for (from top to bottom) probe 1 in linear scale, probe 2 in linear scale and finally both probe 1 and 2 in the same plot, logarithmically plotted. To the right of each plot there are slots, in which boundaries for the axes can be entered (see Figure A.2). By pressing the button **"change axis"**, the plots will update the axes. If the button is not pressed, the plots will change axes when the next plotting is done (e.g. when stepping forward to the next sweep). The button **"enlarge plots"**, plots the graphs in separate windows, in which the plots can be zoomed and manipulated.

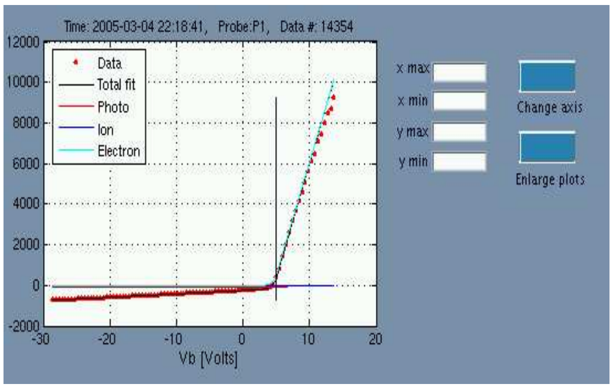


Figure A.2: The buttons **"change axis"** and **"enlarge plots"** can be useful when analysing data.

A.1.1 Loading data

The first thing to note is that there exist two radiobuttons, in the lower right corner, just above the panel marked "fitting panel" (see Figure A.3), one for each probe. This way one can choose which probe to look at (or if one wishes, both at the same time).

In order to analyse data one first need to obtain this data. The way to do this is by entering the date and time of the data one wishes to obtain in the slot to the far right of the GUI (right under the text "give date of data") and pressing the button **"go"**. The time should be specified in the format

yyyy mm dd hh mm ss

One need not enter the entire date and time. The application will search for the first data possible in the last digits entered. For example, if one enters **2004 03 05 22** the software will look for the first data set at ten o'clock of the fifth of march 2004. If the user has chosen to look at both probes, the first data set found will be for probe 1 and then the closest possible data set for probe 2, forward in time, will be selected (since there is no possibility to obtain data from both probes at exactly the same time).

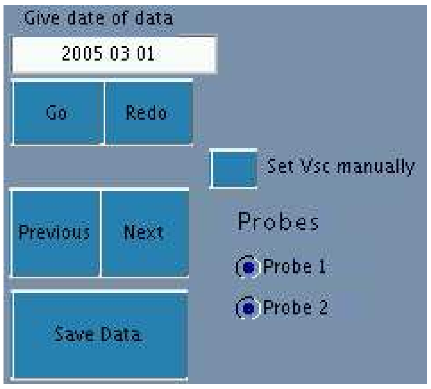


Figure A.3: The buttons used for loading and browsing data

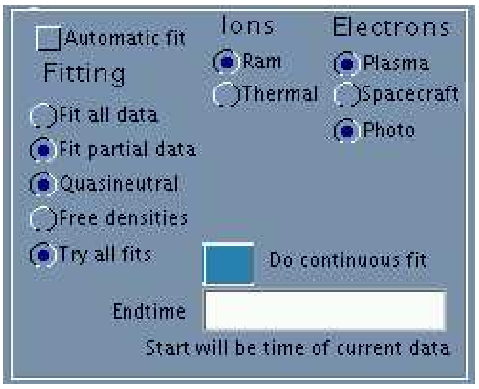
To browse the data the two buttons **"previous"** and **"next"** can be used. Again, it will be probe 1 that determines the next data set. Pressing "previous", probe 1 will step one set back, and *then* probe 2 will be moved to the set forward in time that is closest to the data set of probe 1. What this mean is that while probe 1 will always respond by going forward and backward as the buttons are pressed, probe 2 may stand still. An example may clarify things

Probe 1 is the data set ***2005-03-04 at 22:15:41****. The closest data set for probe 2 is* ***2005-03-04 at 22:18:30****. By pressing "next" probe 1 will step to* its *next data set, which is* ***2005-03-04 at 22:17:25****. Now probe 2 will be moved to the set which is closest to* ***2005-03-04 at 22:17:25****, but this is still* ***2005-03-04 at 22:18:30*** *and probe 2 will thus stand still.*

The button **"redo"** will load the current data set again. This is useful for comparing different models of the current on the same data.

A.1.2 The fitting panel

Located in the lower right corner, the fitting panel is used to analyse the data of the sweeps, utilising the automatic fitting routines developed (see Appendix 6). Figure A.4 is a screen caption of the panel.



☐ Automatic fit

Fitting
- Fit all data
- [✓ Fit partial data]
- [✓ Quasineutral]
- Free densities
- [✓ Try all fits]

Ions
- [✓ Ram]
- Thermal

Electrons
- [✓ Plasma]
- Spacecraft
- [✓ Photo]

Do continuous fit

Endtime

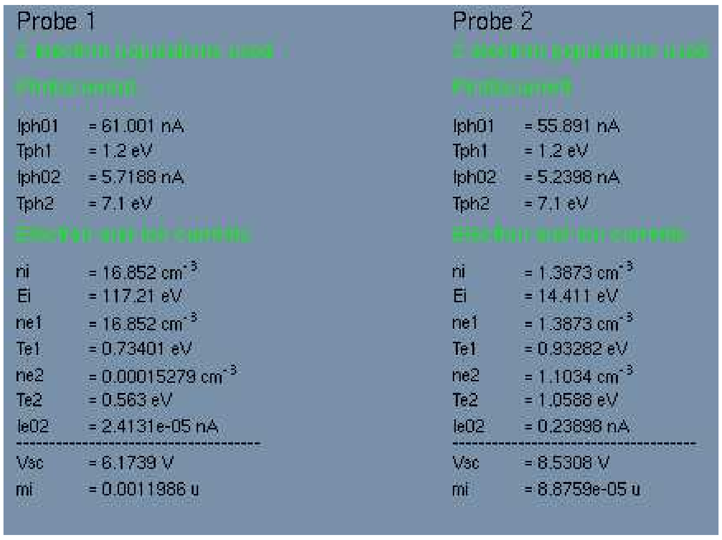
Start will be time of current data

Figure A.4: The panel used to tailor the fitting procedure

By selecting **"automatic fit"**, the program will try to fit the sweeps in the plots. Leaving it ticked, every new sweep will be analysed. The process of analysing takes some time and will slow down the browsing considerably. When the fitting is finished the results for both probes (or if only one is chosen, the information for that probe) will be displayed in the top right corner and the fit and all the constituent currents will be plotted together with the data. The information displayed is itemised below (such as it is displayed in the GUI)

$I_{ph,01}$	- Photoelectron current one
$I_{ph,02}$	- Photoelectron current two
$T_{ph,1}$	- Photoelectron temperature one, in eV
$T_{ph,2}$	- Photoelectron temperature two, in eV
n_i	- Ion density, in particles per cm^3
E_i	- Ion energy, in eV
$n_{e,1}$	- Plasma electron density, in particles per cm^3
$T_{e,1}$	- Plasma electron temperature, in eV
$n_{e,2}$	- Spacecraft photoelectron density, in particles per cm^3
$T_{e,2}$	- Spacecraft photoelectron temperature, in eV
I_{e02}	- Spacecraft photoelectron current, in nA
V_{sc}	- Spacecraft potential, in Volts
m_i	- effective ion mass (if applicable), in u

The reason why the spacecraft photoelectrons are given both as a current and a density lies in the interpretation of the data. Since the spacecraft photoelectrons are electrons stripped from the s/c body and collected by the probe we do not really know how homogenous they are, making them hard to discuss in terms of densities. On the other hand, giving them in the form of a density, a comparison between the plasma electrons and spacecraft electrons is easier to perform. This is the reasons for having both forms in the GUI. Figure A.5 below show what the information obtained trough the fitting can look like.



```
Probe 1                                   Probe 2
2 electron populations used               2 electron populations used
Photocurrent                              Photocurrent
Iph01    = 61.001 nA                      Iph01    = 55.891 nA
Tph1     = 1.2 eV                         Tph1     = 1.2 eV
Iph02    = 5.7188 nA                      Iph02    = 5.2398 nA
Tph2     = 7.1 eV                         Tph2     = 7.1 eV
Electron and ion currents                 Electron and ion currents
ni       = 16.852 cm^-3                   ni       = 1.3873 cm^-3
Ei       = 117.21 eV                      Ei       = 14.411 eV
ne1      = 16.852 cm^-3                   ne1      = 1.3873 cm^-3
Te1      = 0.73401 eV                     Te1      = 0.93282 eV
ne2      = 0.00015279 cm^-3               ne2      = 1.1034 cm^-3
Te2      = 0.563 eV                       Te2      = 1.0588 eV
Ie02     = 2.4131e-05 nA                  Ie02     = 0.23898 nA
------------------------------------      ------------------------------------
Vsc      = 6.1739 V                       Vsc      = 8.5308 V
mi       = 0.0011986 u                    mi       = 8.8759e-05 u
```

Figure A.5: A screen caption showing how the physical parameters are displayed in the GUI

There are several radio buttons which can be selected when fitting the models to the data. This provides the opportunity to tailor the fitting for a specific kind of data. The buttons and what they mean are given below.

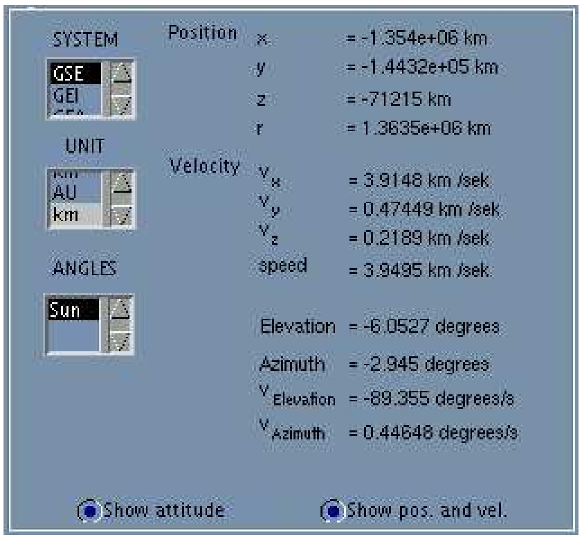
Button	**Fitting**
Fit all	By selecting this button all data points of the sweep are analysed.
Fit partial	Here only points below $V_b = 8$ volts are used in the analysis.
Quasineutral	This button enforces quasineutrality between the plasma electrons and the ions (though not for the spacecraft electrons).
Free densities	Here the densities are set free, so that the electrons and the ions can have different densities.
Try all	When selecting this button all different permutations of models are tried and compared (from all categories, i.e. from the fitting, ion and electron category). The different fits are carried out in a specific order, and some fits use the results from the fit before them as initial values. Out of all permutations, the best one is determined and displayed. A word of caution: this is a very slow process since it must try all different fits, typically 10-20 times slower than a normal fit.
	Ions
Ram	Ram ions are used in the fit, see Section 3.4.2.
Thermal	Thermal ions are used in the fit, see Section 3.4.1. Ram and thermal models are mutually exclusive, by selecting one the program automatically deselects the other.
	Electrons
Plasma	This button allows the program to use plasma electrons in the fit.
Spacecraft	Selecting this enables spacecraft photoelectrons in the fitting.
Photo	Photoelectrons emitted from the probe are taken into account.

Unfortunately not all combinations can be used (i.e. they have not been implemented). Plasma electrons must always be used in the fit, but the button for selecting them is still in the GUI. This will make it easy to include this option in a later version of the application. When a forbidden permutation is selected the GUI will automatically select another, allowed combination.

The button **"Do continuous fit"** is used in conjunction with the editor just below it. In the editor the end time should be specified in the same manner as the start time is specified in the editor by the **"Go"** button. Now the GUI will fit all data between the start and the end time with the choices set for the fitting. Usually when using the continuous fit, the option of trying all fits will be used, since Rosetta will probably encounter different plasmas over a lengthy period of time. The information of the fit will be saved in a .mat file in the same format as when using **"Save Data"**. See Section A.1.4.

A.1.3 The position and velocity panel

This panel is located in the middle of the GUI. It is used to get the position and velocity, as well as the attitude, for Rosetta at the given time of the data set (if both probes are used, it is the time of probe 1 that are used for the position). In order to see the position and velocity the button **"show position and velocity"** must be selected and to see the attitude one must select the button **"show attitude"**.



```
Position  x              = -1.354e+06 km
          y              = -1.4432e+05 km
          z              = -71215 km
          r              = 1.3635e+06 km
Velocity  Vx             = 3.9148 km /sek
          Vy             = 0.47449 km /sek
          Vz             = 0.2189 km /sek
          speed          = 3.9495 km /sek

          Elevation      = -6.0527 degrees
          Azimuth        = -2.945 degrees
          V Elevation    = -89.355 degrees/s
          V Azimuth      = 0.44648 degrees/s

(●) Show attitude          (●) Show pos. and vel.
```

Figure A.6: The position and velocity panel

By selecting the option of showing the position and attitude the process of showing data will slow down. This is because the program must first read a file containing specific position data and then calculate Rosettas current position. The program can save information about certain specific time intervals (such as the different Earth flyby's and the Mars flyby), so these need only be loaded once, thus speeding up the process for the next data set.

The drop-boxes in the upper left corner of the panel provide the option of choosing the unit and frame of reference for the position and the velocity to be given in. The attitude is always given as the radius vector pointing to the sun in a reference frame fixed in Rosetta. The y-axis is along the solar panels and the negative x-axis point out through the side of the spacecraft on which the probes are located.

A.1.4 Save Data

Saving the current data is done by pressing the **"Save Data"** button. Only the fit of the *current* set will be saved. In order to save several sets the user must utilise the **"Do continuos fit"** option (see Section A.1.2). The information will be saved in a .txt file named

data_probeX_timeYYYYMMDDTHHMMSS.txt

where X can be either 1 or 2. The information from the same data set is saved on the same row, in several columns. If several sets of data are saved they will be added to the existing file, on the next row. Below, the information that is saved in each row for a single data set, is given.

Column	**Saved information**
1	Year
2	Month
3	Day
4	Hour
5	Minute
6	Second
7	Code for coordinate system; 1 = GSE, 2 = GEI, 3 = GEA, 4 = HEI and 5 = HEA
8-10	Position coordinates; x, y and z in [m]
11-13	Velocity coordinates; v_x, v_y and v_z [m/s]
14-19	Attitude of Rosetta; ϕ, θ, ρ, v_ϕ, v_θ and v_ρ
20	The probe used, 1 or 2
21	The kind of sweep looked at; 1 = up, 2 = down, 3 = up and down and 4 = down and up
22	Minimum V_b in [V]
23	Maximum V_b in [V]
24	Step size in [V]
25	Number of data points in the original sweep
26	The time the sweep took in [s]
27-35	The mathematical parameters in the following order: a b c d g h Vsc f z
36	The error
37	The model used: 1 = plasma electrons, ram ions and partial data sweep 11 = plasma electrons, ram ions and full (all) data sweep 2 = plasma and photoelectrons, ram ions and partial data sweep 21 = plasma and photoelectrons, ram ions and full data sweep 3 = plasma electrons, thermal ions and partial data sweep 31 = plasma electrons, thermal ions and full data sweep 4 = plasma and photoelectrons, thermal ions and partial data sweep 41 = plasma and photoelectrons, thermal ions and full data sweep 5 = plasma and s/c-photoelectrons, ram ions and full data sweep 51 = plasma and s/c-photoelectrons, ram ions and partial data sweep 6 = plasma, photo- and s/c-photoelectrons, ram ions and full data sweep 61 = plasma, photo- and s/c-photoelectrons, ram ions and partial data sweep 7 = plasma and s/c-photoelectrons, thermal ions and full data sweep 71 = plasma and s/c-photoelectrons, thermal ions and partial data sweep 8 = plasma, photo- and s/c-photoelectrons, thermal ions and full data sweep 81 = plasma, photo- and s/c-photoelectrons, thermal ions and partial data sweep
38	If **try all** is used, all models will be saved. A column containing the value NaN will separate the models and then the parameters, error and what kind of model used will be given (11 columns in total).

A.1.5 Set spacecraft potential manually

This button (see Figure A.3) can be used to set the spacecraft potential manually if one is unsatisfied with the potential the automatic process has found. When the button has been pressed the left mousebutton is used to place the spacecraft potential in the plot. The GUI will show where the potential has been set. Clicking the left mouse button again designates that the spacecraft potential is satisfactory set. To redo the procedure press the right mousebutton. This step is repeated until the spacecraft potential has been set for both probes. Now this spacecraft potential will be used as the initial guess for V_{sc} in the nonlinear fit instead of the potential obtained through the second derivative (see Section 6.2.2).

Appendix B

The software routines

In which the developed code is gathered.

B.1 Main program

B.1.1 grafical.m

```
function varargout = grafical(varargin)
% GRAFICAL M-file for grafical.fig
%      GRAFICAL, by itself, creates a new GRAFICAL or raises the existing
%      singleton*.
%
%      H = GRAFICAL returns the handle to a new GRAFICAL or the handle to
%      the existing singleton*.
%
%      GRAFICAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GRAFICAL.M with the given input arguments.
%
%      GRAFICAL('Property','Value',...) creates a new GRAFICAL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before grafical_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to grafical_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help grafical

% Last Modified by GUIDE v2.5 08-Feb-2006 13:41:23

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @grafical_OpeningFcn, ...
                   'gui_OutputFcn',  @grafical_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before grafical is made visible.
function grafical_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to grafical (see VARARGIN)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Load the structures, connect to database%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ROS  = Rosetta_info;
PAR1 = parameter_struct;
PAR2 = parameter_struct;
if(ROS.DB == 0)     %Connect to Database only if not already connected
    ROS.DB = Mat_DbOpen(ROS.Database);  %Connecting to ISDAT
end
if(ROS.DB ~= 0)     %If connected to ISDAT
    [ROS.DAT_CONT1, ROS.DAT_DUR1] = isGetContentLite(ROS.DB, ROS.PROJ,...
        ROS.MEM, ROS.INST, ROS.SIG, ROS.SEN1, ROS.CHAN1, ROS.PARAM);
    [ROS.DAT_CONT2, ROS.DAT_DUR2] = isGetContentLite(ROS.DB, ROS.PROJ,...
        ROS.MEM, ROS.INST, ROS.SIG, ROS.SEN2, ROS.CHAN1, ROS.PARAM);
end

%Set starting handles

%handles for data
handles.where1         = 0;      %Setting some handles
handles.where2         = 0;
handles.ROS            = ROS;    %Sets ROS as one of the handles
handles.PAR1           = PAR1;   %Sets PAR1 as one of the handles
handles.PAR2           = PAR2;   %Sets PAR2 as one of the handles
handles.data           = 0;      %data = 0 no data, data = 1 data
handles.start          = 0;      %Where to start dataplotting
handles.end_of_data    = 0;      %Where to stop the continous fitting
handles.fb             = 0;      %0 stands for forward

%Handles for misc.
handles.Vsc_manual     = 0;
handles.force          = 0;
handles.debug          = 0;      %Debug mode on or off?

%handles for choosing axis
handles.xmax_co        = 0;
handles.xmin_co        = 0;
handles.xmax_co2       = 0;
handles.xmin_co2       = 0;
handles.xmax_co3       = 0;
handles.xmin_co3       = 0;
handles.ymax_co        = 0;
handles.ymin_co        = 0;
handles.ymax_co2       = 0;
handles.ymin_co2       = 0;
handles.ymax_co3       = 0;
handles.ymin_co3       = 0;

%handles for probes
handles.show_probe1    = 1;         %Show both probes from start
handles.show_probe2    = 1;

%handles for attitude and position
handles.system_choice  = 1;         %GSE Default
handles.unit_choice    = 1;         %Re default
handles.show_posvel    = 0;         %Do not show as default
handles.show_att       = 0;         %Not showing attitude as default
handles.events_loaded  = zeros(5,3);
handles.RVA_data       = 0;
handles.enl_plots      = 0;
handles.vsc            = 0;

%handles for fitting
handles.fit                = 0;
handles.fit_all            = 0;             %From start only fit_ion
handles.fit_partial        = 1;             %will be used and compared
handles.fit_free_densities = 0;             %No free densities from start
                                            %since it should be
handles.fit_quasineutral   = 1;             %quasineutral
handles.ion_mass           = 1.6726e-27;    %1 proton mass from start (u)
handles.fit_try_all        = 0;             %Try all possibilities, off
                                            %from start
%ions
handles.i_ram           = 1;    %Start with default ram ions
handles.i_therm         = 0;
%handles.i_mixed         = 0;
%electrons
handles.e_plasma        = 1;    %Start with e_plasma and e_photo as default
handles.e_sc            = 0;
handles.e_photo         = 1;

%clear the axes
axes(handles.parameter_edit);
axis off;
axes(handles.output_text);
axis off;
axes(handles.pos_x_disp);
axis off;
axes(handles.pos_y_disp);
axis off;
axes(handles.pos_z_disp);
axis off;
axes(handles.pos_r_disp);
axis off;
axes(handles.vel_x_disp);
axis off;
axes(handles.vel_y_disp);
axis off;
axes(handles.vel_z_disp);
axis off;
```

```
axes(handles.vel_speed_disp);
axis off;
axes(handles.pos_phi_disp);
axis off;
axes(handles.pos_theta_disp);
axis off;
axes(handles.vel_phi_disp);
axis off;
axes(handles.vel_theta_disp);
axis off;

%Read the data needed%
%timediff = datenum([-4713 1 1 12 0 0])+327;
%[handles.RE, handles.datee,handles.jde]   = readjpltraj('e_traj_hea.txt');      %jordens bana i hea
%[handles.VE, handles.dateve,handles.jdve] = readjpltraj('ve_traj_hea.txt');     %jordens hast i hea
%[handles.RS, handles.dates,handles.jds] = readjpltraj('s_traj_gea.txt');        %solens bana i gea
%handles.jde  = handles.jde + timediff;
%handles.jdve = handles.jdve + timediff;
%handles.jds  = handles.jds + timediff;

% Choose default command line output for grafical
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes grafical wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = grafical_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles;

% --- Executes on button press in button_back.
function button_back_Callback(hObject, eventdata, handles)
% hObject    handle to button_back (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
        axes(handles.output_text);
        cla;
        text(0.0,1.0,['Work in progress...'],'FontSize',14,'Color','green');
        pause(0.1)
        handles.fb      = 1;
        handles.where1  = handles.where1-1;
        handles.where2  = handles.where2-1;
        [handles.ROS,handles.PAR1,handles.PAR2,handles.events_loaded,...
            handles.RVA_data] = get_datagraf(handles.ROS,handles.PAR1,...
            handles.PAR2,hObject,eventdata,handles);
        handles.where1  = handles.ROS.where1;
        handles.where2  = handles.ROS.where2;
        guidata(hObject, handles);
        plot_sweepgraf2(handles.ROS,handles.PAR1,handles.PAR2,handles);
        displayinfo(handles.ROS,handles.PAR1,handles.PAR2,handles);

% --- Executes on button press in button_next.
function button_next_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to button_next (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
        axes(handles.output_text);
        cla;
        text(0.0,1.0,['Work in progress...'],'FontSize',14,'Color','green');
        pause(0.1);
        handles.fb      = 0;
        handles.where1  = handles.where1+1;
        handles.where2  = handles.where2+1;
        [handles.ROS,handles.PAR1,handles.PAR2,handles.events_loaded,...
            handles.RVA_data] = get_datagraf(handles.ROS,handles.PAR1,...
            handles.PAR2,hObject,eventdata,handles);
        handles.where1  = handles.ROS.where1;
        handles.where2  = handles.ROS.where2;
        guidata(hObject, handles);
        plot_sweepgraf2(handles.ROS,handles.PAR1,handles.PAR2,handles);
        displayinfo(handles.ROS,handles.PAR1,handles.PAR2,handles);

function editor_Callback(hObject, eventdata, handles)
% hObject    handle to editor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editor as text
%        str2double(get(hObject,'String')) returns contents of editor as a double

handles.start = get(hObject,'String');
hajime        = str2num(handles.start);;
stl           = size(hajime,2);
if (stl < 1)
    return;
end
if (stl == 1)
    hajime(2:3) = 0;
end
if (stl == 2)
    hajime(3) = 0;
end
if (stl == 4)
    hajime(5:6) = 0;
end
if (stl == 5)
    hajime(6) = 0;
end
handles.start = datenum(hajime);
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function editor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in button_go.
```

```
function button_go_Callback(hObject, eventdata, handles)
% hObject    handle to button_go (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
        axes(handles.output_text);
        cla;
        text(0.0,1.0,['Work in progress...'],'FontSize',14,'Color','green');
        pause(0.1);
        handles.fb                = 0;
        handles.where1            = handles.start;
        handles.where2            = 0;
        [handles.ROS,handles.PAR1,handles.PAR2,handles.events_loaded,...
            handles.RVA_data] = get_datagraf(handles.ROS,handles.PAR1,...
            handles.PAR2,hObject,eventdata,handles);
        handles.where1            = handles.ROS.where1;
        handles.where2            = handles.ROS.where2;
        guidata(hObject, handles);
%        if handles.ROS.where1 == 0 && handles.ROS.where2 == 0
%            return;
 %       end
        plot_sweepgraf2(handles.ROS, handles.PAR1,handles.PAR2, handles);
        displayinfo(handles.ROS,handles.PAR1,handles.PAR2,handles);

function parameter_edit_Callback(hObject, eventdata, handles)
% hObject    handle to parameter_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of parameter_edit as text
%        str2double(get(hObject,'String')) returns contents of parameter_edit as a double


% --- Executes during object creation, after setting all properties.
function parameter_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to parameter_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Iph_02_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Iph_02_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in save_data.
function save_data_Callback(hObject, eventdata, handles)
% hObject    handle to save_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
save_data(handles,handles.ROS,handles.PAR1,1);
save_data(handles,handles.ROS,handles.PAR2,2);

% --- Executes on button press in set_Vsc.
function set_Vsc_Callback(hObject, eventdata, handles)
% hObject    handle to set_Vsc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (handles.show_probe1 == 1)
    handles.PAR1.Vsc          = scpot_manual(handles,handles.PAR1.Vsc,1);
    handles.Vsc_manual        = 1;
    guidata(hObject,handles);
    [handles.PAR1,handles.events_loaded,handles.RVA_data] = gothrough(handles.ROS.V_DAT1,...
         handles.ROS.I_DAT1,handles.PAR1,handles.ROS,handles,1);
end
if (handles.show_probe2 == 1)
    handles.PAR2.Vsc          = scpot_manual(handles,handles.PAR2.Vsc,2);
    handles.Vsc_manual        = 1;
    guidata(hObject,handles);
    [handles.PAR2,handles.events_loaded,handles.RVA_data] = gothrough(handles.ROS.V_DAT2,...
         handles.ROS.I_DAT2,handles.PAR2,handles.ROS,handles,2);

end

handles.Vsc_manual        = 0;
guidata(hObject,handles);
plot_sweepgraf2(handles.ROS, handles.PAR1,handles.PAR2,handles);
displayinfo(handles.ROS,handles.PAR1,handles.PAR2,handles);

% --- Executes on button press in Vsc_free.
function Vsc_free_Callback(hObject, eventdata, handles)
% hObject    handle to Vsc_free (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Vsc_free
handles.Vsc_free = get(hObject,'Value');
guidata(hObject,handles);

% --- Executes on button press in foce_param.
function foce_param_Callback(hObject, eventdata, handles)
% hObject    handle to foce_param (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of foce_param
handles.force = get(hObject,'Value');
guidata(hObject,handles);
if handles.force == 1
    handles.PAR1 = force_model(handles.ROS.V_DAT1,handles.ROS.I_DAT1,...
        handles.PAR1,handles,1);
    handles.PAR2 = force_model(handles.ROS.V_DAT2,handles.ROS.I_DAT2,...
        handles.PAR2,handles,2);
    guidata(hObject,handles);
    displayinfo(handles.ROS,handles.PAR1,handles.PAR2,handles);
    plot_sweepgraf2(handles.ROS,handles.PAR1,handles.PAR2,handles);
end

function xmax_coord_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24 as a double
handles.xmax_co = str2double(get(hObject,'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function xmax_coord_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function xmin_coord_Callback(hObject, eventdata, handles)
% hObject    handle to xmin_coord (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xmin_coord as text
%        str2double(get(hObject,'String')) returns contents of xmin_coord as a double
handles.xmin_co = str2double(get(hObject,'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function xmin_coord_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xmin_coord (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ymax_coord_Callback(hObject, eventdata, handles)
% hObject    handle to ymax_coord (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ymax_coord as text
%        str2double(get(hObject,'String')) returns contents of ymax_coord as a double
handles.ymax_co = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function ymax_coord_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ymax_coord (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ymin_coord_Callback(hObject, eventdata, handles)
% hObject    handle to ymin_coord (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ymin_coord as text
%        str2double(get(hObject,'String')) returns contents of ymin_coord as a double
handles.ymin_co = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function ymin_coord_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ymin_coord (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function xmax_coord2_Callback(hObject, eventdata, handles)
% hObject    handle to xmax_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xmax_coord2 as text
%        str2double(get(hObject,'String')) returns contents of xmax_coord2 as a double
handles.xmax_co2 = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function xmax_coord2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xmax_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function xmin_coord2_Callback(hObject, eventdata, handles)
% hObject    handle to xmin_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xmin_coord2 as text
%        str2double(get(hObject,'String')) returns contents of xmin_coord2 as a double
handles.xmin_co2 = str2double(get(hObject, 'String'));
```

```
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function xmin_coord2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xmin_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ymax_coord2_Callback(hObject, eventdata, handles)
% hObject    handle to ymax_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ymax_coord2 as text
%        str2double(get(hObject,'String')) returns contents of ymax_coord2 as a double
handles.ymax_co2 = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function ymax_coord2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ymax_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ymin_coord2_Callback(hObject, eventdata, handles)
% hObject    handle to ymin_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ymin_coord2 as text
%        str2double(get(hObject,'String')) returns contents of ymin_coord2 as a double
handles.ymin_co2 = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function ymin_coord2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ymin_coord2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function xmax_coord3_Callback(hObject, eventdata, handles)
% hObject    handle to xmax_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xmax_coord3 as text
%        str2double(get(hObject,'String')) returns contents of xmax_coord3 as a double
handles.xmax_co3 = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function xmax_coord3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xmax_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function xmin_coord3_Callback(hObject, eventdata, handles)
% hObject    handle to xmin_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xmin_coord3 as text
%        str2double(get(hObject,'String')) returns contents of xmin_coord3 as a double
handles.xmin_co3 = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function xmin_coord3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xmin_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ymax_coord3_Callback(hObject, eventdata, handles)
% hObject    handle to ymax_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ymax_coord3 as text
%        str2double(get(hObject,'String')) returns contents of ymax_coord3 as a double
handles.ymax_co3 = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function ymax_coord3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ymax_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ymin_coord3_Callback(hObject, eventdata, handles)
% hObject    handle to ymin_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ymin_coord3 as text
%        str2double(get(hObject,'String')) returns contents of ymin_coord3 as a double
handles.ymin_co3 = str2double(get(hObject, 'String'));
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function ymin_coord3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ymin_coord3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in system_list.
function system_list_Callback(hObject, eventdata, handles)
% hObject    handle to system_list (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns system_list contents as cell array
%        contents{get(hObject,'Value')} returns selected item from system_list
handles.system_choice = get(hObject,'Value');
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function system_list_CreateFcn(hObject, eventdata, handles)
% hObject    handle to system_list (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in unit_list.
function unit_list_Callback(hObject, eventdata, handles)
% hObject    handle to unit_list (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns unit_list contents as cell array
%        contents{get(hObject,'Value')} returns selected item from unit_list
handles.unit_choice = get(hObject,'Value');
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function unit_list_CreateFcn(hObject, eventdata, handles)
% hObject    handle to unit_list (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in show_position.
function show_position_Callback(hObject, eventdata, handles)
% hObject    handle to show_position (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of show_position
handles.show_posvel = get(hObject,'Value');
guidata(hObject,handles);


% --- Executes on button press in show_attitude.
function show_attitude_Callback(hObject, eventdata, handles)
% hObject    handle to show_attitude (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of show_attitude
handles.show_att = get(hObject,'Value');
guidata(hObject,handles);


% --- Executes on selection change in angles_list.
function angles_list_Callback(hObject, eventdata, handles)
% hObject    handle to angles_list (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns angles_list contents as cell array
%        contents{get(hObject,'Value')} returns selected item from angles_list


% --- Executes during object creation, after setting all properties.
function angles_list_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angles_list (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in axis_change.
function axis_change_Callback(hObject, eventdata, handles)
% hObject    handle to axis_change (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
plot_sweepgraf2(handles.ROS,handles.PAR1,handles.PAR2,handles);


% --- Executes on button press in auto_fit_check.
function auto_fit_check_Callback(hObject, eventdata, handles)
% hObject    handle to auto_fit_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of auto_fit_check
handles.fit = get(hObject,'Value');
guidata(hObject,handles);


% --- Executes on button press in enlarge_plots.
function enlarge_plots_Callback(hObject, eventdata, handles)
% hObject    handle to enlarge_plots (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.enl_plots = 1;
guidata(hObject,handles);
plot_sweepgraf2(handles.ROS,handles.PAR1,handles.PAR2,handles);
handles.enl_plots = 0;
guidata(hObject,handles);


% --- Executes on button press in cont_fit.
function cont_fit_Callback(hObject, eventdata, handles)
% hObject    handle to cont_fit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.output_text);
cla;
text(0.0,1.0,['Performing continous fit...'],'FontSize',14,'Color','green');
pause(0.1);

%Start at time given by user
handles.fb               = 0;
handles.where1           = handles.start;
handles.where2           = 0;

where = handles.start;
mate  = handles.end_of_data;

dataP1                   = [];
dataP2                   = [];
while where <=  mate     %Until user specified end
   [handles.ROS,handles.PAR1,handles.PAR2,handles.events_loaded,...
        handles.RVA_data] = get_datagraf(handles.ROS,handles.PAR1,...
        handles.PAR2,hObject,eventdata,handles);
    handles.where1 = handles.ROS.where1;
    handles.where2 = handles.ROS.where2;
    if (handles.show_probe1 == 1)
        handles.where1 = handles.where1+1;
        where = datenum(handles.ROS.TIME1(1,1:6));
    else
```

```
        handles.where2 = handles.where2+1;
        where = datenum(handles.ROS.TIME2(1,1:6));
    end

    if (handles.show_probe1 == 1)
        save_data(handles,handles.ROS,handles.PAR1,1);
    end
    if (handles.show_probe2 == 1)
        save_data(handles,handles.ROS,handles.PAR2,2);
    end
end

function editor2_Callback(hObject, eventdata, handles)
% hObject    handle to editor2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editor2 as text
%        str2double(get(hObject,'String')) returns contents of editor2 as a double

handles.end_of_data = get(hObject,'String');
mate                = str2num(handles.end_of_data);;
stl                 = size(mate,2);
if (stl < 1)
    return;
end
if (stl == 1)
    mate(2:3) = 0;
end
if (stl == 2)
    mate(3) = 0;
end
if (stl == 4)
    mate(5:6) = 0;
end
if (stl == 5)
    mate(6) = 0;
end
handles.end_of_data = datenum(mate);
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function editor2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editor2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in ion_therm.
function ion_therm_Callback(hObject, eventdata, handles)
% hObject    handle to ion_therm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ion_therm
handles.i_therm = get(hObject,'Value');
```

```
if (handles.i_therm == 1)
    set(handles.ion_ram,'Value',0);
    handles.i_ram = get(handles.ion_ram,'Value');

    %See so that allowed combinations are used
    if (handles.e_plasma == 0)
        set(handles.el_plasma,'Value',1);
        handles.e_plasma = get(handles.el_plasma,'Value');
        axes(handles.output_text);
        cla;
        text(0.0,1.0,['Sorry, combination not supported, use plasma electrons'],...
            'FontSize',14,'Color','green');
        pause(3)
        cla;
    end
end
guidata(hObject,handles);

% --- Executes on button press in ion_ram.
function ion_ram_Callback(hObject, eventdata, handles)
% hObject    handle to ion_ram (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ion_ram
handles.i_ram = get(hObject,'Value');
if (handles.i_ram == 1)
    set(handles.ion_therm,'Value',0);
    handles.i_therm = get(handles.ion_therm,'Value');

    %See so that allowed combinations are used
    if (handles.e_plasma == 0)
         set(handles.el_plasma,'Value',1);
         handles.e_plasma = get(handles.el_plasma,'Value');
         axes(handles.output_text);
         cla;
         text(0.0,1.0,['Sorry, combination not supported, use plasma electrons'],...
             'FontSize',14,'Color','green');
         pause(3)
         cla;
    end
end
guidata(hObject,handles);


% --- Executes on button press in el_sc.
function el_sc_Callback(hObject, eventdata, handles)
% hObject    handle to el_sc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of el_sc
handles.e_sc = get(hObject,'Value');
guidata(hObject,handles);

%See so that allowed combinations are used
if (handles.e_plasma == 0)
    set(handles.el_plasma,'Value',1);
    handles.e_plasma = get(handles.el_plasma,'Value');
    axes(handles.output_text);
    cla;
    text(0.0,1.0,['Sorry, combination not supported, use plasma electrons'],...
        'FontSize',14,'Color','green');
    pause(3)
    cla;
end
guidata(hObject,handles);

% --- Executes on button press in el_photo.
function el_photo_Callback(hObject, eventdata, handles)
% hObject    handle to el_photo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of el_photo
handles.e_photo = get(hObject,'Value');
if (handles.e_plasma == 0)
    set(handles.el_plasma,'Value',1);
    handles.e_plasma = get(handles.el_plasma,'Value');
    axes(handles.output_text);
    cla;
    text(0.0,1.0,['Sorry, combination not supported, use plasma electrons'],...
        'FontSize',14,'Color','green');
    pause(3)
    cla;
end
guidata(hObject,handles);

% --- Executes on button press in el_plasma.
function el_plasma_Callback(hObject, eventdata, handles)
% hObject    handle to el_plasma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of el_plasma
handles.e_plasma = get(hObject,'Value');
if (handles.e_plasma == 0)
    set(handles.el_plasma,'Value',1);
    handles.e_plasma = get(handles.el_plasma,'Value');
    axes(handles.output_text);
    cla;
    text(0.0,1.0,['Sorry, combination not supported, use plasma electrons'],...
        'FontSize',14,'Color','green');
    pause(3)
    cla;
end
guidata(hObject,handles);


% --- Executes on button press in fit_all_data.
function fit_all_data_Callback(hObject, eventdata, handles)
% hObject    handle to fit_all_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of fit_all_data
handles.fit_all = get(hObject,'Value');
if (handles.fit_all == 1)
    set(handles.fit_partial_data,'Value',0);
    handles.fit_partial = get(handles.fit_partial_data,'Value');
end
guidata(hObject,handles);

% --- Executes on button press in fit_partial_data.
```

```
function fit_partial_data_Callback(hObject, eventdata, handles)
% hObject    handle to fit_partial_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of fit_partial_data
handles.fit_partial = get(hObject,'Value');
if (handles.fit_partial == 1)
    set(handles.fit_all_data,'Value',0);
    handles.fit_all = get(handles.fit_all_data,'Value');
end
guidata(hObject,handles);


% --- Executes on button press in fit_quasi.
function fit_quasi_Callback(hObject, eventdata, handles)
% hObject    handle to fit_quasi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of fit_quasi
handles.fit_quasineutral = get(hObject,'Value');
if (handles.fit_quasineutral == 1)
    set(handles.fit_free_dens,'Value',0);
    handles.fit_free_densities = get(handles.fit_free_dens,'Value');
end

guidata(hObject,handles);

% --- Executes on button press in fit_free_dens.
function fit_free_dens_Callback(hObject, eventdata, handles)
% hObject    handle to fit_free_dens (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of fit_free_dens
handles.fit_free_densities = get(hObject,'Value');
if (handles.fit_free_densities == 1)
    set(handles.fit_quasi,'Value',0);
    handles.fit_quasineutral = get(handles.fit_quasi,'Value');
end
guidata(hObject,handles);


% --- Executes on button press in show_P1.
function show_P1_Callback(hObject, eventdata, handles)
% hObject    handle to show_P1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of show_P1
handles.show_probe1 = get(hObject,'Value');
guidata(hObject,handles);

% --- Executes on button press in show_P2.
function show_P2_Callback(hObject, eventdata, handles)
% hObject    handle to show_P2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of show_P2
handles.show_probe2 = get(hObject,'Value');
```

```
guidata(hObject,handles);

% --- Executes on button press in fit_tryall.
function fit_tryall_Callback(hObject, eventdata, handles)
% hObject    handle to fit_tryall (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of fit_tryall
handles.fit_try_all = get(hObject,'Value');
guidata(hObject,handles);


% --- Executes on button press in button_redofit.
function button_redofit_Callback(hObject, eventdata, handles)
% hObject    handle to button_redofit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

axes(handles.output_text);
cla;
text(0.0,1.0,['Work in progress...'],'FontSize',14,'Color','green');
pause(0.1);
handles.fb               = 0;
[handles.ROS,handles.PAR1,handles.PAR2,handles.events_loaded,...
    handles.RVA_data] = get_datagraf(handles.ROS,handles.PAR1,...
    handles.PAR2,hObject,eventdata,handles);
handles.where1           = handles.ROS.where1;
handles.where2           = handles.ROS.where2;
guidata(hObject, handles);

plot_sweepgraf2(handles.ROS, handles.PAR1,handles.PAR2, handles);
displayinfo(handles.ROS,handles.PAR1,handles.PAR2,handles);


% --- Executes on button press in debug_mode.
function debug_mode_Callback(hObject, eventdata, handles)
% hObject    handle to debug_mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of debug_mode
handles.debug = get(hObject,'Value');
guidata(hObject,handles);
```

B.1.2 Rosetta_info.m

```
%Rosetta_Info.m, A structure for the Rosetta mission spacecraft
function ROS = Rosetta_Info()

ROS = [];
ROS.DB          = 0;                        %Not connected to database
ROS.Database    = 'squid.irfu.se:24';       %ISDAT Host server
ROS.PROJ        = 'Rosetta';                %Project name
ROS.MEM         = 'RPC';                    %Member name
ROS.INST        = 'LAP_Calibrated';         %Instrument name
ROS.SIG         = '';                       %Signal name, not given
ROS.SEN1        = 'P1';                     %Sensor name, P1
ROS.SEN2        = 'P2';                     %Sensor name, P2
ROS.CHAN1       = '';                          %Channel name 1, not given
ROS.CHAN2       = '';                          %Channel name 2, not given
ROS.PARAM       = '';                          %Parameter name, not given

%For save-information
ROS.DAT_LENGTH1 = [];                          %Nr. of points in the sweep
ROS.DAT_LENGTH2 = [];
ROS.DUR1        = [];                          %Duration of sweep
ROS.DUR2        = [];
ROS.MIN_Vb1     = [];                          %Min bias for probe 1
ROS.MIN_Vb2     = [];
ROS.MAX_Vb1     = [];                          %Max bias for probe 1
ROS.MAX_Vb2     = [];
ROS.STEP_SIZE1  = [];                          %Step size probe 1
ROS.STEP_SIZE2  = [];
ROS.KIND1       = [];                          %What kind of sweep probe 1?
                                               %Up/down, down, up, down/up
ROS.KIND2       = [];
ROS.DAT_CONT1   = [];                          %No Data Contents yet
ROS.DAT_DUR1    = [];                          %Has to do with data contents above
ROS.DAT_CONT2   = [];                          %No Data Contents yet
ROS.DAT_DUR2    = [];
ROS.I_DAT1      = [];                          %Sweep bias data will be saved here%
ROS.V_DAT1      = [];                          %And here too
ROS.TIME1       = [];                          %Sweep time will be saved here
ROS.I_DAT2      = [];                          %Sweep bias data will be saved here%
ROS.V_DAT2      = [];                          %And here too
ROS.TIME2       = [];                          %Sweep time will be saved here

ROS.where       = 0;
ROS.PROBE_RAD   = 0.025;                       %The probe radius, [m]
ROS.PROBE_AREA  = 4*pi*(ROS.PROBE_RAD)^2;      %The probe area, [m^2]


disp([ROS.PROJ, ' spacecraft information structure set']);
```

B.2 Handling the data

B.2.1 get_datagraf.m

```
%GET_DATAGRAF provides the data from the ISDAT server, specifically for the
%            gui developed.
%
%   [ROS,PAR1,PAR2,events_loaded,RVA_data] = get_datagraf(ROS,PAR1,PAR2,...
%                                            hObject,eventdata,handles)
function [ROS,PAR1,PAR2,events_loaded,RVA_data] = ...
    get_datagraf(ROS,PAR1,PAR2, hObject,eventdata,handles)

ROS             = handles.ROS;
PAR1            = handles.PAR1;
PAR2            = handles.PAR2;
[ROS,PAR1,PAR2] = clearRP(handles);

events_loaded   = 0;
RVA_data        = 0;

nodat = 0;                          %Setting up some of
```

```
ROS.SIG     = 'Sweep';              %the information
ROS.CHAN1   = 'Ip';
ROS.CHAN2   = 'Vp';

%surpress warnings
warning off;
handles.where2

if (handles.where2 == 0)       %this is the trigger for looking at data
    time = handles.start;
    [jd1,start_P1] = search_dat(ROS,time,1);  %Look for first sweep
    [jd2,start_P2] = search_dat(ROS,time,2);
else
    start_P1 = handles.where1;
    start_P2 = handles.where2;
end

%Set some output now
ROS.where1 = start_P1;
ROS.where2 = start_P2;

%To continue the loop
dat1 = 0;
dat2 = 0;

if (handles.show_probe1 == 1)
    i = start_P1;
else
    i = start_P2;
end

if (handles.show_probe1 == 1)
    %Probe 1
    while dat1 == 0
        [t_1,I_1] = isGetDataLite(ROS.DB, ROS.DAT_CONT1(i,:), ROS.DAT_DUR1(i),...
            ROS.PROJ, ROS.MEM, ROS.INST, ROS.SIG, ROS.SEN1, ROS.CHAN1);
        [t_1,V_1] = isGetDataLite(ROS.DB, ROS.DAT_CONT1(i,:), ROS.DAT_DUR1(i),...
            ROS.PROJ, ROS.MEM, ROS.INST, ROS.SIG, ROS.SEN1, ROS.CHAN2);

        %It must be a bias sweep and there are strange data I don't understand
        %that I need to get rid of
        if length(V_1) ~= 0 && length(find(diff(V_1) ~= 0)) > 0

            %Probe 1
            t_1           = fromepoch(t_1);
            ROS.I_RAW1    = I_1;
            ROS.V_RAW1    = V_1;
            ROS.TIME_RAW1 = t_1;
            [V_1,I_1,t_1] = adjust_data(V_1,I_1,t_1);    %adjusting the data
            ROS.I_DAT1    = I_1;
            ROS.V_DAT1    = V_1;
            ROS.TIME1     = t_1;
            %Setting some information
            ROS.DUR1        = ROS.DAT_DUR1(i);
            ROS.DAT_LENGTH1 = length(V_1);
            ROS.MIN_Vb1     = min(V_1);
            ROS.MAX_Vb1     = max(V_1);
            test            = diff(V_1);
            if (test >= 0)
                kind = 1;
            else if (test <= 0)
                kind = 2;
                else if (RIS.V_RAW1(1) < 0)
                        kind = 3;               %down up and up down
                    else
                        kind = 4;
                    end
                end
            end
            ind             = (find(diff(V_1) ~= 0,15));
            ROS.STEP_SIZE1  = test(ind(15));          %Might be problems in beginning
            ROS.KIND1       = kind;                   %Upp?t svep just nu
            %Work on the data but only if model-box is checked
            ROS.where1  = i;
            [PAR1,events_loaded,RVA_data] = gothrough(V_1,I_1,PAR1,ROS,handles,1);
            dat1        = 1;
        else
            if handles.fb == 0
                i = i+1;
            else
                i = i-1;
            end
        end
    end
end

%Probe 2
if (handles.show_probe2 == 1)
    while dat2 == 0
        [t_2,I_2] = isGetDataLite(ROS.DB, ROS.DAT_CONT2(i,:), ROS.DAT_DUR2(i),...
            ROS.PROJ, ROS.MEM, ROS.INST, ROS.SIG, ROS.SEN2, ROS.CHAN1);
        [t_2,V_2] = isGetDataLite(ROS.DB, ROS.DAT_CONT2(i,:), ROS.DAT_DUR2(i),...
            ROS.PROJ, ROS.MEM, ROS.INST, ROS.SIG, ROS.SEN2, ROS.CHAN2);

        %It must be a bias sweep and there are strange data I don't understand
        %that I need to get rid of
        if length(V_2) ~= 0 && length(find(diff(V_2) ~= 0)) > 0

            %Probe 1
            t_2          = fromepoch(t_2);
            ROS.I_RAW2    = I_2;
            ROS.V_RAW2    = V_2;
            ROS.TIME_RAW2 = t_2;
            [V_2,I_2,t_2] = adjust_data(V_2,I_2,t_2);   %adjusting the data
            ROS.I_DAT2    = I_2;
            ROS.V_DAT2    = V_2;
            ROS.TIME2     = t_2;

            %Setting some information
            ROS.DUR2        = ROS.DAT_DUR2(i);
            ROS.DAT_LENGTH2 = length(V_2);
            ROS.MIN_Vb2     = min(V_2);
            ROS.MAX_Vb2     = max(V_2);
            test            = diff(V_2);
            if (test >= 0)
                kind = 1;
            else if (test <= 0)
                kind = 2;
                else if (ROS.V_RAW2(1) < 0)
                        kind = 3;               %down up and up down
                    else
                        kind = 4;
                    end
                end
            end
            ind             = (find(diff(V_2) ~= 0,15));
            ROS.STEP_SIZE2  = test(ind(15));

            ROS.KIND2       = kind;                             %Upp?t svep just nu

            %Need to get rid of zero-contributions
            %ROS.I_DAT
            %indz = find(ROS.I_DAT == 0);
            %ROS.I_DAT(indz)  = NaN;
            %ROS.V_DAT(indz)  = NaN;
            %ROS.TIME(indz,:) = NaN;

            %Work on the data but only if model-box is checked
            ROS.where2 = i;
            dat2       = 1;
            [PAR2,events_loaded,RVA_data] = gothrough(V_2,I_2,PAR2,ROS,handles,2);
        else
            if (handles.fb == 1 && handles.show_probe1 == 0)
                i = i-1;
            else
                i = i+1;
            end
        end
    end
end
```

B.2.2 adjust_data.m

```
%adjust_data.m, This program cuts the data collected and sorts it
%in ascending order
%Input: Bias voltage (V), Current (I) and time (t)
%Output: Vc (the cut&sort voltage), Ic (the cut&sort current) and tc (the cut&sort time)

function [Va,Ia,ta] = adjust_data(V,I,t)

cut = 10;



Ic = I(find(V > cut,1):length(V));       %Cut the data
Vc = V(find(V > cut,1):length(V));       %everything under 5 V in start of
tc = t(find(V > cut,1):length(V),:);     %the data set is cut

[Vs,ind] = sort(Vc);                   %Sort the data in ascending order
Is = Ic(ind);                          %We get an upward sweep
ts = tc(ind,:);

%Sort out the same V-measurements (4 every time) and take mean in current

ind = find(diff(Vs) ~= 0);

if length(ind) < 1
    disp('Could not adjust data, data stored in raw form')
    Ia = Is;
    Va = Vs;
    ta = ts;
    return
end
```

```
Va = Vs(ind);                                       %a is for adjust
k = 1;
for i = 1:length(ind)
    Ia(i) = mean(Is(k:ind(i)));
    k = ind(i)+1;
end
ta = ts(ind,:);

%Look if the current is saturated,if yes remove saturated part
ind = find(Ia < 9.60*1e-6); %9.8 from the start
Ia  = Ia(ind)';
Va  = Va(ind);
ta  = ta(ind,:);
```

B.2.3 remove_dat.m

```
%remove_dat.m
%Lets the user remove the data that just messes up the fit

function [ROS,PAR] = remove_dat(handles)

ROS = handles.ROS;
PAR = handles.PAR;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Set up time and other information%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Vb      = ROS.V_DAT;
Ib      = ROS.I_DAT;
logI    = PAR.I_log_abs;
dat_num = ROS.where;
t       = ROS.TIME(1,:);
tim     = ROS.TIME(:,:);
time    = datestr(t);
time    = datestr(time,31);
Vsc     = PAR.Vsc;

again = 0;                      %again = 0 means to do the fit
                                %again = 1 means to stop fitting

axes(handles.output_text);
cla;
text(0.0,1.0,['Pick a point on the log-curve (lower right curve)'],...
    'FontSize',12,'Color','green');
text(0.0,0.5,['No data-points to the right will be used in the fit.'],...
    'FontSize',12,'Color','green');

while again == 0
    axes(handles.coord3);           %Log-curve in coord3
    cla;

    plot(Vb,logI,'k.');
    title(['Time: ', time,',   Probe:', ROS.SEN,',   Data #: ',...
        num2str(dat_num)])
    xlabel('Vb [V]');
    ylabel('log(I)');
    grid on;
    %%%%%%%%%%%%%%%%%%%%
    %Find point on graph%
    %%%%%%%%%%%%%%%%%%%%
    [x,y] = ginput(1);
    point = x;
    if x > 0
        %%%%%%%%%%%%%%%%%%%%%
        %Find first datapoint%
        %%%%%%%%%%%%%%%%%%%%%
        ind1 = find(Vb >= x,1);
        %%%%%%%%%%%
        %Plot line%
        %%%%%%%%%%%
        len     = 1000;                    %How many points
        point_x = linspace(x,x,len);
        point_y = linspace(min(logI),max(logI),len);
        hold on;
        plot(point_x,point_y,'red');

        axes(handles.output_text);
        cla;
        text(0.0,1.0,['Is the interval satisfactory? Yes = rmb, No = lmb'],...
            'Color','green','FontSize',12);
        [X,Y,button] = ginput(1);
        if button == 2 || button == 3
            again = 1;
        end
    else
        axes(handles.output_text);
        cla;
        text(0.0,1.0,['The point must be to the right of Vb = 0'],...
            'Color','green','FontSize',12);
    end
end
%%%%%%%%%%%%
%Set outdata%
%%%%%%%%%%%%
axes(handles.output_text);
cla;
ROS.I_DAT = [];
ROS.V_DAT = [];
ROS.TIME  = [];
ROS.I_DAT = Ib(1:ind1);
ROS.V_DAT = Vb(1:ind1);
ROS.TIME  = tim(1:ind1,:);
```

B.2.4 search_dat.m

```
%SEARCH_DAT looks for data close to the specified date
%
%   tim = search_dat(ROS,time,probe)
%
%   ROS is needed to get the data
%   time is the time the user has specified (in JD)
%   probe is what probe is used (1 or 2)
%
%   tim is the julian date of the index
%   ind is the index for the data set
function [tim,ind] = search_dat(ROS,time,probe)
```

```
if (probe == 1)
    data_time = datenum(ROS.DAT_CONT1(:,:));
else
    data_time = datenum(ROS.DAT_CONT2(:,:));
end

ind = find(data_time(:,1) >= time,1,'first');
tim = data_time(ind,1);                         %The JD
```

B.2.5 save_data.m

```
%SAVE_DATA saves the data in the file named []
%
%   save_data = save_data(handles,ROS,PAR,ref)

function save_data = save_data(handles,ROS,PAR,ref)
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Create matrices for saving%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%First of all determine the model used
if (handles.fit_try_all == 0)
    if (handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
        && handles.e_photo == 0)
        if (handles.fit_partial == 1)
            model = 1;
        else
            model = 11;
        end
    end
    if (handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
            && handles.e_photo == 1)
        if (handles.fit_partial == 1)
            model = 2;
        else
            model = 21;
        end
    end
    if (handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
            && handles.e_photo == 0)
        if (handles.fit_partial == 1)
            model = 3;
        else
            model = 31;
        end
    end
    if (handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 0 ...
            && handles.e_photo == 1)
        if (handles.fit_partial == 1)
            model = 4;
        else
            model = 41;
        end
    end
    if (handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
            && handles.e_photo == 0)
        if (handles.fit_partial == 1)
            model = 51;
        else
```

```
            model = 5;
        end
    end
    if (handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
            && handles.e_photo == 1)
        if (handles.fit_partial == 1)
            model = 61;
        else
            model = 6;
        end
    end
    if (handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 0 ...
            && handles.e_photo == 0)
        if (handles.fit_partial == 1)
            model = 71;
        else
            model = 7;
        end
    end
      if (handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 0 ...
            && handles.e_photo == 1)
        if (handles.fit_partial == 1)
            model = 81;
        else
            model = 8;
        end
      end
else
    model = PAR.parameters(15);
end

if ref == 1
    len   = size(ROS.TIME1,2);
    time  = ROS.TIME1(1,1:6);            %Setting time
    ltime = ROS.TIME1(len,6)-ROS.TIME1(1,6);
    cs    = PAR.CS;                      %Coordinate system
    switch cs
        case 'GSE'
            cs = 1;
        case 'GEI'
            cs = 2;
        case 'GEA'
            cs = 3;
        case 'HEI'
            cs = 4;
        case 'HEA'
            cs = 5;
    end
    un    = PAR.UNITS;                   %Units used
    switch un
        case 'km'
            un = 1e3;           %km
        case 'AU'
            un = 1e11;          %Au
        case 'Re'
            un = 6371.2*1e3;    %Re
        case 'Rm'
            un = 3397*1e3;      %Rm
    end
    x  = PAR.R(1)*un;
    y  = PAR.R(2)*un;
    z  = PAR.R(3)*un;
    vx = PAR.V(1)*un;
    vy = PAR.V(2)*un;
    vz = PAR.V(3)*un;
    phi      = PAR.Sphi;
    theta    = PAR.Stheta;
    rho      = PAR.Srho;
    vphi     = PAR.Vphi;
    vtheta   = PAR.Vtheta;
    vrho     = PAR.Vrho;
    probe    = ref;
    kind     = ROS.KIND1;
    minVb    = ROS.MIN_Vb1;
    maxVb    = ROS.MAX_Vb1;
    ssize    = ROS.STEP_SIZE1;
    nrpoints = ROS.DAT_LENGTH1;
else
    len   = size(ROS.TIME2,2);
    time  = ROS.TIME2(1,1:6);            %Setting time
    ltime = ROS.TIME2(len,6)-ROS.TIME2(1,6);
    cs    = PAR.CS;
    switch cs
        case 'GSE'
            cs = 1;
        case 'GEI'
            cs = 2;
        case 'GEA'
            cs = 3;
        case 'HEI'
            cs = 4;
        case 'HEA'
            cs = 5;
    end
    un    = PAR.UNITS;                   %Units used
    switch un
        case 'km'
            un = 1e3;           %km
        case 'AU'
            un = 1e11;          %Au
        case 'Re'
            un = 6371.2*1e3;    %Re
        case 'Rm'
            un = 3397*1e3;      %Rm
    end
    x  = PAR.R(1)*un;
    y  = PAR.R(2)*un;
    z  = PAR.R(3)*un;
    vx = PAR.V(1)*un;
    vy = PAR.V(2)*un;
    vz = PAR.V(3)*un;
    phi      = PAR.Sphi;
    theta    = PAR.Stheta;
    rho      = PAR.Srho;
    vphi     = PAR.Vphi;
    vtheta   = PAR.Vtheta;
    vrho     = PAR.Vrho;
    probe    = ref;
    kind     = ROS.KIND2;
    minVb    = ROS.MIN_Vb2;
    maxVb    = ROS.MAX_Vb2;
    ssize    = ROS.STEP_SIZE2;
    nrpoints = ROS.DAT_LENGTH2;
end
```

```
%Create the matrix and save in it
Sav1(1,1:6)   = time;
Sav1(1,7)     = cs;
Sav1(1,8:10)  = [x y z];
Sav1(1,11:13) = [vx vy vz];
if (length([phi theta rho vphi vtheta vrho]) == 0)
    Sav1(1,14:19) = 0;
else
    Sav1(1,14:19) = [phi theta rho vphi vtheta vrho];
end
Sav1(1,20)    = probe;
Sav1(1,21)    = kind;
Sav1(1,22)    = minVb;
Sav1(1,23)    = maxVb;
Sav1(1,24)    = ssize;
Sav1(1,25)    = nrpoints;
Sav1(1,26)    = ltime;
%The fits also
if (length(PAR.parameters ~= 0));
    parameters = PAR.parameters;
    Sav1(1,27:36) = parameters(5:14);
    Sav1(1,37)    = model;
end
a = 38;
if (length(PAR.pars) ~= 0)
    pars = PAR.pars;
    for i = 1:9
        Sav1(1,a:a+10) = pars(i,5:15);  %Parameters and model choice
        Sav1(1,a+11)   = NaN;           %To separate models
        a = a+12;
    end
else
    Sav1(1,38:145) = 0;                 %To get same size
end

%Get file and dir, then save
cd save/               %Jumping to data directory
time         = datestr(time,30);
filename = ['Save_probe',num2str(probe)];
save(filename,'Sav1','-ASCII','-APPEND');

%Give output text
axes(handles.output_text);
cla;
text(0.0,0.5,['Data saved'], 'Color', 'green','FontSize',16);
pause(0.5)
cla;

%Move back to original directory
cd ..
```

B.3 Setting the position and velocity

B.3.1 set_r_v.m

```
%SET_R_V provides the position, speed, velocity and attitude of Rosetta
```

```
%       for the chosen coordinate system
%
%   [par_out,events_loaded,RVA_data] = set_r_v(time,handles)
%
%   Some lines taken from Magnus Billvik
function [par_out,events_loaded,RVA_data] = set_r_v(time,handles)


par_out       = {};
att           = handles.show_att;
pos           = handles.show_posvel;
fit           = handles.fit;
inc           = 23.439291*pi/180;         %Earths inclination in radians
system        = handles.system_choice;
unit          = handles.unit_choice;
events_loaded = handles.events_loaded;
RVA_data      = handles.RVA_data;
co_syst       = 'HEI';                    %Default coordinate system
un            = 'km';                     %Default units used

%First get the raw data
[rr,vr,rs_gea,re_hea,ve_hea,Asc_gei,events_loaded,event,RVA_data] = ...
    get_traj(time,inc,att,events_loaded,RVA_data);

%Do some transforms
if re_hea ~= 0
    re_hei = gea2gei(re_hea,inc);
    ve_hei = gea2gei(ve_hea,inc);
end

%check event
if event <= 4
    rr_gei = rr;
    vr_gei = vr;
else
    rr_hei = rr;
    vr_hei = vr;
    rr_hea = gei2gea(rr_hei,inc);
    vr_hea = gei2gea(vr_hei,inc);
    rr_gei = hei2gei(rr_hei,re_hei);
    vr_gei = hei2gei(vr_hei,ve_hei);
end

%Now transform to desired coordinate system
switch system
    case 1  %gse
        rr_gea  = gei2gea(rr_gei,inc);
        vr_gea  = gei2gea(vr_gei,inc);
        rr_gse  = gea2gse(rr_gea,rs_gea);
        vr_gse  = gea2gse(vr_gea,rs_gea);
        %abs_rr_gse = sqrt(rr_gse(1)^2+rr_gse(2)^2+rr_gse(3)^2)
        rr      = rr_gse;
        vr      = vr_gse;
        co_syst = 'GSE';
    case 2 %gei
        rr      = rr_gei;
        vr      = vr_gei;
        co_syst = 'GEI';
    case 3 %gea
        rr_gea  = gei2gea(rr_gei,inc);
        vr_gea  = gei2gea(vr_gei,inc);
        rr      = rr_gea;
        vr      = vr_gea;
        co_syst = 'GEA';
    case 4 %hei
        rr      = rr_hei;
        vr      = vr_hei;
        co_syst = 'HEI';
    case 5 %hea
        rr_hea  = gei2gea(rr_hei,inc);
        vr_hea  = gei2gea(vr_hei,inc);
        rr      = rr_hea;
        vr      = vr_hea;
        co_syst = 'HEA';
end

%Set attitude
if att == 1
    number = size(rr,2);
    Asc_gea(1,:) = gei2gea_a(Asc_gei(1,:),inc);
    Asc_gea(2,:) = gei2gea_a(Asc_gei(2,:),inc);
    Asc_gea(3,:) = gei2gea_a(Asc_gei(3,:),inc);
    Asc_gse(1,:) = gea2gse_a(Asc_gea(1,:),rs_gea);
    Asc_gse(2,:) = gea2gse_a(Asc_gea(2,:),rs_gea);
    Asc_gse(3,:) = gea2gse_a(Asc_gea(3,:),rs_gea);
    Asc          = Asc_gse;
    % HEA(J2000) to ROS
    rs_ros=zeros(3,number);

    if event <= 4
        % translation and rotation to rosetta's ref. frame
        % by def. Asc_gea(:,:,i) is the rotation matrix from GEA to ROS
        for i=1:number
            vr_ros(:,i)=Asc_gea(:,:,i)*vr_gea(:,i);
            rs_ros(:,i)=Asc_gea(:,:,i)*(rs_gea(:,i)-rr_gea(:,i));  % sun's position
        end
    else
        for i=1:number
            vr_ros(:,i)=Asc_gea(:,:,i)*vr_hea(:,i);
            rs_ros(:,i)=Asc_gea(:,:,i)*(-rr_hea(:,i));
        end
    end
    % ROS(x,y,z) to ROS(rho, phi, theta in km and degrees resp.)
    disp('- ros (xyz) -> ros (rho,phi,theta)');
    sphi=zeros(1,number); % declare the variables to speed up the loop!
    stheta=zeros(1,number);
    srho=zeros(1,number);
    vphi=zeros(1,number);
    vtheta=zeros(1,number);
    vrho=zeros(1,number);
    for i=1:number
        [sphi(i), stheta(i), srho(i)]=cart2sphere(rs_ros(1,i), rs_ros(2,i), rs_ros(3,i));
        [vphi(i), vtheta(i), vrho(i)]=cart2sphere(vr_ros(1,i), vr_ros(2,i), vr_ros(3,i));
    end
    sphi=sphi*180/pi;        %elevation
    vphi=vphi*180/pi;
    stheta=stheta*180/pi;    %azimuth
    vtheta=vtheta*180/pi;
    srho;
    vrho;
end

%Set the units used
switch unit
```

```
    case 1  %Re
        rr = rr./(6371.2);        %Re = 6371.2 km
        vr = vr./(6371.2);
        un = 'Re';
    case 2  %Rm
        rr = rr./(3397);          %Rm = 3397 km
        vr = vr./(3397);
        un = 'Rm';
    case 3  %Au
        rr = rr./(1.495978*1e8); %1 AU = 1.495978*1e11 m
        vr = vr./(1.495978*1e8);
        un = 'Au';
end         %km = km

%Set output
if (pos == 1 || fit == 1)
    par_out{1} = rr;
    par_out{2} = vr;
end
if att == 1
    par_out{3}   = sphi;
    par_out{4}   = vphi;
    par_out{5}   = stheta;
    par_out{6}   = vtheta;
    par_out{7}   = srho;
    par_out{8}   = vrho;
end
par_out{9}  = co_syst;    %system and unit
par_out{10} = un;
```

B.3.2 get_traj.m

```
%GET_TRAJ provides the HEI coordinates of Rosetta for the given time, taken
%         from the raw trajectory data file. (HEI = heliocentric equatorial
%         inertial)
%
%   [rr,vr,rs_gea,re_hea,ve_hea,Asc_gei,loaded,event,RVA_data] = ...
%    get_traj(time,inc,att,loaded,RVA_data)
function [rr,vr,rs_gea,re_hea,ve_hea,Asc_gei,loaded,event,RVA_data] = ...
    get_traj(time,inc,att,loaded,RVA_data)

timediff  = datenum([-4713 1 1 12 0 0])+327;
startdate = [2000 1 1 0 0 0]';
enddate   = [2020 1 1 0 0 0]';
event     = 5;
Asc_gei   = 0;      %just in case no attitude
rs_gea    = 0;
re_hea    = 0;
ve_hea    = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Look to see if the time is inside a special event%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (datenum(time) >= 7.323734006942118e+05 && datenum(time) <= 7.323784587248912e+05)
    disp('Earth flyby #1, 2005')
    event     = 1;
    if loaded(1,1) == 0
        loaded(1,1) = 1;
        [RR, VR, dater]  = readros('traj_r_e1.txt', startdate, enddate); %RR in GEI
        [RS, dates, jds] = readjpltraj('traj_s_e1.txt');                %RS in GEI
        RVA_data.RR1     = RR;
        RVA_data.VR1     = VR;
        RVA_data.dater1  = dater;
        RVA_data.RS1     = RS;
        RVA_data.dates1  = dates;
        RVA_data.jds1    = jds;
    else
        disp('Data already loaded')
        RR    = RVA_data.RR1;
        VR    = RVA_data.VR1;
        dater = RVA_data.dater1;
        RS    = RVA_data.RS1;
        dates = RVA_data.dates1;
        jds   = RVA_data.jds1;
    end
    if att == 1
        if loaded(1,2) == 0
            disp('Getting attitude')
            loaded(1,2)     = 1;            %shows that attitude is also loaded
            [Q, datea]      = readatt('attitude.txt', dater(:,1), dater(:,length(dater)));
            RVA_data.Q1     = Q;
            RVA_data.datea1 = datea;
        else if loaded(1,2) == 1
                disp('Attitude already loaded')
                Q     = RVA_data.Q1;
                datea = RVA_data.datea1;
            end
        end
    end
end
if (datenum(time) >= 7.333587520354936e+05 && datenum(time) <= 7.333609954703673e+05)
    disp('Earth flyby #2, 2007')
    event     = 2;
    if loaded(2,1) == 0
        loaded(2,1) = 1;
        [RR, VR, dater] = readros('traj_r_e2.txt', startdate, enddate);
        [RS, dates, jds] = readjpltraj('traj_s_e2.txt');
        RVA_data.RR2     = RR;
        RVA_data.VR2     = VR;
        RVA_data.dater2  = dater;
        RVA_data.RS2     = RS;
        RVA_data.dates2  = dates;
        RVA_data.jds2    = jds;
        %[RMo, datem, jdmo] = readjpltraj('traj_m_e2.txt');
    else
        disp('Data already loaded')
        RR    = RVA_data.RR2;
        VR    = RVA_data.VR2;
        dater = RVA_data.dater2;
        RS    = RVA_data.RS2;
        dates = RVA_data.dates2;
        jds   = RVA_data.jds2;
    end
    if att == 1
        if loaded(2,2) == 0
            loaded(2,2) = 1;
            disp('Getting attitude')
            [Q, datea] = readatt('attitude.txt', dater(:,1), dater(:,length(dater)));
            RVA_data.Q2     = Q;
            RVA_data.datea2 = datea;
        else if loaded(2,1) == 1
                disp('Attitude already loaded')
                Q     = RVA_data.Q2;
                datea = RVA_data.datea2;
            end
        end
    end
end
if (datenum(time) >= 7.340891972105330e+05 && datenum(time) <= 7.340914353604476e+05)
    disp('Earth flyby #3, 2009')
    event = 3;
    if loaded(3,1) == 0
        loaded(3,1) = 1;
        [RR, VR, dater] = readros('traj_r_e3.txt', startdate, enddate);
        [RS, dates, jds] = readjpltraj('traj_s_e3.txt');
        RVA_data.RR3     = RR;
        RVA_data.VR3     = VR;
        RVA_data.dater3  = dater;
        RVA_data.RS3     = RS;
        RVA_data.dates3  = dates;
        RVA_data.jds3    = jds;
        %[RMo, datem, jdmo] = readjpltraj('traj_m_e2.txt');
    else
        disp('Data already loaded')
        RR    = RVA_data.RR3;
        VR    = RVA_data.VR3;
        dater = RVA_data.dater3;
        RS    = RVA_data.RS3;
        dates = RVA_data.dates3;
        jds   = RVA_data.jds3;
    end
    if att == 1
        if loaded(3,2) == 0
            loaded(3,2) = 1;
            disp('Getting attitude')
            [Q, datea] = readatt('attitude.txt', dater(:,1), dater(:,length(dater)));
            RVA_data.Q3     = Q;
            RVA_data.datea3 = datea;
        else if loaded(3,2) == 1
                disp('Attitude already loaded')
                Q     = RVA_data.Q3;
                datea = RVA_data.datea3;
            end
        end
    end
end
if (datenum(time) >= 7.330973250541261e+05 && datenum(time) <= 7.330988350641347e+05)
    disp('Mars flyby #1, 2007')
    event = 4;
    if loaded(4,1) == 0
        loaded(4,1) = 1;
        [RR, VR, dater] = readros('traj_r_m20051019.txt', startdate, enddate);
        [RS, dates, jds] = readjpltraj('suntest.txt');
        RVA_data.RR4     = RR;
        RVA_data.VR4     = VR;
        RVA_data.dater4  = dater;
        RVA_data.RS4     = RS;
        RVA_data.dates4  = dates;
        RVA_data.jds4    = jds;
        %%% H?R SKALL KANSKE PHOBOS OCH DEIMOS MED SEN
        %[RMo, datem, jdmo] = readjpltraj('traj_m_e2.txt');
    else
        disp('Data already loaded')
        RR    = RVA_data.RR4;
        VR    = RVA_data.VR4;
        dater = RVA_data.dater4;
        RS    = RVA_data.RS4;
        dates = RVA_data.dates4;
        jds   = RVA_data.jds4;
    end
    if att == 1
        if loaded(4,2) == 0
            loaded(4,2) = 1;
                disp('Getting attitude')
                [Q, datea] = readatt('attitude.txt', dater(:,1), dater(:,length(dater)));
                RVA_data.Q4     = Q;
                RVA_data.datea4 = datea;
        else if loaded(4,2) == 1
                    disp('Attitude already loaded')
                    Q     = RVA_data.Q4;
                    datea = RVA_data.datea4;
            end
        end
    end
end
if event == 5;

    disp('Whole mission, THE NUMBERS GIVEN CAN NOW BE WRONG ~ 1500 km!')
        if (loaded(5,1) == 0 || loaded(5,3) == 0)
            if (datenum(time) >= 2453279.501388889+timediff && datenum(time) <= 2453310.498611111+timediff)
                if loaded(5,1) == 0
                    [RR, VR, dater] = readros('traj_r_whole.txt', startdate, enddate);
                    loaded(5,1)     = 1;
                else
                    RR    = RVA_data.RR5;
                    VR    = RVA_data.VR5;
                    dater = RVA_data.dater5;
                end
                disp('Probably LAP dance, loading october data')
                [RE, datee, jde]   = readjpltraj('traj_dat/traj_e_OCT_2004.txt');
                [VE, deteve, jdve] = readjpltraj('traj_dat/vel_e_OCT_2004.txt');
                [RS, dates, jds]   = readjpltraj('traj_dat/traj_s_OCT_2004.txt');
                loaded(5,3)      = 1;
                RVA_data.RR5     = RR;
                RVA_data.VR5     = VR;
                RVA_data.dater5  = dater;
                RVA_data.RS5     = RS;
                RVA_data.dates5  = dates;
                RVA_data.jds5    = jds;
                RVA_data.RE5     = RE;
                RVA_data.jde5    = jde;
                RVA_data.VE5     = VE;
                RVA_data.jdve5   = jdve;
            else if loaded(5,1) == 0
                    loaded(5,1)        = 1;
                    [RR, VR, dater]    = readros('traj_r_whole.txt', startdate, enddate);
                    [RE, datee, jde]   = readjpltraj('traj_e_whole.txt');
                    [VE, deteve, jdve] = readjpltraj('vel_e_whole.txt');
                    [RS, dates, jds]   = readjpltraj('traj_s_whole.txt');
                    RVA_data.RR5     = RR;
                    RVA_data.VR5     = VR;
                    RVA_data.dater5  = dater;
                    RVA_data.RS5     = RS;
                    RVA_data.dates5  = dates;
                    RVA_data.jds5    = jds;
                    RVA_data.RE5     = RE;
                    RVA_data.jde5    = jde;
                    RVA_data.VE5     = VE;
                    RVA_data.jdve5   = jdve;
                else if (loaded(5,3) == 0)
                    disp('Data already loaded')
                    RR    = RVA_data.RR5;
                    VR    = RVA_data.VR5;
                    dater = RVA_data.dater5;
                    RS    = RVA_data.RS5;
                    dates = RVA_data.dates5;
                    jds   = RVA_data.jds5;
                    RE    = RVA_data.RE5;
                    jde   = RVA_data.jde5;
                    VE    = RVA_data.VE5;
                    jdve  = RVA_data.jdve5;
                    end
                end
            end
        else
            disp('Data already loaded')
            RR    = RVA_data.RR5;
            VR    = RVA_data.VR5;
            dater = RVA_data.dater5;
            RS    = RVA_data.RS5;
            dates = RVA_data.dates5;
            jds   = RVA_data.jds5;
            RE    = RVA_data.RE5;
            jde   = RVA_data.jde5;
            VE    = RVA_data.VE5;
            jdve  = RVA_data.jdve5;
        end
        if att == 1
            disp('Getting attitude')
            start = time - [0 0 1 0 0 0];
            stop  = time + [0 0 1 0 0 0];
            [Q, datea] = readatt('attitude.txt', start', stop');
        end
end
%%%%%%%%%%%%%%%%%%%%%%%%
%Convert to Julian Date%
%%%%%%%%%%%%%%%%%%%%%%%%
jdr                 = datenum(dater(1,:),dater(2,:),dater(3,:),...
                      dater(4,:),dater(5,:),dater(6,:));
jd                  = datenum(time);
jds                 = jds  + timediff;
if event == 5
    jde                 = jde  + timediff;
    jdve                = jdve + timediff;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Interpolation is needed to get the exact point%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if event <= 4
    rr_gei(1,:)     = interp1(jdr,RR(1,:),jd);
    rr_gei(2,:)     = interp1(jdr,RR(2,:),jd);
    rr_gei(3,:)     = interp1(jdr,RR(3,:),jd);
    vr_gei(1,:)     = interp1(jdr,VR(1,:),jd);
    vr_gei(2,:)     = interp1(jdr,VR(2,:),jd);
    vr_gei(3,:)     = interp1(jdr,VR(3,:),jd);
    rr              = rr_gei;                  %for output
    vr              = vr_gei;                  %this is just for output
    rs_gea(1,:)     = interp1(jds,RS(1,:),jd);
    rs_gea(2,:)      = interp1(jds,RS(2,:),jd);
    rs_gea(3,:)      = interp1(jds,RS(3,:),jd);
else
    rr_hei(1,:)      = interp1(jdr,RR(1,:),jd);
    rr_hei(2,:)      = interp1(jdr,RR(2,:),jd);
    rr_hei(3,:)      = interp1(jdr,RR(3,:),jd);
    vr_hei(1,:)      = interp1(jdr,VR(1,:),jd);
    vr_hei(2,:)      = interp1(jdr,VR(2,:),jd);
    vr_hei(3,:)      = interp1(jdr,VR(3,:),jd);
    rr               = rr_hei;                  %for output
    vr               = vr_hei;                  %this is just for output
    re_hea(1,:)      = interp1(jde,RE(1,:),jd);
    re_hea(2,:)      = interp1(jde,RE(2,:),jd);
    re_hea(3,:)      = interp1(jde,RE(3,:),jd);
    ve_hea(1,:)      = interp1(jde,VE(1,:),jd);
    ve_hea(2,:)      = interp1(jde,VE(2,:),jd);
    ve_hea(3,:)      = interp1(jde,VE(3,:),jd);
    rs_gea(1,:)      = interp1(jds,RS(1,:),jd);
    rs_gea(2,:)      = interp1(jds,RS(2,:),jd);
    rs_gea(3,:)      = interp1(jds,RS(3,:),jd);
end
if att == 1
    jda_tmp = datenum(datea(1,:),datea(2,:),datea(3,:),datea(4,:),datea(5,:),datea(6,:));
    Q_mod(:,1)=Q(:,1);
    jda(1)=jda_tmp(1);
    j=2;
    for i=2:length(jda_tmp)
        jda(j)=jda_tmp(i);
        Q_mod(:,j)=Q(:,i);
        if jda(j)==jda(j-1)
            j=j-1;
        end
        j=j+1;
    end
    clear jda_temp;
    %%%%%%%%%%%%%%%
    %Interpolate Q%
    %%%%%%%%%%%%%%%
    q1 = interp1(jda,Q_mod(1,:),jd);
    q2 = interp1(jda,Q_mod(2,:),jd);
    q3 = interp1(jda,Q_mod(3,:),jd);
    q4 = interp1(jda,Q_mod(4,:),jd);

    % Define rosettas attitude vectors (from the quaternions) in GEI frame of referece for all times (3d matrix),
    % the Rosetta inertial x,y,z axis being the rows of Asc
    % Formula for converting quaternions to attitude vectors are taken from the ESA document file:
    % RO-ESC-IF-5003_2_0_DDID_Appendix_H_Data_Delivery_FD_Products_2003Oct2
    % 3.pdf
    Asc_gei(1,1:3) = [q1^2-q2^2-q3^2+q4^2 2*(q1*q2+q3*q4) 2*(q1*q3-q2*q4)];
    Asc_gei(2,1:3) = [2*(q1*q2-q3*q4) -q1^2+q2^2-q3^2+q4^2 2*(q2*q3+q1*q4)];
    Asc_gei(3,1:3) = [2*(q1*q3+q2*q4) 2*(q2*q3-q1*q4) -q1^2-q2^2+q3^2+q4^2];

    jd=jda;                 % define the time from the attitude data
end
```

B.4 Fitting routines

B.4.1 extraction.m

```
%EXTRACTION determines the different physical parameters that fits the data
%           given. The parameters are returned in a field structure
%           containing the parameters obtained by various ways of fitting
%
%   [ppar,fit,currents,epop,fits,pars,parameters] = ...
%   extraction(Vb,I,handles,phi,probe)

function [ppar,fit,currents,epop,fits,pars,parameters] = ...
    extraction(Vb,I,handles,phi,probe)

ppar     = [];
fit      = [];
currents = [];
epop     = [];

lenV = length(Vb);
lenI = length(I);

if lenV == lenI
    %Start the fitting procedure
    [parameters,fit,pars,fits] = fitting2(Vb,I,handles,phi,probe);

    if (length(parameters) == 0 || length(fit) == 0)    %corrupt data
        ppar = zeros(16,0);
        fit  = zeros(2,1:length(Vb));
        currents = zeros(3,1:length(Vb));
        epop = 0;
        return;
    end

    %Get the physical parameters and the constituent currents
    V = linspace(min(Vb),max(Vb),1000);
    if (handles.fit_try_all == 1)
        if (parameters(15) == 5 || parameters(15) == 6 || ...
            parameters(15) == 7 || parameters(15) == 8)
            ppar                = par2phys2(parameters,handles,2);
            [Iph,Ii,Ie_p,Ie_sc] = get_I_parts(parameters,V,handles);
            currents{1}         = Iph;
            currents{2}         = Ii;
            currents{3}         = Ie_p;
            currents{4}         = Ie_sc;
            epop                = 2
        else
            ppar          = par2phys2(parameters,handles,1);
            [Iph,Ii,Ie_p] = get_I_parts(parameters,V,handles);
            currents{1}   = Iph;
            currents{2}   = Ii;
            currents{3}   = Ie_p;
            epop          = 1;
        end
    else
        if (handles.e_sc ~= 1)  %one e-pop
            ppar          = par2phys2(parameters,handles,1);
            [Iph,Ii,Ie_p] = get_I_parts(parameters,V,handles);
            currents{1}   = Iph;
            currents{2}   = Ii;
            currents{3}   = Ie_p;
            epop          = 1;
        else
            ppar                = par2phys2(parameters,handles,2);
            [Iph,Ii,Ie_p,Ie_sc] = get_I_parts(parameters,V,handles);
            currents{1}         = Iph;
            currents{2}         = Ii;
            currents{3}         = Ie_p;
            currents{4}         = Ie_sc;
            epop                = 2
        end
    end
end
```

B.4.2 fitting2.m

```
%FITTING2 is the main program that coordinates all the different steps in
%         the fitting procedure.
%
%   [parameters,fit,pars,fits] = fitting2(Vb,I,handles,phi,probe)

function [parameter,fit,pars,fits] = fitting2(Vb,I,handles,phi,probe)

parameter = [];
fit       = [];
pars      = [];
fits      = [];

%Turning off warnings
warning off MATLAB:divideByZero
warning off MATLAB:nearlySingularMatrix


%Setting some by hand, now I use the values for probe 2
Iph01 = 64e-9;
Iph02 = 6e-9;
Tph1  = 1.2;
Tph2  = 7.1;

%An initial determination of some parameters is made
dat_out = preliminaries(Vb,I);

if (length(dat_out) == 0)                   %If data is strange
    return;
end

b   = dat_out(2);
c   = dat_out(3);
d   = dat_out(4);
z   = dat_out(7);

if (handles.Vsc_manual == 1)                %Use manually set Vsc
    if (probe == 1)
        Vsc = handles.PAR1.Vsc;
    else
        Vsc = handles.PAR2.Vsc;
    end
else
    Vsc = dat_out(5);
```

```
end

if (handles.e_photo == 1)                   %Photocurrent or not
    f = dat_out(6);
else
    f = 0;
end

a   = dat_out(1)-f*(Iph01+Iph02);
if (a > 0)                                  %a must be negative
    a = -1e-9;
end

if (handles.e_sc == 1)                      %sc-photoelectrons used
    k   = dat_out(8);
    k_g = min([k*0.5 5e-9]);                %This is the k used for g
    k_c = k-k_g;                            %The k used for c

    h   = 1/1.2;                            %1.2 eV starting guess
    g   = k_g*(1/h);                        %Ie02
    d   = 1/1;                              %1 eV pop. 1
    c   = k_c*(1/d);
else
    g = 0;
    h = 0;
end

ph_par = [Iph01 Iph02 Tph1 Tph2];

if (handles.fit_try_all == 0)                       %A user specified fit
    par = [Iph01 Iph02 Tph1 Tph2 a b c d g h Vsc f z];
   [parameter,fit] = fit_data2(Vb,I,par,handles);
else                                                %Fit all combinations
    %First fit ram ions and plasmaelectrons (A)
    c = dat_out(3);
    d = dat_out(4);
    g = 0;
    h = 0;
    par = [Iph01 Iph02 Tph1 Tph2 a b c d g h Vsc f z];

    handles.e_plasma    = 1;        %plasmaelectrons
    handles.e_sc        = 0;        %no sc-electrons
    handles.e_photo     = 0;        %no photoelectrons
    handles.i_ram       = 1;        %ram ions
    handles.fit_partial = 1;        %fit partial data (one e-pop)

    [par1,fit1] = fit_data2(Vb,I,par,handles);

    par1(15) = 1; %This designates what kind of model 1 = A

    %Now fit ram ions, plasma- and photoelectrons (B)

    handles.e_plasma    = 1;        %plasmaelectrons
    handles.e_sc        = 0;        %no sc-electrons
    handles.e_photo     = 1;        %photoelectrons
    handles.i_ram       = 1;        %ram ions
    handles.fit_partial = 1;        %fit partial data (one e-pop)

    [par2,fit2] = fit_data2(Vb,I,par,handles);

    par2(15) = 2; %This designates what kind of model 2 = B
```

```
%Using values from (A) and (B) we now determine (C)-(F)
para = par1(1:13);

para2(1:6)   = par1(1:6);
para2(7)     = 0;          %c
para2(8)     = 1/1;        %d
para2(9)     = par1(7)*(1+par1(8)*par1(11));  %c,d --> g, g = c(1+dVsc)
para2(10)    = par1(8)/(1+par1(8)*par1(11));  %d --> h, h = d/(1+dVsc)
para2(11:13) = par1(11:13);

parb = par2(1:13);

parb2(1:6)   = par2(1:6);
parb2(7)     = 0;          %c
parb2(8)     = 1;          %d
parb2(9)     = par2(7)*(1+par2(8)*par2(11));  %c,d --> g, g = c(1+dVsc)
parb2(10)    = par2(8)/(1+par2(8)*par2(11));  %d --> h, h = d/(1+dVsc)
parb2(11:13) = par2(11:13);

%Fit therm ions and plasmaelectrons (C)

handles.e_plasma    = 1;        %plasmaelectrons
handles.e_sc        = 0;        %no sc-electrons
handles.e_photo     = 0;        %no photoelectrons
handles.i_ram       = 0;        %thermal ions
handles.fit_partial = 1;        %fit partial data (one e-pop)

[par3,fit3] = fit_data2(Vb,I,para,handles);

par3(15) = 3; %This designates what kind of model 3 = C

%Fit therm ions, plasma- and photoelectrons (D)

handles.e_plasma    = 1;        %plasmaelectrons
handles.e_sc        = 0;        %no sc-electrons
handles.e_photo     = 1;        %photoelectrons
handles.i_ram       = 0;        %thermal ions
handles.fit_partial = 1;        %fit partial data (one e-pop)

[par4,fit4] = fit_data2(Vb,I,parb,handles);

par4(15) = 4; %This designates what kind of model 4 = D

%Fit ram ions, plasma- and scphotoelectrons (E)

handles.e_plasma    = 1;        %plasmaelectrons
handles.e_sc        = 1;        %sc-electrons
handles.e_photo     = 0;        %no photoelectrons
handles.i_ram       = 1;        %ram ions
handles.fit_partial = 0;        %fit all data points (two e-pops)

[par5,fit5] = fit_data2(Vb,I,para2,handles);

par5(15) = 5; %This designates what kind of model 5 = E

%Fit ram ions, plasma-, scphoto- and photoelectrons (F)

handles.e_plasma    = 1;        %plasmaelectrons
handles.e_sc        = 1;        %sc-electrons
handles.e_photo     = 1;        %photoelectrons
handles.i_ram       = 1;        %ram ions
handles.fit_partial = 0;        %fit all data points (two e-pops)
```

```
[par6,fit6] = fit_data2(Vb,I,parb2,handles);

par6(15) = 6; %This designates what kind of model 6 = F

%Now using (C)-(F) fit (G) and (H)
parc(1:6)   = par3(1:6);
parc(7)     = 0;          %c
parc(8)     = 1;          %d
parc(9)     = par3(7)*(1+par3(8)*par3(11));  %c,d --> g, g = c(1+dVsc)
parc(10)    = par3(8)/(1+par3(8)*par3(11));  %d --> h, h = d/(1+dVsc)
parc(11:13) = par3(11:13);

pard(1:6)   = par4(1:6);
pard(7)     = 0;          %c
pard(8)     = 1;          %d
pard(9)     = par4(7)*(1+par4(8)*par4(11));  %c,d --> g, g = c(1+dVsc)
pard(10)    = par4(8)/(1+par4(8)*par4(11));  %d --> h, h = d/(1+dVsc)
pard(11:13) = par4(11:13);

pare(1:6)   = par5(1:6);
pare(7)     = 0;          %c
pare(8)     = 1;          %d
pare(9)     = par5(7)*(1+par5(8)*par5(11));  %c,d --> g, g = c(1+dVsc)
pare(10)    = par5(8)/(1+par5(8)*par5(11));  %d --> h, h = d/(1+dVsc)
pare(11:13) = par5(11:13);

parf(1:6)   = par6(1:6);
parf(7)     = 0;          %c
parf(8)     = 1;          %d
parf(9)     = par6(7)*(1+par6(8)*par6(11));  %c,d --> g, g = c(1+dVsc)
parf(10)    = par6(8)/(1+par6(8)*par6(11));  %d --> h, h = d/(1+dVsc)
parf(11:13) = par6(11:13);

%Fit therm ions, plasma- and scphotoelectrons (G)

handles.e_plasma    = 1;        %plasmaelectrons
handles.e_sc        = 1;        %sc-electrons
handles.e_photo     = 0;        %no photoelectrons
handles.i_ram       = 0;        %thermal ions
handles.fit_partial = 0;        %fit all data points (two e-pops)

[par7,fit7] = fit_data2(Vb,I,parc,handles);  %using values from c
[par8,fit8] = fit_data2(Vb,I,pare,handles);  %using values from e

par7(15) = 7; %This designates what kind of model 7 = G
par8(15) = 7; %This designates what kind of model 7 = G

%Fit therm ions, plasma-, scphoto- and photoelectrons (H)

handles.e_plasma    = 1;        %plasmaelectrons
handles.e_sc        = 1;        %sc-electrons
handles.e_photo     = 1;        %photoelectrons
handles.i_ram       = 0;        %thermal ions
handles.fit_partial = 0;        %fit all data points (two e-pops)

[par9,fit9]   = fit_data2(Vb,I,pard,handles);  %using values from d
[par10,fit10] = fit_data2(Vb,I,parf,handles);  %using values from f

par9(15)  = 8; %This designates what kind of model 8 = H
par10(15) = 8; %This designates what kind of model 8 = H
```

```
    %Now we can compare the different models and see which is best

    err         = [par1(14),par2(14),par3(14),par4(14),par5(14),...
                   par6(14),par7(14),par8(14),par9(14),par10(14)];
    fits        = [fit1(2,:);fit2(2,:);fit3(2,:);fit4(2,:);fit5(2,:);...
                   fit6(2,:);fit7(2,:);fit8(2,:);fit9(2,:);fit10(2,:)];
    pars        = [par1;par2;par3;par4;par5;par6;par7;par8;par9;par10];
    [fit_t,parameter] = choose_fit(Vb,I,fits,pars,err);

    stl   = 10*length(Vb);
    V_fit = linspace(min(Vb),max(Vb),stl);
    fit(1,:) = V_fit;
    fit(2,:) = fit_t;
end
```

B.4.3 preliminaries.m

```
%PRELIMINARIES is a function that should be used in the start of the
%              fitting procedure. It calculates starting values of some of
%              the parameters, as well as Vsc, z and other usables
%
%   [dat_out] = preliminaries(Vb,I)

function [dat_out] = preliminaries(Vb,I)

dat_out =  [];

lV = length(Vb);
lI = length(I);

if lV == lI
    %HERE THE LEAK SHOULD BE SUBTRACTED... PERHAPS

    z                = zero_cross(Vb,I);       %determine zero-crossing
    if (length(z) == 0)
        disp('Data possibly corrupt, ending fit')
        return;
    end

    f                = -1;                      %This must be fixed

    [Vsc1,Vsc2,Vsc3] = find_scpot(Vb,I,z);     %determine Vsc

    if (Vsc1+z < -4)                            %Vsc should be in
        Vsc = -z;                               %vicinity of z
    else
        Vsc = Vsc1;
    end

    %Now Ie0 and Te will be determined
    [Ie0,Te,k,Vsc] = fit_single_e(Vb,I,Vsc);

    c = Ie0;
    d = 1/Te;

    %Setting b and a (a is used if the probe is in eclipse)
    Vp  = Vsc + Vb;
    len = length(Vp);
    len2 = length(find(Vp < 0));
    V    = Vp(1:(floor(len2*0.5))); %50 percent of the negative probe sweep
    I    = I(1:(floor(len2*0.5)));

    P    = polyfit(V,I,1);          %fitting straight line
    b    = P(1);
    a    = P(2);

    %Set the output
    dat_out(1)  = a;
    dat_out(2)  = b;
    dat_out(3)  = c;
    dat_out(4)  = d;
    dat_out(5)  = Vsc;
    dat_out(6)  = f;
    dat_out(7)  = z;
    dat_out(8)  = k;
end
```

B.4.4 zero_cross.m

```
%ZERO_CROSS determines the zero crossing of the current (i.e. I = 0)
%
%   z = zero_cross(Vb,I)

function z = zero_cross(Vb,I)

lenV = length(Vb);
lenI = length(I);
z    = 0;
if lenV ~= lenI
    disp('Vb and I need to be same length')
    return;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Find datapoints lower and higher than zero, the following is Reine Gills%
%code (unless otherwise stated)                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lt  = find(I < 0);
gt  = find(I > 0);
Dlt = [Vb(lt), I(lt)];              %Data with current lower than zero
Dgt = [Vb(gt), I(gt)];              %Data with current higher than zero

if(~isempty(Dlt) && ~isempty(Dgt))
    Dlt=sortrows(Dlt,2); % Sort by current
      Dlt=flipud(Dlt);

      % Select atmost the 4 lowest curr. values and sort by bias
      ui=size(Dlt,1);
      if(ui>4)
        ui=4;
      end
      Dlt=sortrows(Dlt(1:ui,:),1);
      Dlt=flipud(Dlt);
      % At this point the first row in Dlt is a data point with
      % high probability to be close and to the left of the rightmost
      % zero crossing..second value has lower probability and so on

      Dgt=sortrows(Dgt,2); % Sort by current
                           % Select atmost the 4 lowest
```

```
                         % curr. values and sort by bias
      ui=size(Dgt,1);
      if(ui>4)
        ui=4;
      end
      Dgt=sortrows(Dgt(1:ui,:),1);
      % At this point the first row in Dlt is a data point with
      % high probability to be close and to the right of the rightmost
      % zero crossing..second value has lower probability and so on

      % Use most probable points draw a line between them
      % and return bias at zero current
      vp=Dgt(1,1);
      ip=Dgt(1,2);
      vn=Dlt(1,1);
      in=Dlt(1,2);

      z=vp-ip*(vp-vn)./(ip-in);
else
    disp('Error, all data above or below zero!');
    z = [];
end
```

B.4.5 moving_average.m

```
%MOVING_AVERAGE calculates a pointvalue at each point, based on the average
%               of its 8 neighbouring points, symmetrically. The
%               endpoints are calculated by neighbours back and
%               forth respectively.
%
%   y_av = moving_average(x,y)

function y_av = moving_average(x,y)

y_av = [];

lx = length(x);
ly = length(y);

if (lx == ly)               %must be same lengths
    %for first points a forward average is taken
    y_av(1)      = (y(1)+y(2)+y(3)+y(4)+y(5)+y(6)+y(7)+y(8)+y(9))/9;
    y_av(2)      = (y(2)+y(3)+y(4)+y(5)+y(6)+y(7)+y(8)+y(9)+y(10))/9;
    y_av(3)      = (y(3)+y(4)+y(5)+y(6)+y(7)+y(8)+y(9)+y(10)+y(11))/9;
    y_av(4)      = (y(4)+y(5)+y(6)+y(7)+y(8)+y(9)+y(10)+y(11)+y(12))/9;
    %middle points using a symmetric one
    y_av(5:lx-4) = (y(1:lx-8)+y(2:lx-7)+y(3:lx-6)+y(4:lx-5)+y(5:lx-4)+...
                    y(6:lx-3)+y(7:lx-2)+y(8:lx-1)+y(9:lx))/9;
    %last 4-2 points are calculated using a 3 point average
    y_av(lx-3)   = (y(lx-4)+y(lx-3)+y(lx-2))/3;
    y_av(lx-2)   = (y(lx-3)+y(lx-2)+y(lx-1))/3;
    y_av(lx-1)   = (y(lx-2)+y(lx-1)+y(lx))/3;
    %last point is taken as a 2 point backward average
    y_av(lx)     = (y(lx-1)+y(lx))/2;
end
```

B.4.6 find_scpot.m

```
%FIND_SCPOT Approximate spacecraft potential
%
%   [Vsc1,Vsc2,Vsc3] = find_scpot(Vb,Ib,z)
%
%   returns an approximation of the spacecraft potential. Vsc1 is just a
%   global determination while Vsc2 is a determination around the
%   zero-point z. Vsc3 is a determination based on where the curve
%   "shoots up" from a line
%
%   Vb is the bias potential, Ib the bias current and z the
%   zero-crossing of the current.
%
%   z is used as a first location of the vicinity in which Vsc should be,
%   then the maximum of the absolute of the second derivative of the
%   current is taken as Vsc, i.e. Vsc = max(abs((d2I))
function [Vsc1,Vsc2,Vsc3] = find_scpot(Vb,Ib,z)

Vsc1 = [];
Vsc2 = [];
Vsc3 = [];

lV = length(Vb);
lI = length(Ib);
if lV == lI
    %     Ib_smooth(1:lI-1) = (Ib(1:lI-1)+Ib(2:lI))./2;
    %     Ib_smooth(lI)     = Ib(lI);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %To smooth the data the mean value of each datapoint with its neighbour is%
    %taken.                                                                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    points = length(Vb);
    lgolay = max([(2*floor(points/12)+1) 5]);

    Ib_smooth = smooth(Ib,lgolay,'sgolay');

    Vb2                 = Vb;
    Ib2                 = Ib_smooth;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Taking the second derivative%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    d2Ib = d2(Vb2,Ib2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %"Global" Vsc as the maximum of the second derivative%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Vsc1 = -Vb2(max(find(abs(d2Ib) == max(abs(d2Ib)))));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Now an interval around z is used to look for Vsc, this is done as a%
    %"quality" check on the spacecraft potential found.                 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if z ~= 0
        interval = 2;           %Size of interval (up and down)
        ind1     = find(Vb2 > z-interval);
        ind2     = find(Vb2 < z+interval);
        ind1     = ind1(1);
        ind2     = ind2(length(ind2));
        V2       = Vb2(ind1:ind2);
        d2Ib2    = d2Ib(ind1:ind2);
        Vsc2     = -V2(max(find(abs(d2Ib2) == max(abs(d2Ib2)))));
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Look for an approximate Vsc by comparing a straight line (given%
    %from the low 10% of the sweep) to the real data, seeing       %
    %where it "shoots up"                                          %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    try
        ind    = floor(0.1*length(Vb2));
        P      = polyfit(Vb2(1:ind),Ib2(1:ind)',1);     %Its a line
        max_er = max(abs(diff(Ib2(1:ind))));
        m      = P(2)+max_er*4;                         %to take care of noise
        I_err  = linspace(m,m,length(Vb2));
        err    = I_err - Ib2;
        indi   = find(err < 0, 1, 'first');
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Now that we have an approximate Vsc lets look at the current   %
        %values that are located around this point. Using derivatives to%
        %get Vsc                                                        %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        interval    = 3;
        ind1        = find(Vb2 > Vb2(indi)-interval);
        ind2        = find(Vb2 < Vb2(indi)+interval);
        ind1        = ind1(1);
        ind2        = ind2(length(ind2));
        V3          = Vb2(ind1:ind2);
        d2Ib3       = d2Ib(ind1:ind2);
        Vsc3        = -V3(max(find(abs(d2Ib3) == max(abs(d2Ib3)))));
    catch
        Vsc3 = 0;
    end
end
```

B.4.7 d2.m

```
%d2 Approximate second derivative
%   [d2y] = d2(x,y) returns the second derivative of y with respect to x.
%   d2y is a vector of the same length as y. The vectors should be sorted
%   in ascending order. If the length of the data is less than 5 gradient
%   will be used.
%
%   The differentiation is:
%   y_0''=(2*y_4-27*y_3+270*y_2-490*y_0+270*y_-1-27*y_-2+2*y_-3)/(180*h^2)
%
%   [d2y,d3y] = d2(x,y) provides both the second and the third derivative
%   where the third is evaluated using a similar scheme but only 6 points

function [d2y,d3y] = d2(x,y)

d2y = [];
d3y = [];

lx = length(x);   %checking sizes
ly = length(y);

lgolay = max([(2*floor(lx/12)+1) 5]);
y      = smooth(y,lgolay,'sgolay');

% if (lx == ly && lx >= 5)
%     h           = x(2)-x(1);    %the steplength is the same throughout
%     %%%%%%%%%%%%%%%%%%%%%%%
%     %The second derivative%
%     %%%%%%%%%%%%%%%%%%%%%%%
%     %Use forward differentiation for left boundary
%     d2y(1)      = (11*y(5)-56*y(4)+114*y(3)-104*y(2)+35*y(1))/(12*h^2);
%     d2y(2)      = (11*y(6)-56*y(5)+114*y(4)-104*y(3)+35*y(2))/(12*h^2);
%     d2y(3)      = (11*y(7)-56*y(6)+114*y(5)-104*y(4)+35*y(3))/(12*h^2);
%     %Now use the 7-point symmetric formula
%     d2y(4:lx-3) = (2*y(7:lx)-27*y(6:lx-1)+270*y(5:lx-2)-490*y(4:lx-3)+...
%                    270*y(3:lx-4)-27*y(2:lx-5)+2*y(1:lx-6))/(180*h^2);
%     %Use backward differentiation for right boundary
%     d2y(lx-2)   = (35*y(lx-2)-104*y(lx-3)+114*y(lx-4)-...
%                    56*y(lx-5)+11*y(lx-6))/(12*h^2);
%     d2y(lx-1)   = (35*y(lx-1)-104*y(lx-2)+114*y(lx-3)-...
%                    56*y(lx-4)+11*y(lx-5))/(12*h^2);
%     d2y(lx)     = (35*y(lx)-104*y(lx-1)+114*y(lx-2)-...
%                    56*y(lx-3)+11*y(lx-4))/(12*h^2);
%     %%%%%%%%%%%%%%%%%%%%%%
%     %The third derivative%
%     %%%%%%%%%%%%%%%%%%%%%%
%     %Use forward differentiation for left boundary
%     d3y(1)      = (-3*y(5)+14*y(4)-24*y(3)+18*y(2)-5*y(1))/(2*h^3);
%     d3y(2)      = (-3*y(6)+14*y(5)-24*y(4)+18*y(3)-5*y(2))/(2*h^3);
%     d3y(3)      = (-3*y(7)+14*y(6)-24*y(5)+18*y(4)-5*y(3))/(2*h^3);
%     %Now use the 6-point symmetric formula
%     d3y(4:lx-3) = (-y(7:lx)+8*y(6:lx-1)-13*y(5:lx-2)+13*y(3:lx-4)-...
%                    8*y(2:lx-5)+y(1:lx-6))/(8*h^3);
%     %Use backward differentiation for the right boundary
%     d3y(lx-2)   = (5*y(lx-2)-18*y(lx-3)+24*y(lx-4)-14*y(lx-5)+...
%                    3*y(lx-6))/(2*h^3);
%     d3y(lx-1)   = (5*y(lx-1)-18*y(lx-2)+24*y(lx-3)-14*y(lx-4)+...
%                    3*y(lx-5))/(2*h^3);
%     d3y(lx)     = (5*y(lx)-18*y(lx-1)+24*y(lx-2)-14*y(lx-3)+...
%                    3*y(lx-4))/(2*h^3);
% else
    %using simple gradient
    d2y = gradient(smooth(gradient(y),lgolay,'sgolay'),x);

    d3y = gradient(smooth(d2y,lgolay,'sgolay'),x);
%end
```

B.4.8 fit_single_e.m

```
%FIT_SINGLE_E fits the upper part of the electron side as a line assuming
%             one electron population
%
%   [Ie0,Te,k] = fit_single_e(Vb,I,Vsc)

function [Ie0,Te,k,Vsc1] = fit_single_e(Vb,I,Vsc0)

Ie0 = [];
Te  = [];

Vp = Vb + Vsc0;

%Take the last 70% as the data to fit to
indp = find(Vp > 0);
V    = Vp(indp);
I    = I(indp);
```

```
lenV = length(V);
indl = ceil(lenV*0.30);
Vh   = V(indl:lenV);
Ih   = I(indl:lenV);

%Now fit using a polynomial of degree 1, i.e. a line
P = polyfit(Vh,Ih,1);
k = P(1);
m = P(2);

Te0 = 0.5;

if (k > 25e-9)
    Vsc1 = Vsc0-Te0+m/k;
else
    Vsc1 = Vsc0;
end
m   = k*Te0;
Ie0 = m;
Te  = Te0;
```

B.4.9 determine_ecl.m

```
%DETERMINE_ECL does a determination whether the probe is in eclipse (wake)
%             of the spacecraft or not.
%
%   f = determine_ecl(Vb,I,Vsc,phi,probe)
%
%   phi is the elevation angle of the sun in radians
%   probe is an integer specifying what probe is used (1 or 2)
function f = determine_ecl(Vb,I,Vsc,phi,probe)

ecl_value = 2e-9;   %This is the value used to check for eclipse
%abs(sum(I(find(Vb < Vsc))))
if probe == 1;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Probe one is in eclipse for phi >= 48 deg and phi <= 75 deg approx.%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if phi >= 0.83775804095728 && phi <= 1.30899693899575
        disp('Probe 1 is in eclipse')
        f = -eps;
    else
        disp('Probe 1 is probably not in eclipse')
        if (abs(sum(I(find(Vb < Vsc)))) < ecl_value)
            disp('Correction, probe 1 is in eclipse from absolute ion value')
            f = -eps;
        else
            f = -1;
        end
    end
end
if probe == 2
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Probe 2 is in eclipse for phi <= -27 deg and phi >= -72 deg approx.%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     if phi <= -0.47123889803847 && phi >= -1.25663706143592
         disp('Probe 2 is in eclipse')
         f = -eps;
     else
```

```
         disp('Probe 2 is probably not in eclipse')
         if (abs(sum(I(find(Vb < Vsc)))) < ecl_value)
             disp('Correction, probe 2 is in eclipse from absolute ion value')
             f = -eps;
         else
             f = -1;
         end
     end
end
```

B.4.10 fit_data2.m

```
%FIT_DATA2 fits the parameters to the data in several steps and in different
%          ways depending on what models are used
%
%   [par_out,fits_out] = fit_data2(Vb,I,par,handles)
%
%   par is a row or column vector containing all the parameters needed for
%   the fit: [Iph01 Iph02 Tph1 Tph2 a b c d g h Vsc f z]

function [par_out,fits_out] = fit_data2(Vb,I,par,handles)

par_out  = [];
fits_out = [];
fit2_ok  = 0;

%Setting up the parameters
Iph01 = par(1);
Iph02 = par(2);
Tph1  = par(3);
Tph2  = par(4);
a     = par(5);
b     = par(6);
c     = par(7);
d     = par(8);
g     = par(9);
h     = par(10);
Vsc   = par(11);
f     = par(12);

ph_par = [Iph01 Iph02 Tph1 Tph2];

%Setting up some bounds
amin = -1e-3;
amax = -1e-12;
bmin = 0;
bmax = 1e-2;
cmin = 0;
cmax = 1e-2;
dmin = 1/100;
dmax = 1/0.1;
gmin = 0;
gmax = 1e-2;
hmin = 1/100;
hmax = 1/0.0001;
if (max(I) < 300e-9)
    Vscmin = Vsc-7;
    Vscmax = Vsc+7;
else
```

```
    Vscmin = Vsc-1;
    Vscmax = Vsc+1;
end
fmin = -2;
fmax = 0;

%Determine the interval to use
% if (handles.fit_partial == 1)       %use data below 50 nA
%     V_1  = Vb(find(I < 50e-9));
%     I_1  = I(find(I < 50e-9));
%     if (V_1(length(V_1)) < Vsc)        %if this is below Vsc, use Vsc+2 and
%         V_1 = Vb(find(Vb < (-Vsc+2)));  %below
%         I_1 = I(find(Vb < (-Vsc+2)));
%     end
% else                                %use entire data set
%     V_1 = Vb;
%     I_1 = I;
% end
% V_1
% I_1

if (handles.fit_partial == 1)
    V_1 = Vb(find(Vb < 8));
    I_1 = I(find(Vb < 8));
else
    V_1 = Vb;
    I_1 = I;
end

%Set up information for the fit evaluation
nr_points = 10*length(Vb);                        %# points used to fit
V_fit     = linspace(min(Vb),max(Vb),nr_points);  %Entire data used

%If debug mode, plot the initial value plot
if (handles.debug == 1)
    fitu = feval(@model,[a b c d g h Vsc f],V_fit,ph_par,18,[],handles);
    figure(10), plot(Vb,I,'.',V_fit,fitu,'red');
    grid on;
    title('Initial guess plot');
    xlabel('press any key');
    format long;
    par'
    pause
end

%Defining the optimization ranges for lsqcurvefit
%Change here if a lower tolerance is desirable
opt = optimset('lsqcurvefit');
opt.TolX        = 1E-25;
opt.TolFun      = 1E-15;
opt.MaxFunEvals = 5000;
opt.MaxIter     = 5000;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Starting the fitting procedure%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Trying to determine Vsc better. Here different
%parameters are kept fixed depending on what the user has choosen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
[V2,I2]    = weighting(V_1,I_1,-Vsc,5,100);

%Ram ions, plasmaelectrons (A) or Therm ions, plasmaelectrons (C)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in = [a Vsc];
    lowbound  = [amin Vscmin];
    highbound = [amax Vscmax];%[amax Vscmax zmax];
    vargin    = [b c d];
    ref       = 1;
end
%Ram ions, plasma- and photoelectrons (B) or Therm ions, plasma- and photo-
%                                            electrons (D)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 1) ||(handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in = [a Vsc f];
    lowbound  = [amin Vscmin fmin];
    highbound = [amax Vscmax fmax];%[amax Vscmax fmax zmax];
    vargin    = [b c d];
    ref       = 2;
end
%Ram ions, plasma- and sc-electrons (E) or Therm ions-, plasma- and
%                                          sc-electrons (G)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in = [a Vsc];
    lowbound  = [amin Vscmin];
    highbound = [amax Vscmax];%[amax Vscmax zmax];
    vargin    = [b c d g h];
    ref       = 3;
end
%Ram ions, plasma-, sc- and photoelectrons (F) or Therm ions, plasma-,
%                                                 sc- and photoelectrons
%                                                 (H)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 1) || (handles.e_plasma == 1 ...
    && handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in = [a Vsc f];
    lowbound  = [amin Vscmin fmin];
    highbound = [amax Vscmax fmax];%[amax Vscmax fmax zmax];
    vargin    = [b c d g h];
    ref       = 4;
end
try
    [params_out,resnorm]  = lsqcurvefit(@model,params_in,V2,I2,lowbound,...
                                        highbound,opt,ph_par,ref,vargin,...
                                        handles);
catch
    params_out(1:4) = 0;
    resnorm         = inf;
end
%Save fit
a   = params_out(1);
Vsc = params_out(2);
if (ref == 2 || ref == 4)
```

```
    f   = params_out(3);
else
    f = 0;
end

fit1  = feval(@model,params_out,V_fit,ph_par,ref,vargin,handles);
pars1 = [a b c d g h Vsc f];

%Plot if in debug mode
if (handles.debug == 1)
    figure(10), plot(Vb,I,'.',V_fit,fit1,'red');
    grid on;
    title('Fit nr. 1, Focus on Vsc');
    xlabel('press any key');
    pars1'
    pause
end

%Now a better evaluation of Te is obtained by looking close to the
%spacecraft potential (PROBLEM: Vsc maste vara bestamd ok nu iaf.)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

indic = find(Vb < -Vsc+1,3,'last');
Itry = I(indic);
Vtry = Vb(indic);
if (handles.debug == 1)
    figure(10),plot(Vb,I,Vtry,Itry,'.')
    grid on
    pause
end

opt.TolX        = 1e-55;
opt.TolFun      = 1e-25;
opt.MaxFunEvals = 15000;
opt.MaxIter     = 15000;

a     = par(5);
b     = par(6);
c     = par(7);
d     = par(8);
g     = par(9);
h     = par(10);
Vsc   = par(11);
f     = par(12);

params_intry = [c d f];
lowbound     = [cmin dmin fmin];
highbound    = [cmax dmax fmax];%[amax Vscmax fmax zmax];
vargintry    = [a b g h Vsc];
reftry       = 23;
try
    [params_outry,resnormtry]  = lsqcurvefit(@model,params_intry,Vtry,Itry,lowbound,...
                                        highbound,opt,ph_par,reftry,vargintry,...
                                        handles);
catch
    params_outry(1:3) = 0;
    resnormtry        = inf;
end

c = params_outry(1);
d = params_outry(2);
f = params_outry(3);
```

```
paru   = [a b c d g h Vsc f];
fittry = feval(@model,paru,V_fit,ph_par,18,[],handles);

%Plot if in debug mode
if (handles.debug == 1)
    figure(10), plot(Vb,I,'.',V_fit,fittry,'red');
    grid on;
    title('Electron try');
    xlabel('press any key');
    paru'
    pause
end

opt.TolX        = 1e-25;
opt.TolFun      = 1e-15;
opt.MaxFunEvals = 5000;
opt.MaxIter     = 5000;

%Now try to get a good fix on the electron populations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (Vsc > 0)                             %only if Vsc > 0 since otherwise
    ind        = find(Vb > 0);           %there is no possibility to
    percent    = ceil(0.70*length(ind));%distinguish the two e-pops.
    weight     = Vb(ind(percent));
    fit2_ok    = 1;                      %This is for later
    if (weight < 10)
        interval = 3;
    else
        interval = (weight-8+2);
    end
    [V2,I2]    = weighting(V_1,I_1,weight,interval,100);

    %Ram ions, plasmaelectrons (A) or Therm ions, plasmaelectrons (C)
    if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
        && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
        handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 0))

        params_in2 = [c d Vsc];
        lowbound   = [cmin dmin Vscmin];
        highbound  = [cmax dmax Vscmax];%[dmax Vscmax zmax];
        vargin2    = [a b];
        ref2       = 5;
    end
    %Ram ions, plasma- and photoelectrons (B) or Therm ions, plasma- and photo-
    %                                            electrons (D)
    if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
        && handles.e_photo == 1) ||(handles.e_plasma == 1 && ...
        handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 1))

        params_in2 = [c d Vsc f];
        lowbound   = [cmin dmin Vscmin fmin];
        highbound  = [cmax dmax Vscmax fmax];
        vargin2    = [a b];
        ref2       = 6;
    end
    %Ram ions, plasma- and sc-electrons (E) or Therm ions-, plasma- and
    %                                       sc-electrons (G)
    if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
        && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
        handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 0))
        params_in2 = [c d g h Vsc];
        lowbound   = [cmin dmin gmin hmin Vscmin];
        highbound  = [cmax dmax gmax hmax Vscmax];
        vargin2    = [a b];
        ref2       = 7;
    end
    %Ram ions, plasma-, sc- and photoelectrons (F) or Therm ions, plasma-,
    %                                                  sc- and photoelectrons
    %                                                  (H)
    if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
        && handles.e_photo == 1) || (handles.e_plasma == 1 ...
        && handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 1))

        params_in2 = [c d g h Vsc f];
        lowbound   = [cmin dmin gmin hmin Vscmin fmin];
        highbound  = [cmax dmax gmax hmax Vscmax fmax];
        vargin2    = [a b];
        ref2       = 8;
    end

    try
        [params_out2,resnorm2]  = lsqcurvefit(@model,params_in2,V2,I2,...
                                              lowbound,highbound,opt,ph_par,...
                                              ref2,vargin2,handles);

    catch
        params_out2(1:6) = 0;
        resnorm2         = inf;
    end

    %Save fit
    c = params_out2(1);
    d = params_out2(2);
    if (length(params_out2) == 3)
        Vsc = params_out2(3);
        f   = 0;
        g   = 0;
        h   = 0;
    end
    if (length(params_out2) == 4)
        Vsc = params_out2(3);
        f   = params_out2(4);
        g   = 0;
        h   = 0;
    end
    if (length(params_out2) == 5)
        g   = params_out2(3);
        h   = params_out2(4);
        Vsc = params_out2(5);
        f   = 0;
    end
    if (length(params_out2) == 6)
        g   = params_out2(3);
        h   = params_out2(4);
        Vsc = params_out2(5);
        f   = params_out2(6);
    end

    fit2  = feval(@model,params_out2,V_fit,ph_par,ref2,vargin2,handles);
    pars2 = [a b c d g h Vsc f];
```

```
    if (handles.debug == 1)
        figure(10), plot(Vb,I,'.',V_fit,fit2,'red');
        grid on;
        title('Fit nr. 2, Focus on electron population');
        xlabel('press any key');
        pause
        pars2'
    end
end

%Now try to get a good fix on the ion side
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ind         = find(Vb < 0);
percent     = floor(0.50*length(ind));
weight      = Vb(ind(percent));
[V2,I2]     = weighting(V_1,I_1,weight,10,300);

%Ram ions, plasmaelectrons (A) or Therm ions, plasmaelectrons (C)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in4 = [a b c d Vsc];
    lowbound   = [amin bmin cmin dmin Vscmin];
    highbound  = [amax bmax cmax dmax Vscmax];%[dmax Vscmax zmax];
    vargin4    = [];
    ref4       = 14;
end
%Ram ions, plasma- and photoelectrons (B) or Therm ions, plasma- and photo-
%                                            electrons (D)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 1) ||(handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in4 = [a b c d Vsc f];
    lowbound   = [amin bmin cmin dmin Vscmin fmin];
    highbound  = [amax bmax cmax dmax Vscmax fmax];
    vargin4    = [];
    ref4       = 15;
end
%Ram ions, plasma- and sc-electrons (E) or Therm ions-, plasma- and
%                                          sc-electrons (G)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in4 = [a b c d Vsc];
    lowbound   = [amin bmin cmin dmin Vscmin];
    highbound  = [amax bmax cmax dmax Vscmax];
    vargin4    = [g h];
    ref4       = 16;
end
%Ram ions, plasma-, sc- and photoelectrons (F) or Therm ions, plasma-,
%                                                 sc- and photoelectrons
%                                                 (H)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 1) || (handles.e_plasma == 1 ...
    && handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in4 = [a b c d Vsc f];
    lowbound   = [amin bmin cmin dmin Vscmin fmin];
    highbound  = [amax bmax cmax dmax Vscmax fmax];
    vargin4    = [g h];
    ref4       = 17;
end

try
    [params_out4,resnorm4]  = lsqcurvefit(@model,params_in4,V2,I2,...
                                          lowbound,highbound,opt,ph_par,...
                                          ref4,vargin4,handles);
catch
    params_out4(1:6) = 0;
    resnorm4         = inf;
end

%Save fit
a = params_out4(1);
b = params_out4(2);
c = params_out4(3);
d = params_out4(4);
if (ref4 == 14)
    Vsc = params_out4(5);
    f   = 0;
    g   = 0;
    h   = 0;
end
if (ref4 == 15)
    Vsc = params_out4(5);
    f   = params_out4(6);
    g   = 0;
    h   = 0;
end
if (ref4 == 16)
    Vsc = params_out4(5);
end
if (ref4 == 17)
    Vsc = params_out4(5);
    f   = params_out4(6);
end

fit4  = feval(@model,params_out4,V_fit,ph_par,ref4,vargin4,handles);
pars4 = [a b c d g h Vsc f];

if (handles.debug == 1)
    figure(10), plot(Vb,I,'.',V_fit,fit4,'red');
    grid on;
    title('Fit nr. 3, Focus on ion side');
    xlabel('press any key');
    pars4'
    pause
end

%Finally fit all with the obtained values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (fit2_ok == 1)
    res         = [resnorm,resnorm2,resnorm4];
    fits        = [fit1;fit2;fit4];       %Matrix containing the fits
    pars        = [pars1;pars2;pars4];    %Matrix containing the parameters
else
    res         = [resnorm,resnorm4];
    fits        = [fit1;fit4];           %Matrix containing the fits
    pars        = [pars1;pars4];         %Matrix containing the parameters
end
```

```
[fit,param] = choose_fit(Vb,I,fits,pars,res);

a   = param(1);
b   = param(2);
c   = param(3);
d   = param(4);
g   = param(5);
h   = param(6);
Vsc = param(7);
f   = param(8);

[V2,I2] = weighting(V_1,I_1,-Vsc,1,100);

%Ram ions, plasmaelectrons (A) or Therm ions, plasmaelectrons (C)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in3 = [a c d Vsc];
    lowbound   = [amin cmin dmin Vscmin];
    highbound  = [amax cmax dmax Vscmax];
    vargin3    = [b];
    ref3       = 9;
end
%Ram ions, plasma- and photoelectrons (B) or Therm ions, plasma- and photo-
%                                            electrons (D)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 1) ||(handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in3 = [a c d Vsc f];
    lowbound   = [amin cmin dmin Vscmin fmin];
    highbound  = [amax cmax dmax Vscmax fmax];
    vargin3    = [b];
    ref3       = 10;
end
%Ram ions, plasma- and sc-electrons (E) or Therm ions-, plasma- and
%                                          sc-electrons (G)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in3 = [a c d g h Vsc];
    lowbound   = [amin cmin dmin gmin hmin Vscmin];
    highbound  = [amax cmax dmax gmax hmax Vscmax];
    vargin3    = [b];
    ref3       = 11;
end
%Ram ions, plasma-, sc- and photoelectrons (F) or Therm ions, plasma-,
%                                                 sc- and photoelectrons
%                                                 (H)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 1) || (handles.e_plasma == 1 ...
    && handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in3 = [a c d g h Vsc f];
    lowbound   = [amin cmin dmin gmin hmin Vscmin fmin];
    highbound  = [amax cmax dmax gmax hmax Vscmax fmax];
    vargin3    = [b];
    ref3       = 12;
end
```

```
try
    [params_out3,resnorm3]  = lsqcurvefit(@model,params_in3,V2,I2,...
                                        lowbound,highbound,opt,ph_par,...
                                        ref3,vargin3,handles);
catch
    params_out3(1:7) = 0;
    resnorm3         = inf;
end

%Save the data for this fit
a = params_out3(1);
c = params_out3(2);
d = params_out3(3);
if (length(params_out3) == 4)
    Vsc = params_out3(4);
    f   = 0;
    g   = 0;
    h   = 0;
end
if (length(params_out3) == 5)
    Vsc = params_out3(4);
    f   = params_out3(5);
    g   = 0;
    h   = 0;
end
if (length(params_out3) == 6)
    g   = params_out3(4);
    h   = params_out3(5);
    Vsc = params_out3(6);
    f   = 0;
end
if (length(params_out3) == 7)
    g   = params_out3(4);
    h   = params_out3(5);
    Vsc = params_out3(6);
    f   = params_out3(7);
end


fit3  = feval(@model,params_out3,V_fit,ph_par,ref3,vargin3,handles);
pars3 = [a b c d g h Vsc f];

if (handles.debug == 1)
    figure(10), plot(Vb,I,'.',V_fit,fit3,'red');
    grid on;
    title('Fit nr. 4, All parameters free');
    xlabel('press any key');
    pars3'
    pause
end

%%%%%%%%%%%%%%%%%%%%%%%%
%Decide what fit to use%
%%%%%%%%%%%%%%%%%%%%%%%%
if (size(fit1,1) > 1)
    fit1 = fit1';
end
if (fit2_ok == 1)
    if (size(fit2,1) > 1)
        fit2 = fit2';
    end
end
```

```
if (size(fit3,1) > 1)
    fit3 = fit3';
end
if (size(fit4,1) > 1)
    fit4 = fit4';
end

if (fit2_ok == 1)
    res         = [resnorm,resnorm2,resnorm3,resnorm4];
    fits        = [fit1;fit2;fit3;fit4];      %Matrix containing the fits
    pars        = [pars1;pars2;pars3;pars4];  %Matrix containing the parameters
else
    res         = [resnorm,resnorm3,resnorm4];
    fits        = [fit1;fit3;fit4];      %Matrix containing the fits
    pars        = [pars1;pars3;pars4];   %Matrix containing the parameters
end

[fit,param] = choose_fit(Vb,I,fits,pars,res);

%plot the choosen fit if in debug mode
if (handles.debug == 1)
    figure(10), plot(Vb,I,'.',V_fit,fit,'red');
    grid on;
    title('Choosen fit, next comes determination of Te');
    xlabel('press any key');
    param'
    pause
end

%Determine d,c better
%%%%%%%%%%%%%%%%%%%%%
a   = param(1);
b   = param(2);
c   = param(3);
d   = param(4);
g   = param(5);
h   = param(6);
Vsc = param(7);
f   = param(8);

[V2,I2]    = weighting(V_1,I_1,-Vsc+1,1.5,100);

opt.TolX        = 1e-55;
opt.TolFun      = 1e-15;
opt.MaxFunEvals = 5000;
opt.MaxIter     = 5000;

%Ram ions, plasmaelectrons (A) or Therm ions, plasmaelectrons (C)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in5 = [c d];
    lowbound   = [cmin dmin];
    highbound  = [cmax dmax];%[amax Vscmax zmax];
    vargin5    = [a b Vsc];
    ref5       = 19;
end
%Ram ions, plasma- and photoelectrons (B) or Therm ions, plasma- and photo-
%                                         electrons (D)
if ((handles.e_plasma == 1 && handles.e_sc == 0 && handles.i_ram == 1 ...
    && handles.e_photo == 1) ||(handles.e_plasma == 1 && ...
    handles.e_sc == 0 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in5 = [c d];
    lowbound  =  [cmin dmin];
    highbound =  [cmax dmax];%[amax Vscmax fmax zmax];
    vargin5    = [a b Vsc f];
    ref5       = 20;
end
%Ram ions, plasma- and sc-electrons (E) or Therm ions-, plasma- and
%                                           sc-electrons (G)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 0) || (handles.e_plasma == 1 && ...
    handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 0))

    params_in5 = [c d];
    lowbound   = [cmin dmin];
    highbound  = [cmax dmax];%[amax Vscmax zmax];
    vargin5    = [a b g h Vsc];
    ref5       = 21;
end
%Ram ions, plasma-, sc- and photoelectrons (F) or Therm ions, plasma-,
%                                                  sc- and photoelectrons
%                                                  (H)
if ((handles.e_plasma == 1 && handles.e_sc == 1 && handles.i_ram == 1 ...
    && handles.e_photo == 1) || (handles.e_plasma == 1 ...
    && handles.e_sc == 1 && handles.i_ram == 0 && handles.e_photo == 1))

    params_in5 = [c d];
    lowbound   = [cmin dmin];
    highbound  = [cmax dmax];%[amax Vscmax fmax zmax];
    vargin5    = [a b g h Vsc f];
    ref5       = 22;
end
try
    [params_out5,resnorm5]  = lsqcurvefit(@model,params_in5,V2,I2,lowbound,...
                                    highbound,opt,ph_par,ref5,vargin5,...
                                    handles);
catch
    params_out5(1:4) = 0;
    disp('c and d fit not correct')
    resnorm5         = inf;
end

%Save fit
c = params_out5(1);
d = params_out5(2);

fit5  = feval(@model,params_out5,V_fit,ph_par,ref5,vargin5,handles);
pars5 = [a b c d g h Vsc f];

%Plot if in debug mode
if (handles.debug == 1)
    figure(10), plot(Vb,I,'.',V_fit,fit5,'red');
    grid on;
    title('Fit nr. 5, Focus on d and c (electron temp.)');
    xlabel('press any key');
    pars5'
    pause
end

%Determine an error
p_in(1) = pars5(1); %a
p_in(2) = pars5(2); %b
p_in(3) = pars5(3); %c
p_in(4) = pars5(4); %d
p_in(5) = pars5(5); %g
p_in(6) = pars5(6); %h
p_in(7) = pars5(7); %Vsc
p_in(8) = pars5(8); %f
v_in    = [];
reference = 18;

fit_e = feval(@model,p_in,V_1,ph_par,reference,v_in,handles);
err   = errorest(I_1,fit_e');  %The error is calculated

%Set the output
fits_out(1,1:nr_points)     = V_fit;
fits_out(2,1:nr_points)     = fit5;

par_out(1:4)  = par(1:4); %Iph01,Iph02,Tph1,Tph2
par_out(5:12) = pars5;    %The parameters
par_out(13)   = par(13);  %z is unaltered
par_out(14)   = err;      %The error
```

B.4.11 model.m

```
%MODEL calculates the model value given the different parameters
%
%   I = model(params,Vb,ph_par,ref,vargin,handles)
%
%   params are the variable parameters while vargin are the ones kept
%   fixed, a reference (ref) is used to choose which parameters to keep
%   fixed according to the following:
%
%   1, params = [a Vsc],   vargin = [b d] (ram,plasma)
%   2, params = [a Vsc f], vargin = [b d] (ram,plasma,photo)
%   3, params = [
%
%   ph_par is a vector containing the photoelectron parameters and should
%   be [Iph01 Iph02 Tph1 Tph2]
%
%   handles is a structure containing information about what currents to
%   use in the fit (eg. handles.e_plasma = 1, handles.e_photo = 0 etc.)

function I = model(params,Vb,ph_par,ref,vargin,handles)

I     = 0;            %clear

Iph   = 0;
Ii    = 0;
Ie_p  = 0;
Ie_sc = 0;

%The parameters are set
if (ref == 1)
    a   = params(1);
    Vsc = params(2);
    b   = vargin(1);
    c   = vargin(2);
    d   = vargin(3);
    f = 0;
    g = 0;
    h = 0;
end
if (ref == 2)
    a   = params(1);
    Vsc = params(2);
    f   = params(3);
    b   = vargin(1);
    c   = vargin(2);
    d   = vargin(3);

    g = 0;
    h = 0;
end
if (ref == 3)
    a   = params(1);
    Vsc = params(2);
    b   = vargin(1);
    c   = vargin(2);
    d   = vargin(3);
    g   = vargin(4);
    h   = vargin(5);

    f = 0;
end
if (ref == 4)
    a   = params(1);
    Vsc = params(2);
    f   = params(3);
    b   = vargin(1);
    c   = vargin(2);
    d   = vargin(3);
    g   = vargin(4);
    h   = vargin(5);
end
if (ref == 5)
    c   = params(1);
    d   = params(2);
    Vsc = params(3);
    a   = vargin(1);
    b   = vargin(2);

    g = 0;
    h = 0;
    f = 0;
end
if (ref == 6)
    c   = params(1);
    d   = params(2);
    Vsc = params(3);
    f   = params(4);
    a   = vargin(1);
    b   = vargin(2);

    g = 0;
    h = 0;
end
if (ref == 7)
    c   = params(1);
    d   = params(2);
    g   = params(3);
    h   = params(4);
    Vsc = params(5);
    a   = vargin(1);
    b   = vargin(2);

    f = 0;
end
if (ref == 8)
    c   = params(1);
    d   = params(2);
    g   = params(3);
    h   = params(4);
    Vsc = params(5);
    f   = params(6);
    a   = vargin(1);
    b   = vargin(2);
end
if (ref == 9)
    a   = params(1);
    c   = params(2);
    d   = params(3);
    Vsc = params(4);
    b   = vargin(1);

    f = 0;
    g = 0;
    h = 0;
end
if (ref == 10)
    a   = params(1);
    c   = params(2);
    d   = params(3);
    Vsc = params(4);
    f   = params(5);
    b   = vargin(1);

    g = 0;
    h = 0;
end
if (ref == 11)
    a   = params(1);
    c   = params(2);
    d   = params(3);
    g   = params(4);
    h   = params(5);
    Vsc = params(6);
    b   = vargin(1);

    f = 0;
end
if (ref == 12)
    a   = params(1);
    c   = params(2);
    d   = params(3);
    g   = params(4);
    h   = params(5);
    Vsc = params(6);
    f   = params(7);
    b   = vargin(1);
end
if (ref == 13)
```

```
    Vsc = params(1);
    a   = vargin(1);
    b   = vargin(2);
    d   = vargin(3);
    g   = vargin(4);
    h   = vargin(5);
    f   = vargin(6);
end
if (ref == 14)
    a   = params(1);
    b   = params(2);
    c   = params(3);
    d   = params(4);
    Vsc = params(5);

    g = 0;
    h = 0;
    f = 0;
end
if (ref == 15)
    a   = params(1);
    b   = params(2);
    c   = params(3);
    d   = params(4);
    Vsc = params(5);
    f   = params(6);

    g = 0;
    h = 0;
end
if (ref == 16)
    a   = params(1);
    b   = params(2);
    c   = params(3);
    d   = params(4);
    Vsc = params(5);
    g   = vargin(1);
    h   = vargin(2);

    f = 0;
end
if (ref == 17)
    a   = params(1);
    b   = params(2);
    c   = params(3);
    d   = params(4);
    Vsc = params(5);
    f   = params(6);
    g   = vargin(1);
    h   = vargin(2);
end
if (ref == 18)              %Good for plotting
    a   = params(1);
    b   = params(2);
    c   = params(3);
    d   = params(4);
    g   = params(5);
    h   = params(6);
    Vsc = params(7);
    f   = params(8);
end
if (ref == 19)
```

```
    c   = params(1);
    d   = params(2);
    a   = vargin(1);
    b   = vargin(2);
    Vsc = vargin(3);

    g = 0;
    h = 0;
    f = 0;
end
if (ref == 20)
    c   = params(1);
    d   = params(2);
    a   = vargin(1);
    b   = vargin(2);
    Vsc = vargin(3);
    f   = vargin(4);

    g = 0;
    h = 0;
end
if (ref == 21)
    c   = params(1);
    d   = params(2);
    a   = vargin(1);
    b   = vargin(2);
    g   = vargin(3);
    h   = vargin(4);
    Vsc = vargin(5);

    f = 0;
end
if (ref == 22)
    c   = params(1);
    d   = params(2);
    a   = vargin(1);
    b   = vargin(2);
    g   = vargin(3);
    h   = vargin(4);
    Vsc = vargin(5);
    f   = vargin(6);
end
if (ref == 23)
    c   = params(1);
    d   = params(2);
    f   = params(3);
    a   = vargin(1);
    b   = vargin(2);
    g   = vargin(3);
    h   = vargin(4);
    Vsc = vargin(5);
end

Iph01 = ph_par(1);
Iph02 = ph_par(2);
Tph1  = ph_par(3);
Tph2  = ph_par(4);

Vp = Vb + Vsc;      %Probe potential

%Setting indices for the intervals
ind1 = find(Vp < 0);
```

```
ind2 = find(Vp >= 0);
ind3 = find(Vb < 0);
ind4 = find(Vb >= 0);
lVp  = length(Vp);

%Calculating the constituent currents

if (handles.i_ram == 1)                         %Ram ions
    Ii(1,1:lVp) = (a+b*Vp-abs(a+b*Vp))/2;
else                                            %Thermal ions
%Ii = zeros(size(Vp))';
    Ii(ind1) = a+b.*Vp(ind1);
    Ii(ind2) = a*exp((b/a)*Vp(ind2));
end

if (handles.e_photo == 1)                       %Photoelectrons
    Iph(ind1) = f*(Iph01+Iph02);
    Iph(ind2) = f*(Iph01*(1+Vp(ind2)/Tph1).*exp(-Vp(ind2)/Tph1)+...
                   Iph02*(1+Vp(ind2)/Tph2).*exp(-Vp(ind2)/Tph2));
end

if (handles.e_plasma == 1)                      %Plasma electrons
    Ie_p(ind1) = c.*exp(d.*Vp(ind1));
    Ie_p(ind2) = c.*(1+d.*Vp(ind2));
end

if (handles.e_sc == 1)                          %Spacecraft photoelectrons
    Ie_sc(ind3) = g.*exp(h.*Vb(ind3));
    Ie_sc(ind4) = g.*(1+h.*Vb(ind4));
end

%Calculate the output current
I = Ii+Iph+Ie_p+Ie_sc;

if ref == 23
    I = I';
end
%Logarithmical weighting?
% if (ref == 19 || ref == 20 || ref == 21 || ref == 22)
%     I = log(abs(I'));
% end
```

B.4.12 compute_c.m

```
%COMPUTE_C determines a value for the variable c, given the other
%          parameters
%
%   c = compute_c(Vf,a,b,d,g,h,f,Vsc,ph_par,handles)
%
%   ph_par = [Iph01 Iph02 Tph1 Tph2]

function c = compute_c(Vf,a,b,d,g,h,f,Vsc,ph_par,handles)

c     = [];
Iph   = 0;
Ii    = 0;
Ie_p  = 0;
Ie_sc = 0;
```

```
%Set up the photoelectron parameters
Iph01 = ph_par(1);
Iph02 = ph_par(2);
Tph1  = ph_par(3);
Tph2  = ph_par(4);
z = Vf-Vsc;

%Determine the different currents, provided that they should be used

if (handles.e_photo == 1)       %Photo-electron current
    if (Vf <= 0)
        Iph = f*(Iph01+Iph02);
    else
        Iph = f*(Iph01*(1+Vf/Tph1)*exp(-Vf/Tph1)+...
                 Iph02*(1+Vf/Tph2)*exp(-Vf/Tph2));
    end
end

if (handles.i_ram == 1)         %Ram-ions
    if (Vf <= -(a/b))
        Ii = a+b*Vf;
    else
        Ii = 0;
    end
else                            %Thermal ions
    if (Vf <= 0)
        Ii = a+b*Vf;
    else
        Ii = a*exp((b/a)*Vf);
    end
end

if (handles.e_plasma == 1)      %Plasma electrons
    if (Vf <= 0)
        Ie_p = exp(d*Vf);
    else
        Ie_p = (1+d*Vf);
    end
end

if (handles.e_sc == 1)          %Spacecraft photoelectrons
    if (z <= 0)
        Ie_sc = g*exp(h*z);
    else
        Ie_sc = g*(1+h*z);
    end
end

%Now the parameter c can be calculated
c = -(Iph+Ii+Ie_sc)/(Ie_p);

if (c < 0)
    c = 0;
end
```

B.4.13 weighting.m

```
%WEIGHTING weighs the data in the interval around V_weight by the amount
%          specified by the user
```

```
%
%   [Vw,Iw] = weighting(V,I,V_weight,interval,weight)

function [Vw,Iw] = weighting(V,I,V_weight,interval,weight)

Vw = [];
Iw = [];

if (length(V) == length(I))
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Weighing the data in an interval close to Vp = 0%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    i1          = find(V > V_weight-interval,1);
    i2          = find(V < V_weight+interval,1,'last');
    Vw(1:i1-1) = V(1:i1-1);
    Iw(1:i1-1) = I(1:i1-1);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Here the weighing is performed%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    k = i1;
    for i = i1:i2
        Vw(k:k+weight) = V(i);
        Iw(k:k+weight) = I(i);
        k = k+weight+1;
    end
    len            = length(V(i2+1:length(V)));
    Vw(k:k+len-1) = V(i2+1:length(V));
    Iw(k:k+len-1) = I(i2+1:length(I));
else
    disp('I and V are not the same length!');
end
```

B.4.14 choose_fit.m

```
%CHOOSE_FIT compares the different fits given and returns the one that fits
%           the data best
%
%   [fit,param] = choose_fit(Vb,I,fits,pars,res)
%
%   fits should be a matrix containing one fit (i.e. current) on each row
%
%   pars should be a matrix containing the parameters on each row
%
%   res should be a row or column vector containing the resnorms (errors)
%   of the fits, if this isn't included the routine will perform a least
%   square evaluation of the data.

function [fit,param] = choose_fit(Vb,I,fits,pars,res)

stl = size(fits,1);                     %Test how many?
[row1,col1] = find(imag(pars) ~= 0);    %complex automatically removed
if length(row1 >= 1)
    pars(row1,:) = 0;
    fits(row1,:) = 0;
    if (exist('res'))
        res(row1)    = inf;
    end
end
go = 1;
```

```
if (res ~= 0)
    while (go == 1)
        ind   = find(res == min(res)); %extracting the fit with least error
        if (length(ind) > 1)
            ind = 1;
            go  = 0;
        else
            if (length(pars(1,:)) == 9)
                if ((pars(ind,1) > 0) || (pars(ind,3) < 0));
                    res(ind) = inf;
                else
                    go = 0;
                end
            else
                if ((pars(ind,5) > 0) || (pars(ind,7) < 0));
                    res(ind) = inf;
                else
                    go = 0;
                end
            end
        end
    end
else
    for i = 1:stl
        err(i) = errorest(I,fits(i,:));
    end
    ind = find(err == min(err));
end

fit   = fits(ind,:);
param = pars(ind,:);
```

B.4.15 errorest.m

```
%ERROREST calculates the root mean square norm of the theoretical values
%         to the data and returns it
%
%   err = errorest(Idat,Icalc)
function err = errorest(Idat,Icalc)

err = sum((Idat-Icalc).^2);
err = sqrt(err)/length(Idat);
```

B.4.16 compare_fits.m

```
%COMPARE_FITS compares the different fits given and chooses what fit to use
%             for the real data, returning its parameters and fits.
%
%   [par,ppar,fit,currents] = compare_fits(Vb,I,un,data,epop,handles)
%
%   Data is a structure containing both the fits and the parameters in the
%   following form: Data.par.a, Data.fit.a, Data.par.b, Data.fit.b etc.
%
%   vsc is used for the conversion to physical parameters, being the
%   spacecraft speed
%
```

```
%   epop = 1 means one electron population, epop = 2 means two

function [par,ppar,fit,currents] = compare_fits(Vb,I,data,epop,handles)

%Set up the fits and errors
err = [1e8 1e8];                    %This is in case only one is used
if (handles.fit_partial == 1)
    if (epop == 1)
        fit1   = data.fit.a(1:2,:);
        err(1) = data.par.a(12);
    else
        fit1   = data.fit.a(1:2,:);
        err(1) = data.par.a(14);
    end
end
if (handles.fit_all == 1)
    if (epop == 1)
        fit2   = data.fit.b(1:2,:);
        err(2) = data.par.b(12);
    else
        fit2   = data.fit.b(1:2,:);
        err(2) = data.par.b(14);
    end
end

%compare the errors and
choice = find(min(err) == err);

%Plotting the fits
%figure,plot(Vb,I,'.',fit1(1,:),fit1(2,:),'red');
%title('Partial data set used')
%figure,plot(Vb,I,'.',fit2(1,:),fit2(2,:),'red');
%title('All data used')

if length(choice > 1)           %Can have same error
    choice = choice(1);
end

%Minimum error is choosen
switch choice
    case 1
        par  = data.par.a;
        fit  = fit1;
    case 2
        par  = data.par.b;
        fit  = fit2;
end

%Setting the physical parameters and the constituent currents
V = linspace(min(Vb),max(Vb),1000);
if (epop == 1)
    ppar         = par2phys2(par,handles,1)
    [Iph,Ii,Ie]  = get_I_parts(par,V,handles);
else
    ppar            = par2phys2(par,handles,2);
    [Iph,Ii,Ie,Ie2] = get_I_parts(par,V,handles);
end

%Finally the output which isn't already set are now set
currents{1} = Iph;
currents{2} = Ii;
currents{3} = Ie;
```

```
if (epop == 2)
    currents{4} = Ie2;
end
```

B.4.17 par2phys2.m

```
%PAR2PHYS converts the parameters used in the sweep to physical units
%
%   [phys_par] = par2phys(par,handles,ref)
%
%   The reference indicates one or two electron populations

function phys_par = par2phys(par,handles,ref)

phys_par = [];

Ap = 0.0078539816;      %Probe area          [m^2]
rp = 0.025;             %Probe radius        [m]
qe = 1.60217733e-19;    %elementary charge   [C]
%kB = 8.617385e-5;      %Boltzmanns constant [eV/K]
kB = 1.380658e-23;      %Boltzmanns constant [J/K]
me = 9.1093897e-31;     %electron mass       [kg]

%setting s/c speed and ion mass
mi  = handles.ion_mass;
vsc = handles.vsc;

Tph1  = par(3)*11600;
Tph2  = par(4)*11600;
Iph01 = -par(1)*par(12);
Iph02 = -par(2)*par(12);
Ii0   = -par(5);
Ti    = -par(5)/par(6);
Ie01  = par(7);
Te1   = 1/par(8)*11600;
Ie02  = par(9);
Te2   = 1/par(10)*11600;
Vsc   = par(11);
f     = par(12);
z     = par(13);




if (handles.fit_try_all == 1)
    switch par(15);     %need to determine which model was choosen
        case 1
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 2
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 3
```

```
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 4
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 5
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
        case 6
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
        case 7
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
        case 8
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
    end
end

%Determine the densities
n_e1 = Ie01/(Ap*qe*sqrt((kB*Te1)/(2*pi*me)));  %density of electrons 1
n_e2 = Ie02/(Ap*qe*sqrt((kB*Te2)/(2*pi*me)));  %density of electrons 2
if (handles.fit_free_densities == 1)
    n_i  = Ii0/(Ap*qe*sqrt((kB*Ti*11600)/(2*pi*mi)));
end
%If we have quasineutrality, ni = ne, so it can be determined

if (handles.fit_quasineutral == 1)
    n_i = n_e1;
end

if (handles.i_ram == 1)                 %ram ions
    mi  = Ti*2*qe/(vsc^2);
else                                    %thermal ions
    mi = (Ap*qe*n_i/Ii0)^2*(kB*Ti*11600)/(2*pi);
end




phys_par(1)  = Vsc;
%phys_par(2)  = e;
phys_par(3)  = f;
phys_par(4)  = z;
phys_par(5)  = Iph01;
phys_par(6)  = Iph02;
phys_par(7)  = Tph1;
phys_par(8)  = Tph2;
phys_par(9)  = n_i;
phys_par(10) = Ti;
phys_par(11) = mi;
phys_par(12) = n_e1;
phys_par(14) = n_e2;
phys_par(13) = Te1;
phys_par(15) = Te2;
phys_par(16) = Ie02;
if (handles.fit_try_all == 1)
    phys_par(17) = par(15);
else
    phys_par(17) = 0;
end
```

B.4.18 get_I_parts.m

```
%GET_I_PARTS provides the different currents that add up to the sweep
%
%   [varargout] = get_I_parts(params,Vb,handles)
%   where params is a vector containing the parameters Iph01,Iph02,Tph1,
%   Tph2,a,b,c,d,Vsc,f,z,g,h, in that order
%
%   The function can be called with three or four output arguments:
%
%   Iph,Ii,Ie
%   Iph,Ii,Ie,Ie2

function [varargout] = get_I_parts(params,Vb,handles)

Iph   = zeros(size(Vb));
Ii    = zeros(size(Vb));
Ie    = zeros(size(Vb));
Ie2   = zeros(size(Vb));
Iph_2 = zeros(size(Vb));
Ii_2  = zeros(size(Vb));
Ie1_2 = zeros(size(Vb));
Ie2_2 = zeros(size(Vb));

%Set up the parameters
Iph01 = params(1);
Iph02 = params(2);
Tph1  = params(3);
Tph2  = params(4);
a     = params(5);
b     = params(6);
c     = params(7);
d     = params(8);
g     = params(9);
h     = params(10);
Vsc   = params(11);
f     = params(12);
z     = params(13);
if (handles.fit_try_all == 1)
    switch params(15);     %need to determine which model was choosen
```

```
        case 1
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 2
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 3
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 4
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 0;        %no sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 1;        %fit partial data (one e-pop)
        case 5
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
        case 6
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 1;        %ram ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
        case 7
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 0;        %no photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
        case 8
            handles.e_plasma    = 1;        %plasmaelectrons
            handles.e_sc        = 1;        %sc-electrons
            handles.e_photo     = 1;        %photoelectrons
            handles.i_ram       = 0;        %thermal ions
            handles.fit_partial = 0;        %fit all data points (two e-pops)
    end
end
%Determine the currents
Vp = Vb + Vsc;

ind1 = find(Vp < 0);
ind2 = find(Vp >= 0);
ind3 = find(Vb < 0);
ind4 = find(Vb >= 0);
lVp  = length(Vp);

if (handles.i_ram == 1)
    Ii(1,1:lVp)   = (a+b.*Vp-abs(a+b.*Vp))/2;
end
```

```
if (handles.i_therm == 1)
    Ii(ind1) = a+b.*Vp(ind1);
    Ii(ind2) = a.*exp((b/a).*Vp(ind2));
end
if (handles.e_photo == 1)
    Iph(ind1)     = f.*(Iph01+Iph02);
    Iph(ind2)     = f.*(Iph01.*(1+Vp(ind2)./Tph1).*exp(-Vp(ind2)./Tph1)+...
                    Iph02.*(1+Vp(ind2)./Tph2).*exp(-Vp(ind2)./Tph2));
end
if (handles.e_plasma == 1)
    Ie(ind1)      = c.*exp(d.*Vp(ind1));
    Ie(ind2)      = c.*(1+d.*Vp(ind2));
end
if (handles.e_sc == 1)
    Ie2(ind3)     = g.*exp(h.*Vb(ind3));
    Ie2(ind4)     = g.*(1+h.*Vb(ind4));
end

%setting the output
if (nargout == 3)
    varargout = {Iph Ii Ie};
end
if (nargout == 4)
    varargout = {Iph Ii Ie Ie2};
end
```