LATMOS

Hyb 3D. Technical report and developer notes

version 1.0

Author:
Ronan Modolo
Université de Versailles
Saint-Quentin
LATMOS
ronan.modolo@latmos.ipsl.fr

Co-Author:
Marco Mancini
Observatoire de Paris-Meudon
LUTH
Marco.Mancini@obspm.fr

Co-Author:
Sebastien Hess
LATMOS
sebastien.Hess@latmos.ipsl.fr

January 6, 2015

Contents

Abstract

This document contains the description of the **hyb_3D** code: its philosophy, its routines and the manual to implements new environments. Information concerning the submission process on the CICLAD platform is also developped as well as diagnostic and grid description

Chapter 1

The Hybrid formalism

There are many theoretical approaches used to study the plasma interaction with solar system objects. A complete description of phenomena occurring at electron and ion scales (spatial and temporal) require a fully kinetic model in which each species "s" is described by a distribution function $f_s(\mathbf{x}, \mathbf{v}, t)$. The evolution of the distribution function is governed by the Vlasov equation, which includes the electromagnetic field determined self-consistently from the dynamic of all the particles via the Maxwell equations. Three-dimensional fully kinetic models are generally not viable to study the whole physical system of an ionized environment of a moon or a planet, mainly due to computational resources limitations, and are usually restricted to study local processes. To investigate large-scale phenomena it is therefore required to simplify the model of the plasma by dropping out some details of its behavior. The usual procedure to decrease the complexity of the model consist of a substitution of the Vlasov equation of species "s" by a hierarchy of evolution of moment's equation of the distribution function integrated over velocity space. The hierarchy of the moment's equation is truncated at a given order with a closure equation, which couples the last retained moment to inferior moment order. When the reduction procedure is applied to all species of the plasma we obtain a multi-fluid model. The simplest model is an ideal MHD model which describes the plasma as a single non-resistive fluid coupled to a magnetic induction equation. An intermediate approach, between fully kinetic and MHD models, consist of representing the electron as a fluid while ions are modeled by numerical particles leading to a kinetic description of ions behavior. Such an approach, although more binding in term of computational resources, is based on less assumption and includes more physics.

1.1 Hybrid formalism

Hybrid models are able to describe physical phenomena at ion scales, which is not viable with the MHD formalism. In a hybrid model, electrons are treated as an inertia less fluid, contributing to electric current and enforcing the neutrality of the plasma, while a kinetic description is adopted for the ions. Ions are therefore represented by individual particles, called macro-particles. A macro-particle does not represent one ion but a cloud of ions with a given density and with the same properties (ions with the same charge, mass, velocity). The position and velocity of a macro-particle "j" obey to the laws of motion of physical particles:

$$\frac{d\mathbf{x}_j}{dt} = \mathbf{v}_j \tag{1.1}$$

$$m_j \frac{d\mathbf{v}_j}{dt} = q_j(\mathbf{E} + \mathbf{v}_j \times \mathbf{B}) \tag{1.2}$$

where q_j and m_j are the charge and the mass of the particle "j" and $\mathbf{x}_j$ and $\mathbf{v}_j$ the position and velocity of the particle. This set of macro-particles defines macroscopic moments as follows.

The density n_s of particle species "s" is determined from the distribution function

$$n_s(\mathbf{x},t) = \int f_s(\mathbf{x},\mathbf{v},t)d^3v \tag{1.3}$$

The bulk velocity and the ionic current of the species is therefore

$$\mathbf{U_s}(\mathbf{x},t) = \frac{1}{n_s(\mathbf{x},t)}\int \mathbf{v} f_s(\mathbf{x},\mathbf{v},t)d^3v \tag{1.4}$$

$$\mathbf{J_s}(\mathbf{x},t) = q_s n_s(\mathbf{x},t)\mathbf{U_s}(\mathbf{x},t) \tag{1.5}$$

The total ionic current corresponds to the sum of ionic current of all of ion species ($\mathbf{J_i} = \sum_s \mathbf{J_s}(\mathbf{x},t)$). The assumption of a mass less and charge neutralizing fluid, for the description of electrons, leading to $n_e = \sum_s n_s = \rho$, implies that the electron plasma oscillations and electron inertial lengths cannot be described. Hence the electric field is a function of state computed from the electron momentum equation

$$\mathbf{E} = -\frac{\mathbf{J_i}\times\mathbf{B}}{\rho} + \frac{(\nabla\times\mathbf{B})\times\mathbf{B}}{\mu_0\rho} - \frac{\nabla p_e}{\rho} \tag{1.6}$$

The time evolution of the magnetic field results from Faraday's equation, meanwhile satisfying the solenoidal condition:

$$\frac{\partial\mathbf{B}}{\partial t} = -\nabla\times\mathbf{E} \tag{1.7}$$

$$\nabla\cdot\mathbf{B} = 0 \tag{1.8}$$

E and **B** are computed on two identical grids shifted each other by half a grid size in all directions and ensure a divergence free magnetic field as a second-order approximation. Like for MHD models a closure equation is required. Depending of the models, different assumptions are made on the electron temperature, such an adiabatic behavior, or more elaborated model can also solve an electron temperature equation.

Different schemes exist to solve the huge system of coupled equations 1.1 to 1.8 governing the motion of particles and the evolution of the electromagnetic field.

1.2 Numerical scheme

Hybrid codes are numerical models which resolve the temporal evolution of the plasma described by a set of discrete equations equivalent to a Vlasov system. Most of hybrid codes developped for space physics are based on two main algorithms: the [**?**] scheme and the [**?**] scheme.

We use two shifted grid by half a spatial step (in each direction). This is a common aspect in hybrid codes and more generally in electromagnetic codes. The magnetic field, the density, the electron pressure, ionic currents and other momentums are computed on the top of the cell of the first grid, that we will call the B-grid, in reference to the magnetic field. The electric field is defined on the top of the cell of the other grid, that we will call the E-grid. As it has been mentionned above, such trick allow an optimal computation of rotational and gradients that appear in Maxwell equations 1.1. $\nabla\times\mathbf{E}$ is located at the same position than **B** and is accurate to the second order inΔx. Reciprocally $\nabla\times\mathbf{B}$, $\mathbf{J}\times\mathbf{B}$ and ∇P_e are computed on the E-grid.

Ions are described by a set of macroparticles. Each macroparticle has the volume and the shape of a numerical cell. The position of the macroparticle is identified by its center position. Charge density and ionic current at a given grid point (k) are defined by the position and the speed of the macroparticles via the following formulas :

$$\rho_k = \sum_s \phi_k(\mathbf{x}_s)q_s \tag{1.9}$$

$$\mathbf{J}_k = \sum_s \phi_k(\mathbf{x}_s)q_s\mathbf{v}_s \tag{1.10}$$

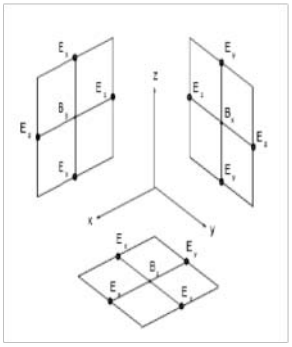

Figure 1.1: Conjugate grid E and B

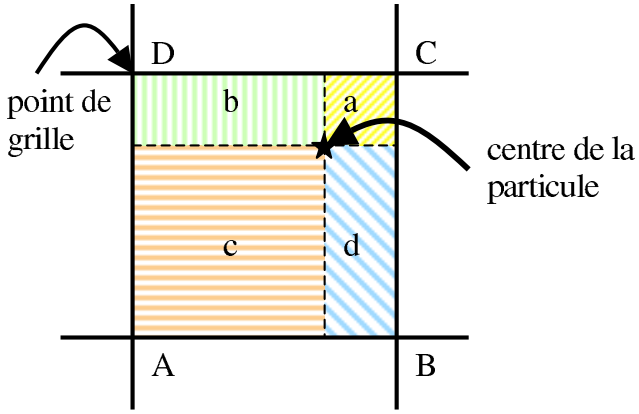

Figure 1.2: Schematical representation of the attribution of contribution of particle 'weight to the neighboured grid points. The surface are assigned to grid points, *i.e.* the surface a contribute to the grid pint A, and so on.

$\phi(\mathbf{x}_k)$ represent the contribution of the macroparticle s, with its center lovcalized in $\mathbf{x}_s$, to the grid point k. We can detail here the 2D case or repartition of weight of the macroparticle to the grid points but it can easily be extended to the 3D configuration. Figure 1.2 shows schematic contribution repartition of the macroparticle weight to density and currents on the grid points which are the closest grid points of the particle center: ech grid point receives a fraction of the weight of each macroparticle in the cell. The contribution are weighted by the function $\phi_{ks} = \phi(\mathbf{x}_k, \mathbf{x}_s)$ which is dependent of the position $\mathbf{x}_s$ and the grid point $\mathbf{x}_k$.

The fraction of the total charge of the macroparticle assigned to the grid point k are proportional to the surface $\phi_{ks} = \phi_k(\vec{x}_s)$ (Fig. 1.2). Functions ϕ_{ks} are such that $\sum_k = \phi_{ks} = 1$ (each particle have a weight equal to identity). A linear interpolation in 3D are used to evaluate the magnetic and electric field aplied to the macroparticle in function of the fields which are defined at the grid pointswhich are the closest neighbours of the particle center.

$$\mathbf{E}(\mathbf{x}_s) = \sum_k \phi_k(\mathbf{x}_s)\mathbf{E}_k \tag{1.11}$$

To interpolate $\mathbf{E}$ and $\mathbf{B}$ we should use the same functions ϕ_{ks} to compute the charge density and the ionic current in order to avoid articficial forces exerced by the grid on the particles [**?**].

A leap-frog scheme is used to advance the particle from time step $n-1/2$ to $n+1/2$ with equation 1.1 and determining the velocity with equation 1.2 at time step n. We introduce the following notation where exponent refers to the time step:

$$x^n = x(t_n) = x(t_0 + n\Delta t) \tag{1.12}$$

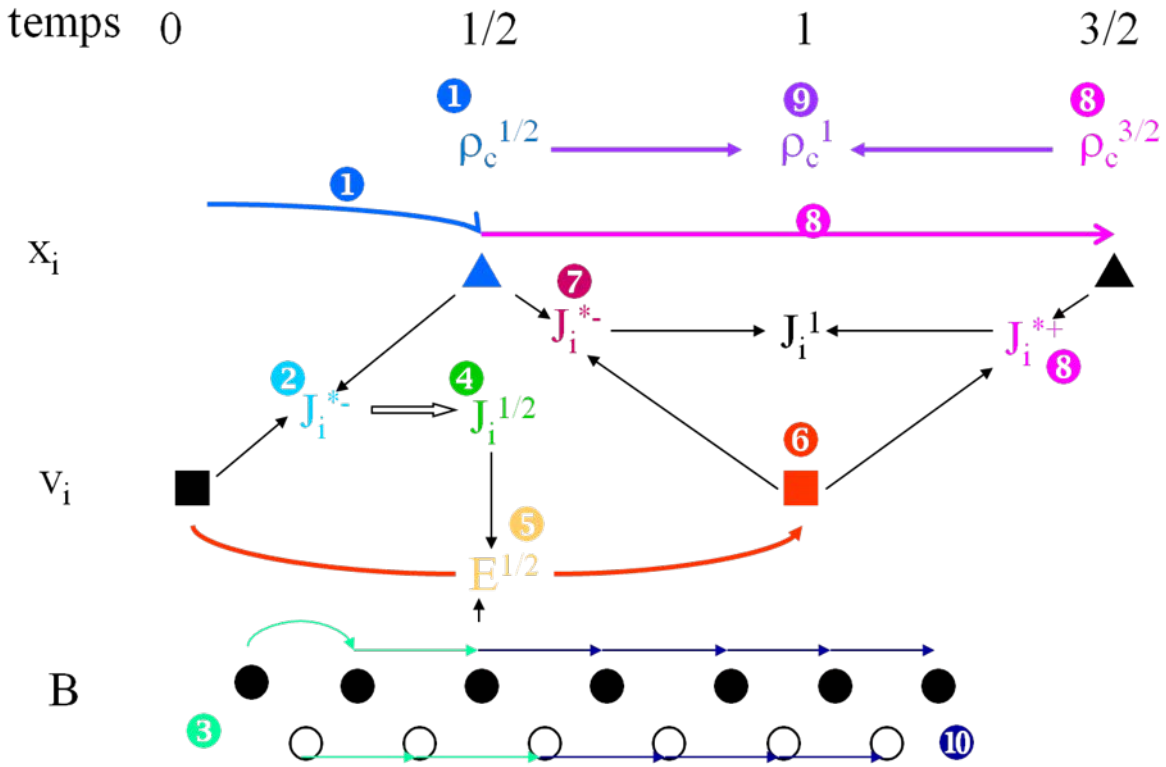

Figure 1.3: Diagram of temporal evolution of the CAM CL algorithm [**?**].

The differential equation1.1-1.2 are discretized by a centerd finite diffrenece scheme accurate up to the second order in Δt.

$$\mathbf{x}^{n+1/2} = \mathbf{x}^{n-1/2} + \mathbf{v}^n \Delta t \tag{1.13}$$

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \frac{q}{m} \left(\mathbf{E}^{n+1/2}(\mathbf{x}^{n+1/2}) + \mathbf{v}^{n+1/2} \times \mathbf{B}^{n+1/2}(\mathbf{x}^{n+1/2}) \right) \tag{1.14}$$

The problem in equation 1.14 is that neither $\mathbf{E}^{n+1/2}$ nor $\mathbf{B}^{n+1/2}$ are known at the beginning of the time step. The [**?**] algorithm solved this problem by computing in time the ionic current of hal a time step with appropriate equations.

Figure **??** presents the temporal scheme evolution of tha algorithm [**?**].

Chapter 2

Few words on the code parallelization

Sequential simulation models, *i.e.* which are executed on a single processor and treating instructions one by one, are the simplest in term of programming effort but present few drawbacks which can be penalizing when we look for modeling small spatial scales or if we wish to make some parametric studies which require to re-run the simulation several times. Performances of these sequential models are linked to the available computational capabilities in term of CPU clock. For kinetic simulations, hybrid or full particle-in-cell, the number of numerical particles are a dimensionning factor. It affects the memory requirements for the simulation. The sequential hybrid model requires about 4Gb memory to handle about 50 million particles. If one wishes to improve the spatial resolution while keeping the same physical size of the simulation box, we will increase the number of numerical particles to manage, the number of particles is proportional (to first order) the number of cells . We quickly reach the limits of available resources, that is to say, the memory associated with a processor.

Add to that a computation time result (which can reach ten days), and also increases with the number of particles, we confront directly the limitations of sequential models. To overcome these constraints and to better optimize the simulation model, a parallelization effort was undertaken. Parallelization is to modify the simulation program to make it run on multiple processors or cores simultaneously. There are various methods of parallelism and the choice of method is guided by the problem that the physical address is desired, the investment and the developer of the most important characteristics of the computing platform available or contemplated. The best known are: the OpenMP parallelization, which is a multi-task parallelization for shared memory machines (presence of many cores on a single node / machine), the parallelization on graphics card (using coprocessors on GPU graphics cards to deport Calculation of the node), and parallelization of message exchanges for distributed memory machines (cluster servers / nodes). Parallelization technique used is based on exchange protocols ("message-passing") and its implementation is based on the standard "Message Passing Interface" (MPI). MPI is a multi-processor model whose mode of communication between processes is explicit (that is to say, the responsibility of the user / developer). The purpose of this Chapter is to briefly explain the parallelization implemented in the simulation model and the reader is referred to the course materials for the method of parallelization and OpenMP graphics card (http://www.idris.fr/data/cours/parallel/openmp/OpenMP_cours.pdf, http://gpu.epfl.ch/cours2.pdf). It is interesting to note that these methods of parallelization can be complementary. The MPI standard is available [**?**] and an excellent summary is offered as support during the Institute of Development and Resources in Scientific Computing. (http://www.idris.fr/data/cours/parallel/mpi/IDRIS_MPI_cours_couleurs.pdf).

2.1 Domain decomposition

The domain decomposition method applies well to distributed memory parallel architectures . It decomposes the simulation domain into subdomains with as many subdomains as process and each process calculations are performed locally . The information found at the interfaces of the subdomains are managed through communications between processes (physical or virtual) sharing these

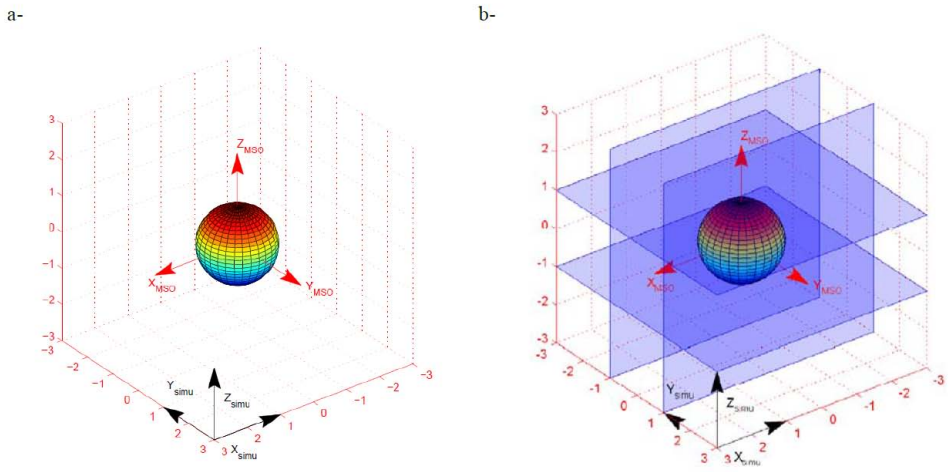

Figure 2.1: Schematic representation of the overall simulation domain (a) and domain decomposition (b). The interfaces between real processes are shown in blue in the figure (b). Vectors and the center of the reference coordinate system of the simulation is shown in black while the MSO coordinate system is represented by vectors in red.

interfaces , the term "neighbor" . This method subdomain is relatively natural and has the advantage of using the best local memory and has many MPI procedures developed . To return to the more specific problem of the hybrid model , the objective is to model the interaction between the incident plasma , the solar wind or magnetospheric plasma , and the environment of the obstacle , Mercury , Mars, Ganymede, Titan .... There is therefore a preferred direction which is the direction of flow of the incident plasma. The coordinate system of the simulation is chosen such that this direction coincides with the axis X of the simulation mark . To minimize the number of communications, it avoids all communications particles flowing in the direction of the incident plasma, domain decomposition will not occur in the X direction The topology of the domain decomposition is two-dimensional . An example of the decomposition of the domain simulation is shown in Figure 2.1 . In this example , the total area was divided into nine equal volume subdomains. The number of sub- field depends on the choice of the user (and the number of processors available for the simulation) .

Note that the decomposition model is developed in the two Cartesian dimensions . However, the volumes of sub- areas may be different and are determined by the user. This decomposition leads to associate each subdomain to a process. Each subdomain manages part of the global mesh and particles . Each process will therefore conduct the calculations in its subdomain , either at field calculations and times for the management of the particles. Each process is identified by a unique identifier, the rank of the processor and its coordinates in the grid processes (Figure 2.2). This information allows to easily determine the neighbors of a given process and determined the physical boundaries of the domain. Processes are grouped into communicators that allow the exchange of messages. As recalled in the previous sections , two types of boundary conditions are used in the Y and Z (perpendicular to the direction of flow of the incident plasma) directions. Fields, times and the incident plasma particles using periodic conditions, against particles representing a planetary species are being applied open conditions (particles pass freely in the field of simulation). These two types of boundary conditions naturally lead to the manipulation of two communicators. These communication thus differ by identifying neighbors of a given process .

Thus, from Figure 2.2, the neighbors in the process number 5 communicator using periodic conditions and in the neighboring communicator using open conditions are shown in Table 2.1.

Domain decomposition allows to distribute the management of different aspects of simulation on several processes, thereby reducing the recovery time results while allowing work on simulations requiring large memory.

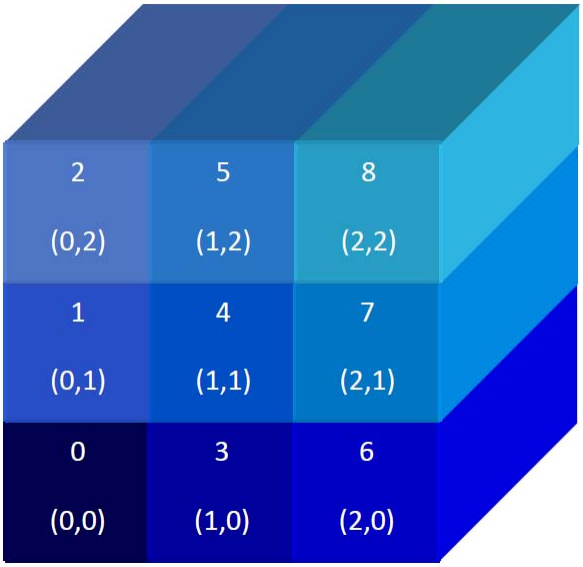

Figure 2.2: rank and coordinated processes in the grid. The overall simulation domain is the union of the subdomains..

neighbor	periodic communicator	open communicator
North	3	MPI_PROC_NULL
North-West	6	MPI_PROC_NULL
West	8	8
South-West	7	7
South	4	4
South-East	1	1
East	2	2
North-East	0	MPI_PROC_NULL

Table 2.1: Determination neighboring process 5 for periodic and open communicators. Assuming the simulation domain was divided into nine sub-domains.

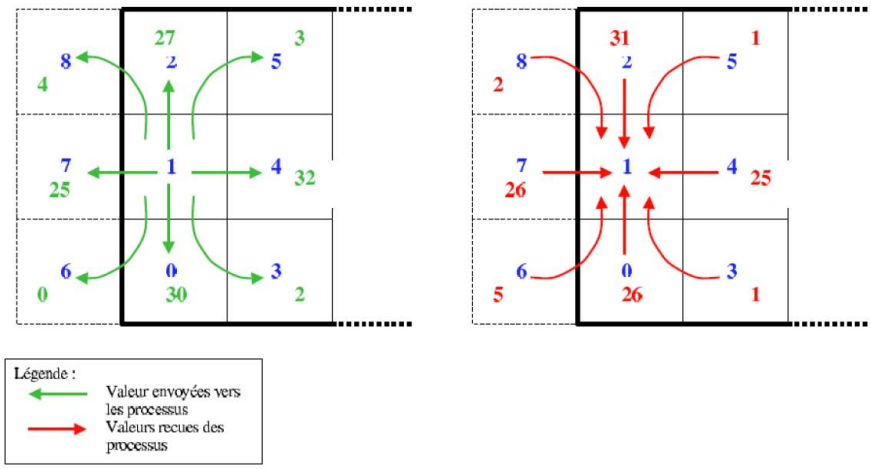

Figure 2.3: Mapping of the sending and receiving of the number of particles of one process to the different neighbors. The total area (including the physical limit is determined by the thick black line) was divided into nine sub-domains. In this example the communicator using periodic boundary conditions was used

2.2 Individual and collective Communications

Individual communications, or point-to-point used to exchange information between two processes while collective communications allow a process to send this information to all processes belonging to the same communicator. Collective communications are used in the simulation model for global simulation information such as total number of particles in the box, or the total magnetic energy of the simulation, ...

The point-to-point represent the vast majority of calls made in the model and are neighborhoods communications, that is to say between a process and its eight neighbors. These communications are involved in the management of the particles and for the management of interfaces between sub-domains grid process.

2.2.1 particle management

At each time step, each process "pushes" the particles of which it manages. Of course some of these particles will leave the physical subdomain. It is necessary to identify the particles that come out of the subdomain to count and determine their destination , that is to say in which physical subdomain they are now advanced . The message exchanges are performed in two steps . A first step , for a given process , to count the particles that will be sent to different neighbors and send this number to the neighbors in question. This step allows you to prepare and size the array size to send. Figure 2.3 shows how this step. Process 1 informsd process 2 that it will receive 27 particles , it informs the process 5 he will receive 3 particles , etc ... In this operation process 1 receives these neighbors the number of particles it will have the management . So the process 5 will send 1 particle to process 1 , etc. ...

The second step is to send the information about the particle (position, speed, load, weight, origin, ...) to the destination process. This information is collected in a table structure created specifically for these items. Each process sends this information to its neighbors and receives their neighbors of the same information. These two steps must be performed separately for representative species of particles incident plasma and plasma species with the appropriate global communicators.

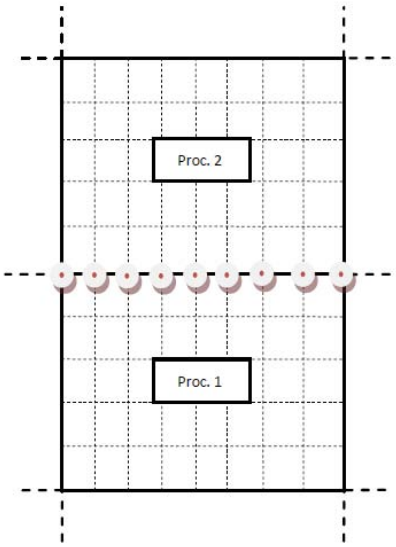

Figure 2.4: Common mesh nodes to process 1 and 2. Subdomains 1 and 2 are represented by the black solid lines while the mesh in each sub-field is indicated by black dotted lines.

2.2.2 Managing the interface between subdomains

Calculations of fields and times are on the mesh nodes. The grid points may be common to several subdomains if located on the interfaces of the gate as the process illustrated in Figure 2.4.

Values that are the interfaces must be identical for different processes sharing these grid points. It is necessary to pay particular attention to the calculation of the moments . Indeed these calculations (density , current , ...) are performed from the information of the particles in the vicinity of the grid point . If the grid point is coincident with an interface, the contribution of particles in the neighboring sub- domain is missing in the calculation. For example when the process 1 performs the calculation of the density of points common with the gate process 2 , the contribution of the process run by particles 2 is missing, and vice versa. It is therefore necessary to add the contribution of neighbors and this process operation is carried out using point- to-point . To calculate the electric field , the parallel model uses , as the sequential model , ghost cells. The values of the electric field at the apices of these phantom cells correspond to the electric field values lying within the sub- fields of the neighboring processes .

2.3 Perfomance

Parallelized simulation code has been tested , validated and is currently used on a server to calculate the IPSL : Ciclad (http://ciclad-web.ipsl.jussieu.fr/accueil/) . The administrator of this machine is Philippe Weill (LATMOS) . Modeling the interaction between an incident plasma and neutral planetary envelopes is supported by several organizations that provided funding for computing machines . These machines are dedicated to the project and financed by the ANR HELISOARES (ANR -09- BLAN -223) , CNES , INSU -CSA and LPP (via Europlanet FP7 program), were added to the render farm Ciclad . Three servers to 32 cores (AMD opteron 6134 and 4 GB of memory per core) and four servers to 64 cores , also with 4GB core , are available for the project. Farm calculation is based on a x86-64 architecture. The calculation servers are interconnected by cables Infiniband . Compilers used for code and installed on Ciclad are the Intel compilers (ifort 2011 version) and Portland (pgfortran 2011 version) . Parallelized simulation code was distributed to LPP and IRAP , two laboratories working on aspects of planetary plasmas. It also works on a machine of the LPP (" Bender " , 32 cores , 4GB by core) .

To determine the quality of the parallelization, it is possible to compare the performance of the

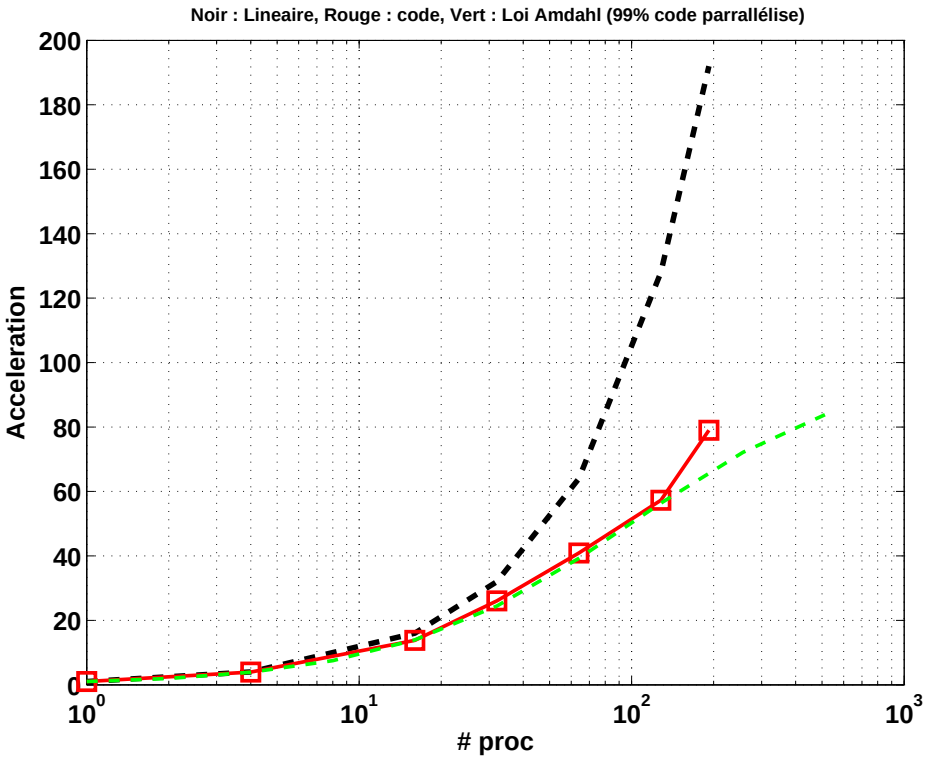

Figure 2.5: Test "speedup", the black curve represents an ideal linear / behavior, features red shows the performances of the code and the green curve shows the perofrmance a parallelized code 99 percent according to Amdahl's law)

simulation model predictions or theoretical behavior. For this we appeal to notions of acceleration (speed-up in English) and efficiency. The acceleration is the ratio between the sequential time and recovery time. The recovery time is the physical time elapsed between the beginning and the end of the run. The acceleration is given by

$$acceleration(p) = \frac{sequential \quad time}{restitution \quad time(p)}$$

where p is the the number of processors used for parallel simulation. In the ideal case (linear), not taking into account communication, acceleration is equal to the number of processors. Amdahl's Law to determine the maximum acceleration of parallel code based on the percentage of time spent in the parts of code that are non-parallelizable and parallelized parties.

$$acceleration_{Amdhal}(p) \leq \frac{1}{\frac{s}{p} + (1-s)}$$

where s is the percentage of the code which is parallelized.

Scale tests (scalability) were performed on the Ciclad machine. This is a control flow modeling a plasma wind in a box dimension of 50x500x500 and during 1000 simulation time step. This simulation requires about 50 GB of memory. Figure 2.5 shows the result of the acceleration test on the code Ciclad machine. Despite a distance from the ideal acceleration (black dotted curve) the behavior of code is relatively good. According to the Amhdal's law, the code shows performances similar to a 99 percent parallelized code.

Chapter 3

The global structure

The structure of the code is such that a minimum number of files should be edited/created by the user. Namely, general parameters for the simulation are defined in defs_parameters.F90, a env_*name*.F90 file must be created in not already existing, and pointers to routines in this file may be set in environment.F90. Such architecture maximize the variety of planetary environment described. The same code is therefore used to model MArs, Titan, Ganymede , Mercury environments as well as the interaction between a magnetic cloud a magnetospheric obstacle. Figure **??** illustrate some results for the different simulated environement.

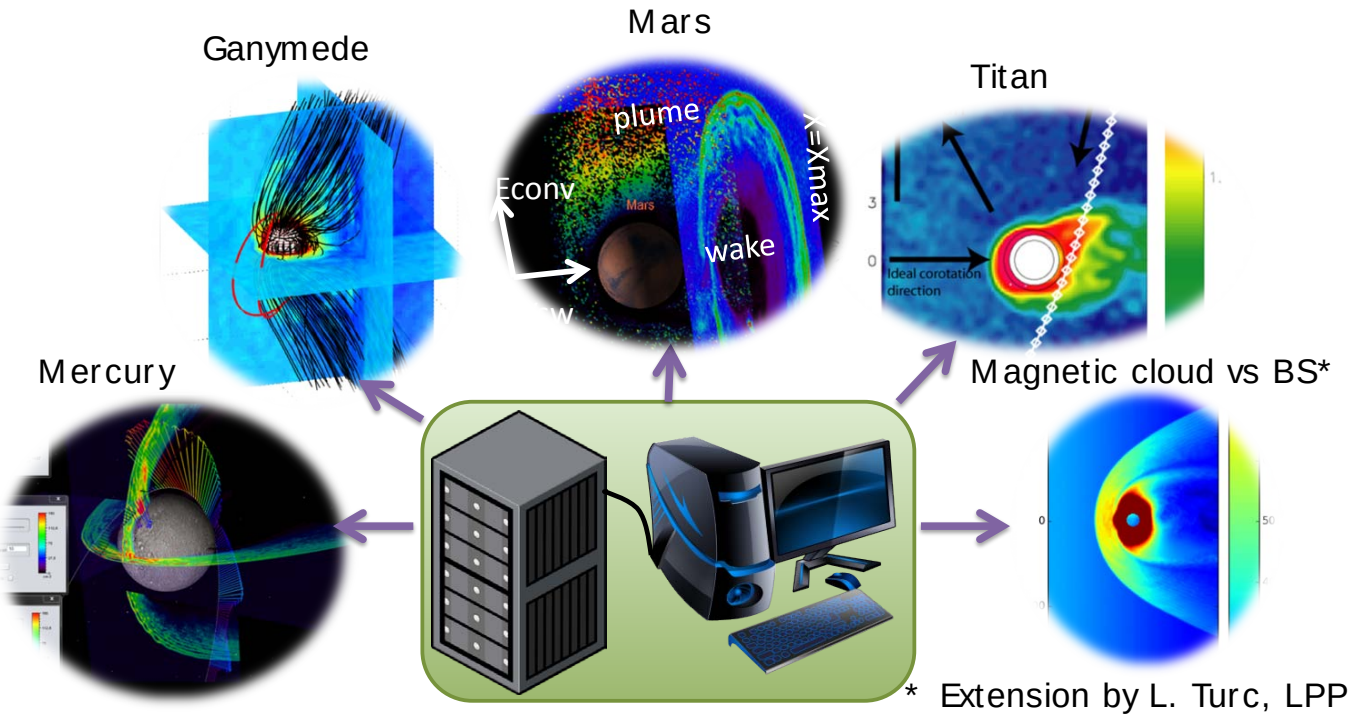

Figure 3.1: Application of the 3D parallelized multi-species hybrid model developped at LATMOS

3.1 Module naming convention

Because of FORTRAN limitations, modules and routines cannot share the same name. This issue was dodged by adding "m_" in front of the module name. However, a better usage could be done of this mandatory prefix. Hence prefixes are now:

- "defs_", if the module mainly contains structures or variable definitions.
- "diag_", if the module is related to diagnostics.
- "env_", if the module contains the environment of a given planet.

• "atm_", for the modules containing the subroutines shared by the planet environment files (photoproduction, ionosphere,...). **atm_ modules should only be called by env_ modules**.
• "field_", if the module is related to fields.
• "particle_", if the module is related to particles.
• "m_", for the micellaneous modules which are directly related to the main program.
All other modules contains mostly general routines which cannot be classified into one of the groups mentioned above. Note that in addition to hyb_3d.F90, which contains the main program, initialisation.F90 and time_schedule.F90, which do the initialization and the iterations have no prefix, as they can be understood as part of the main program. Figure 1 shows a summary of all the modules and the kind (Block) they belong to. Note that this division in blocks is not only related to the physical objects they deal with, but that modules in the same bloc are more closely related (through calls) than modules in different blocs.

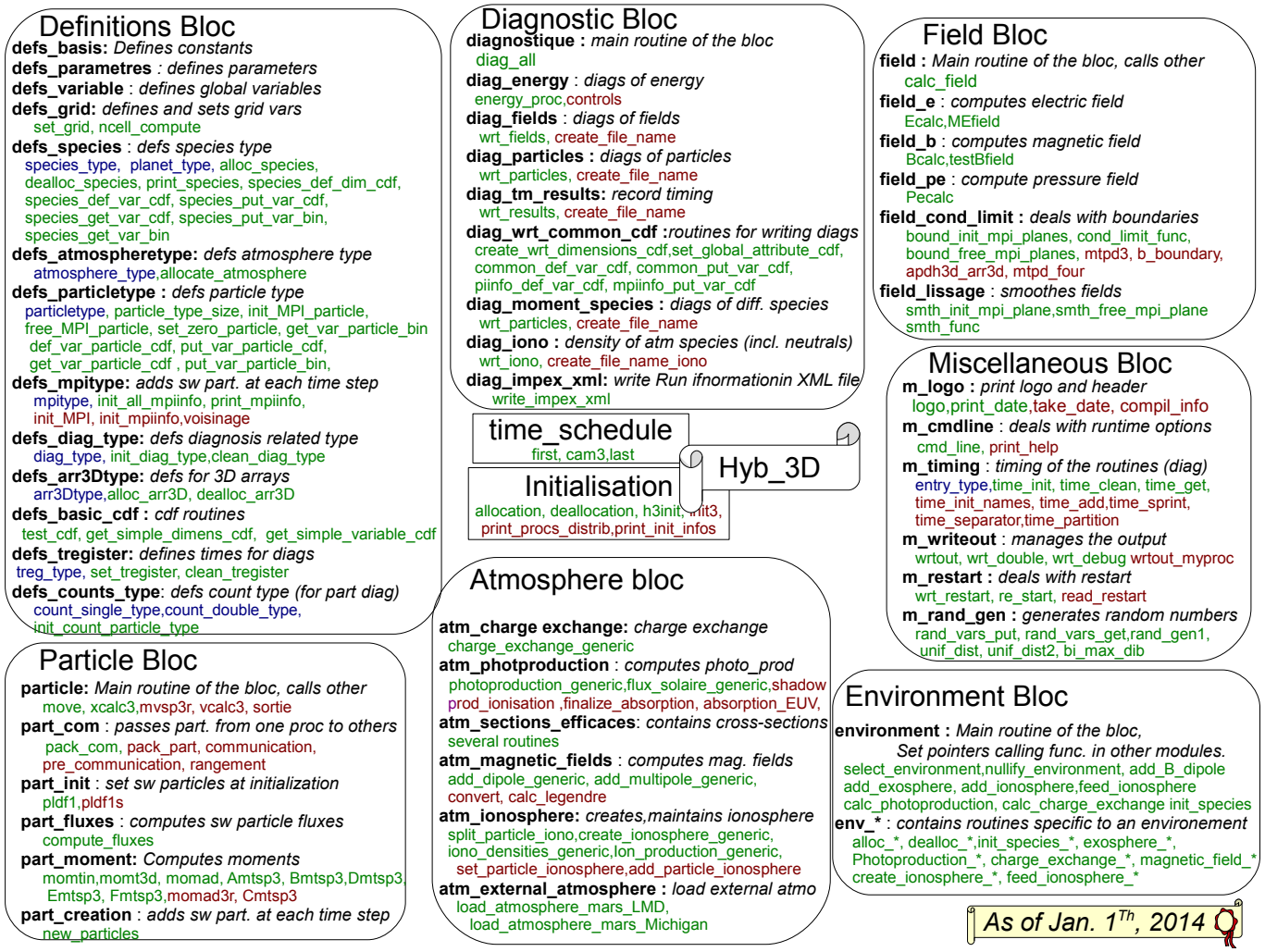

Figure 3.2: Organization of the modules, routines in the modules are shown in green for public ones and red for private ones. Data types are also shown in the module which defines them, in blue.

3.2 post-treatment and diagnostic

Different directories and Makefile commands allows to diagnoses basic and highlevel information concerning outputs of the simulation.
• "80_mpi_enseble" is the directory where fundamental post-treatment are developped. The different modules included in this directory allows reconstructing 3D global maps. It merges information provided by each process in the processus map in order to reconstruct full 3D information. The data cubes resulting of this post-treatment are Magnetic and Electrif fields, electron density, electron

temperatures and bulk speed and the moments (density, speed, temperature) for each ion species tracked in the simuations. These files are the basis of other diagnostics.
• "sputtering_diag" : this functionnality is validated only for the Martian environment. It allows to compute production rates in order to be used as inputs, together with electromagnetic cubes, in particle-test model. Three files are returning by this diagnostic corresponding to three kinds of production sources : Photoproduction, electron impact production and charge exchanges.
• "Ion_flux" : this functionnality is validated only for Mars. It allows computing escaping flux for O+, O2+, CO2+ and H+ ions for the different faces of the simulation box (and the total escaping flux). It allows also to set up an energy thresold in the O+ escaping flux in order to seprate the contribution of hot and cold plasma components.
• "IMPEX_WS" : These routines are used to pre-compute data products in XML format (2Dcuts, 1D along spaccraft trajectory). It has vocation to be archived in the LatHyS catalogue. It also allows to interpolate information (in 1D,2D or 3D) and is used for webservices purposes.
• "IMPEX_FL" : this functionnality is used to generate an executable for webservice puproses. This executable allows to compute field or flow lines for specified footprint.

Chapter 4

Parameters, definitions, nomenclature and structure types

4.1 Defs_parametre.F90

This file define most of the parameters of the simulation which are not specifically related to the environment. In particular, all the parameters which can be modified through the command line are defined here.

4.1.1 Space and time (relate them here)

• ncx0,ncy0,ncz0. Size of the grid, in cell number. Note that the code may automatically resize the grid so to be commensurable with the number of processes.
• gstep(3). Size of a cell, in units of the inertail length of the main ionic specie.
• dt. Time step
• nhm. Number of iterations

4.2 IMF and Planet

The IMF direction is defined with the spherical variables psi (Z versus XY) and phi (X versus Y). The module of the IMF is defined in the environment file.

The environment file which will be used depends on planetname.

4.2.1 Fields and Ionosphere additional settings

Some particular settings can be set to describe how the ionosphere interacts we the surrounding plasma. These settings are defined in defs_parametre.F90:
• absorp_surface. If set to 1, the particles reaching the planet surface are absorbed, otherwise they are lost.
• t_iono_release. Can be used to freeze the ionosphere up to a given time, for a smoother initialization.
• ipe. Defines the polytropic law : 0 = adiabatic; 1 = isothermal; 2 = adiabatic in solar wind and hydrostatic in the ionosphere ($\gamma \propto 1/\log(density)$); 3= isothermal in solar wind and hydrostatic in the ionosphere.
• dn_lim_inf_conduc. Above this density (in simulation units, i.e. must be multiplied by the solar wind density to have physical units), the conductivity of the ionosphere is assumed to be infinity: the $v_i \times B$ term is set to 0. Use this to cancel a abnormal diffusion of negative E_y field in the ionosphere which leads to the ejection of a large part of the ionosphere.
• idisp, iresis. Set to 1 to use dissipation or resistive terms in the electric field computation.

• resis. Set the value of the resistivity.
• dmin. Minimum density for the computation of the electric field.

4.2.2 Optimization

As the simulation box is a cube, with a planet or satellite in its center, some regions on the corners of the simulation box may not play any role in the simulation result (they just see an unperturbed flow of plasma). Moreover, if the planet/satellite as an ionosphere/exosphere, there will be more macroparticles close to it than in the corners of the simulation box. If the simulation box is splitted between the different process in sub-boxes of same sizes, this will lead to (1) processes processing more particles than others, (2) processes simulating nothing interesting. To limit this, it is possible to define a region of interest (ROI) in defs_parametres.F90. This ROI must be defined manually, through 6 variables:
• y_min, y_max: define the position of the ROI in y, in fraction of the total box size in y (y_min=0.33 and y_max=0.67 correspond to a ROI occupying the central third of the box).
• z_min, z_max: define the position of the ROI in z, in fraction of the total box size in z.
• excess : define by how much the size of the box in the ROI must be shrinked (0 means no changes).
• excess_part : allow to inject more particle in the ROI (0 means no changes).

4.3 Environment.F90

This module serves as an interface between the global part of the code and the code specific to the environment simulated. This is done through the definition of a few pointers to the function specific to the environment. This pointers are associated with functions which: (1) make computations specific to an environment, (2) fill the to structures which defines the property of the plasma flow (species structure) and of the exosphere/ionosphere/magnetic field/anything associated with the planet (atmosphere).

To define an environment, a new env_*name*.F90 file must be created and the following pointers should be associated with routines in this file to enable the related physical processes:

4.3.1 planet_species =>

This pointer must be associated with a routine which initializes the species structure defined in defs_species.F90.

4.3.2 dealloc_planet =>

This pointer must be associated with a routine which deallocates arrays definied in the env_*name*.F90 file

4.3.3 exosphere =>

This pointer must be associated with a routine which computes the density of the neutral species in the exosphere.

4.3.4 photoproduction =>

This pointer must be associated with a routine which computes the photoproduction rates of some species. Note that if the cross-sections of the photoproduction reactions have previously been stored

in the atmosphere variable, this pointer can directly be associated with photoproduction_generic which is a generic function.

4.3.5 charge_exchange =>

This pointer must be associated with a routine which implementes the charge exchange processes. Note that if the cross-sections of the charge exchange reactions are constant and have previously been stored in the atmosphere variable, this pointer can directly be associated with charge_exchange_generic which is a generic function.

4.3.6 ionosphere =>

This pointer must be associated with a routine which creates an ionosphere at initialization.

4.3.7 b_dipole =>

This pointer must be associated with a routine which compute the internal magnetic field of the planet.

4.3.8 feed_production =>

This pointer must be associated with a routine which computes the rates of ionization of the plasma through the simulation domain and create the corresponding macro-particles.

4.3.8.0.1 Important Does not change the number and the order of dummy arguments of the subroutines. The procedure pointers are sensible to the dummy argument list so if change the dummy argument order into `m_exo_future.o` you have to do that for all environment-dependent exosphere calculations.

4.4 The Species structure

This structure contains most of the information about the incident plasma flow and the obstacle. It should be filled in the routines associated to the planet_species pointer in environment.F90. before the call to planet_species, the number of incident species ns must be defined. This structure contains:

4.4.1 Reference quantities

These quantities are used to normalize the simulation variables.

- species%ref%density.Physical density of reference, which normalizes density (m^{-3})
- species%ref%c_omegapi. Ions inertial length, which normalizes lengths (km)
- species%ref%mag. Magnetic Field, which normalizes fields (Tesla)
- species%ref%alfvenspeed. Alfvén speed, which normalizes velocity ($km.s^{-1}$)
- species%ref%inv_gyro. Inverse of the gyrofrequency, which normalizes time (s).
- species%ref%maxabs. maximum absorption length (km).

4.4.2 Obstacle parameters

- species%planetname. Name of the environment.

All the following distances or lengths are in simulation units:

- species%P%centr. Position of the planet center.

• species%P%radius. Radius of the planet.
• species%P%r_exo. Radius of the exobase.
• species%P%r_lim. Radius of the bottom limit of the ionosphere.
• species%P%r_iono. Radius of the top limit of the ionosphere (when loaded explicitely).
• species%P%speed. Planet velocity.

4.4.3 Incident plasma parameters

• species%S(:)%name. Names of the incident species.
• species%S(:)%ng. Number of particles per cell for each species.
• species%S(:)%vxs; ...%vys; ...%vzs. Directed velocities along x, y and z.
• species%S(:)%percent. Ratio of each species in the plasma flow. Total must be **1**.
• species%S(:)%rcharge. Species charges (ratio to the dominant specie).
• species%S(:)%rmass. Species masses (ratio to the dominant specie).
• species%S(:)%rtemp. Species temperatures (ratio to the dominant specie).
• species%S(:)%betas. β of each species.
• species%betae. β of the electrons.
• species%S(:)%rspeed. Ratio $\frac{v_\parallel}{v_\perp}$ for each species.
• species%S(:)%rvth. Species thermal velocities (ratio to the dominant specie).
• species%S(:)%rmds. Species macroparticle weights (ratio to the dominant specie).
• species%S(:)%qms. Species charge over mass ratio (ratio to the dominant specie).
• species%S(:)%vth1. Species parallel velocities in simulation units.
• species%S(:)%vth2. Species perpendicular velocities in simulation units.
• species%S(:)%sm. Mass of the macroparticles of each species.
• species%S(:)%sq. Charge of the macroparticles of each species.

4.4.4 Ionospheric plasma parameters

• species%tempe_ratio. Temperature of the ionospheric electrons (ratio to the incident plasma electrons).
• species%viscosity. Collision frequency of the ions in the ionosphere (in $cm^3.s^{-1}$).

4.5 The Atmosphere structure

The exophere is primarily defined through the use of the `density_exo` and `prod_pp` arrays defined in `environment.F90`. From here, there are two ways of loading exospheric particles in the simulation: The first one is "by hand", through customized routines. You can use the `exophere`, `photoproduction`, `ionosphere` and `charge_exchange` pointers in `environment.F90` to call your routines. The second possibility is to use generic functions (those with the "atm_" prefix) and an *atmosphere* structure. The following addresses the second option.

4.5.1 Global environment variables

The *atmosphere* structure is defined through the call, in environment.F90, of a subroutine alloc_*planetname* which must exist in your env_*planetname*.F90 file. You must then define:
• the number of exospheric species (neutrals and ions): n_species.
• the number of charge exchange reactions : n_exc.
• the number of electron impact ionization reactions : n_ei.

• the number of photoproduction reactions : n_pp.
• the number of species produced by the photoproduction reactions (if two reactions produce the same ion) : n_spe_pp.
• the number of EUV wavelength for the photoproduction computations: nb_lo.
• the number of photoproduction reactions whose rate does not depends on the wavelength : n_pp_freq_fixed.

The *atmosphere* structure is obtained by calling:
allocate_atmosphere(ncm,atmosphere,density,prod_pp). Example:

```
atmosphere%n_species=7 !number of exospheric species (neutral and ions)

atmosphere%n_pp=4 !number of photoproduction reactions

atmosphere%n_spe_pp=3 !number of species obtained by photoproduction

atmosphere%n_pp_freq_fixed=0 !number of photoproduction reactions with fixed rates

atmosphere%nb_lo=37 !number of wavelength in the EUV spectrum

atmosphere%n_exc=4 !number of charge exchange photoreactions

atmosphere%n_ei=2 !number opf electron impact reactions

call allocate_atmosphere(ncm,atmosphere,density,prod_pp)
```

Each species has then to be defined, one should precise:
• the species mass.
• the species charge.
• if the species opacity in EUV must be used (.TRUE. or .FALSE.).
• if the species must be load as ionospheric particle (.TRUE. or .FALSE.).
• the pointer to the photoproduction array for photoproduced species (only).
• the species name (no blank in front).

Examples:

```
atmosphere%species(1)%name = "O+ "
atmosphere%species(1)%mass = 16._dp
atmosphere%species(1)%charge= one
atmosphere%species(1)%opaque= .FALSE.
atmosphere%species(1)%iono = .TRUE.
atmosphere%species(1)%prod => prod_pp(:,:,:,1)

atmosphere%species(2)%name = "O "
atmosphere%species(2)%mass = 16._dp
atmosphere%species(2)%charge= zero
atmosphere%species(2)%opaque= .TRUE.
atmosphere%species(2)%iono = .FALSE.

atmosphere%species(3)%name = "H "
atmosphere%species(3)%mass = 1._dp
atmosphere%species(3)%charge= zero
atmosphere%species(3)%opaque= .TRUE.
atmosphere%species(3)%iono = .FALSE.
```

4.5.1.0.2 Important The ordering of the species is important. **All ion species** have to be descibed **first** following by all neutral compnents.

4.5.2 Photoproduction reactions

The photoproduction has to be defined in the alloc_*planetname* subroutine. It is defined in the *atmosphere* structure, by setting pointers to the mother and daughter species. Example:

```
!O->O+
atmosphere%photo_reactions(1)%mother => atmosphere%species(2) !O species
atmosphere%photo_reactions(1)%daughter => atmosphere%species(1) !O+ species
```

The fields atmosphere%EUVFLX (EUV flux), atmosphere%ion_abs (absorption cross-section) and atmosphere%ion_react (ionization cross-section) will have to be filled before calling the subroutine photoproduction_generic(Spe,ncm,gstep, s_min_loc,atmosphere), which computes the prod_pp arrays in a generic way. Note that if these values (generally obtained through calls to routines of the atm_sections_efficaces.F90 module) are stored by alloc_*planetname*, you do not need to write a specific photoproduction routine and can compute the photoproduction rate by associating directly the photoproduction pointer to the photoproduction_generic routine in `environment.F90`.

4.5.3 Charge exchange reactions

The charge exchange reactions have to be defined in the alloc_*planetname* subroutine. they are defined in the *atmosphere* structure, by setting pointers to the mother neutral and ion species as well as the charge/mass ratio of the mother ion specie and the cross-section of the reaction (if constant). Example:

```
!H +O+ ->H+ + O

atmosphere%exc_reactions(1)%qsm = one/16._dp

atmosphere%exc_reactions(1)%ion =>atmosphere%species(1)

atmosphere%exc_reactions(1)%neutral =>atmosphere%species(3)

atmosphere%exc_reactions(1)%cross_section = 9.E-20
```

The field atmosphere%cross_section will have to be filled before calling the subroutine charge_exchange_generic(ijk,s_p,v_p,w,Spe,particule,atmosphere), which implementes the charge exchanges in a generic way. Note that if this valueis constant and is stored by alloc_*planetname*, you do not need to write a specific charge exchange routine routine and can compute the charge exchange rate by associating directly the charge_exchange pointer to the charge_exchange_generic routine in `environment.F90`.

4.5.4 Electron impact reactions

The electron impact reactions have to be defined in the alloc_*planetname* subroutine. they are defined in the *atmosphere* structure, by setting pointers to the mother (neutral) and daughter (ion) species as well as the cross-section of the reaction defined by $\sigma = \sum_{i=1}^{n} \text{coeff}(i) * \ln(T_e(K))^{i-1}$. Example:

```
!O + e-> O+ atmosphere%ei_reactions(2)%ion =>atmosphere%species(1)

atmosphere%ei_reactions(2)%neutral =>atmosphere%species(2)

atmosphere%ei_reactions(2)%n = 5

atmosphere%ei_reactions(2)%coeff(1:5) = (/-1233.29,347.764,-37.4128,1.79337,-0.032277/)
```

The electron impact reactions are automatically implemented if defined in atmosphere, without needing further declaration in environment.F90.

Chapter 5

Platform, compiler information and job submission

5.1 The CICLAD platform, information for the set up

CICLAD is the platform used to execute parallel simulations. Log in on this machine is restricted to a SSH acces (SSH key have to be activated). The first time we have to configure our working environment. The list of command and functionnality for the platform are availabel at http://ciclad-web.ipsl.jussieu.fr/ciclad-utilisation-en.pdf. The first thing to do is to select the OPNEMPI version relevant to the COMPILER choice 5.2.1. Commands to select the mpi implementation are the following

- **mpi-selector –query** : indicate your active mpi implementation

the displayed results is

```
default:openmpi-1.6.4-pgf-x86_64
level:user
```

- **mpi-selector –list** : indicate the list of possible mpi implementation

```
lam-x86_64
openmpi-1.2.8-gcc-x86_64
openmpi-1.2.8-pgf-x86_64
openmpi-1.4.2-g95-x86_64
openmpi-1.4.2-gfortran-x86_64
openmpi-1.4.2-pgf-x86_64
openmpi-1.4.3-ifort-x86_64
openmpi-1.4.3-pgf-x86_64
openmpi-1.4.3-pgfgcc-x86_64
openmpi-1.6.4-pgf-x86_64
```

- **mpi-selector –set openmpi-1.6.4-pgf-x86_64** : activate the mpi implementation

We suggest to use the Potrland Group fortran compiler and the corresponding opnempi version (shown above).

5.1.0.0.3 Important The change of the mpi implementation is activated after the next login.

5.1.0.0.4 Activation of the compiler version To activate the compiler version (for Portland Group Fortran) the user has to exectute the following command (only once).

```
touch pgi2011
```

5.2 The Makefile

The `Makefile` is a gnu Makefile and permits to compile easily the `quiet_plasma`, `diag`, `diag_SPUTTERING`, `diag_IMPEX`, `ion_flux`, `interpole_fields` and `getFieldLine` binaries.

5.2.1 Compiler choice

It contains the COMPILER flag where you have to put the path toward the mpi fortran. Then there are three flags which permit to select the compile option for anyone of the three compiler tested:

IFORTFLAG Compilation flags for intel fortran (tested with version 10.1 and 11.1).

GFORTFLAG Compilation flags for gfortran (tested for versions ≥ 4.0). Some version of gnu compiler does not procedure pointer.

PGFORTFLAG Compilation flags for Portland Group fortran (work for version ≥ 10).

To select the good compiler flags you have to change the value of the FFLAGS variable accordingly with the compiler chosen.

The source files are compiled by "make" command. The corresponding object files are listed in the `Makefile` by respecting the dependencies between the modules. At the moment the program structure and the object file list permits to compile the program in parallel with two threads. So with the command "make -j 2" you can compile `quiet_plasma` very quickly.

5.2.2 Compilation Options

The `quiet_plasma` program have some compilation options which enabling or not some *PREPROCESSOR MACROS* permits to compile the program for some specific purposes.

HAVE_TIMING Complete time analysis.

HAVE_NETCDF Read and Write of diagnostic files in NetCDF format.

HAVE_DEBUG Enable debug. For any called routine, the begin and the end are annotated. More controls on the quantities. Add also the good path to headers and libraries in the INC_CDF and LIB_CDF flags.

HAVE_DOUBLE_PRECISION Use double precision real instead of simple precision.

HAVE_NO_PLANET No planet is not considered, only the environment.

HAVE_WAVE_TEST The input plasma is a Alfven wave. Use with HAVE_NO_PLANET.

HAVE_MAXSIZEWRITE If in writing, the system have some limitation in the buffer size, then this option permits to write some big quantities (like particles) not at once.

To activate one of more of the compilation option add -DHAVE_"option" to the compilation flag in use.

5.3 The basic steps to perform a simulation

The fundamentals steps to compile, submit and post-treat a job are described below.

1. Connect to CICLAD cluster through SSH
2. Compile the sources (in the directory where the Makefile is located). "make"

3. Copy the executable file to the "data" disk. Repertory "/data/you/..."

4. submit the job via a job submission file (**??**). The command is
   qsub -l nodes=1:ppn=64 *hyb3d_...*
   where *nodes* is the number of nodes (server) and *ppn* is the number of processor per node.

5. The simulation is in the queue system and one can check is ecolution witht he command
   qstat -u *you*

6. Once the simulation is done (or when diagnostic files are created). Compile the diagnostic executable file, in your home directory (where the Makefile is located), compile the diagnostic file with the command "make diag".

7. Copy the "diag" binary file in your data directory "/data/you/..."

8. Execute the following command "./diag -d DD_MM_YY -t TTTTT" where DD_MM_YY is the runID (date of the simualtion and "TTTTT" is the time of the diagnostic.

9. Data cubes are created and can be visualize with visualization softwares.

5.3.1 The batch submission file

To submit the job CICLAD requires a batch simulation file. Example of the batch is detailed below

```
#!/bin/sh
#PBS -qparalong
#####PBS -k eo
dir=/data/modolo/Mars3D_sim/MAVEN_case1_80km/

cd ${dir}
mpirun -mca btl self,sm,openib ${dir}/quiet_plasma -ncxyz 200 380 380  -nhm 1800
0  -pn mars -s sortie_mars -dt .03333333333 -dx 0.705 -ssl 180.   > sortie_ecran
.txt
```

The first three lines precise the shell environment, the queue requested (walltime), the standard and error output. The "dir" line indicate the directory where the simulation files will be written.

The code allows a certain number of parameter change with respect to the compile sources specification. The list of command/parameter which can be change are available by executing the ./quiet_plasma -help. This command returns :

```
------------------------------------------------------------
 ------------------------------------------------------------
||        ||     __      ||
|| \\  || ||   //  \\    ||
||  \\ || ||       ||    ||
||   \\|| ||       //    ||
||\\  \|| ||\\  ===|   //||
|| \\  || || \\    \\ // ||
|| ||  || || //    || \\ ||
|| ||  || ||// \\__//  \\||
                ||||  Hyb 3D
   An Hybrid-Model 3D-code for Mars/Solar Wind Interaction
   MARS3D : periodic bcs in Y and Z  open bcs in X
 ------------------------------------------------------------
 ------------------------------------------------------------
```

```
  Last BZR revision number:                          245
  Compiler:                                      Unknown
  Precision:                                      simple
  HAVE_NO_PLANET:                                     No
  HAVE_WAVE_TEST:                                     No
  HAVE_NETCDF:                                       Yes
  HAVE_DEbUG:                                         No
  Compiling date:                            Feb 07 2014


  Known problems:
  1)
  2)


quiet_plasma options:

 -v, --version         print version information and exit
 -h, --help            print usage information and exit
 -t, --time            print time
 -s, --qp_out          set the ouput file name
 -pn, --planetname     set the planet environment,
                       followed by a string:
                              mars
                              mercure
                              moon
                              ganymede
                              mars3try
                              titan
 -r, --restart         set the restart,
                       followed by:
                           0, restart is off
                           1, simple restart
                           2, restart adding cells in Y
                           3, restart adding cells in X
                       Note: when restarting be careful to impose the same
                             number of cells (ncx,ncy,ncz) and  planet name
                             used in the previuos run.
                             dt,dx are selected automatically.
 -rf, --restart_file   set the restart file,
                       followed by 0 or 1
 -nhm, --maxiter       set the max number of iteration,
                       followed by 1 integer.
 -ncxyz, --cell_size   set the cell number ncx,ncy,ncz at once
                       followed by 3 integers.
 -ncx,                 set the cell number in X (ncx)
                       followed by 1 integer.
 -ncy,                 set the cell number in Y (ncy)
                       followed by 1 integer.
 -ncz,                 set the cell number in Z (ncz)
                       followed by 1 integer.
```

```
-dt, --time_step      set time step (dt)
                      followed by 1 real.
-dx, --gstep          set the spatial step (gstep)
                      followed by 1 real (all the
                       composent of gstep will have the same value).
-ug, --uniform_grid   set a uniform domain decomposition
-le, --load_exosphere load exospheric neutral density from a file
                      the file name corresponds to input name
-la, --load_atmosphere load atmospheric neutral density from a file
                      the file name corresponds to input name
-nm, --no_mag set to 1 for unmagnetized Mars object
-ssl, --planet_ssl    fix the solar longitude of the main crustal source
```

Chapter 6

Data cubes and data format

6.1 Simulation Outputs

Global simulation outputs provide a 3D view of the state of the simulation and quasi-stationary solutions. The set of files (and physical parameters associated) which are generated for a typical Martian simulation run :
• "Magw_XXXX.nc" : 3D magnetic field components (Bx,By,Bz)
• "Elew_XXXX.nc" : 3D electric field components (Ex,Ey,Ez)
• "Thew_XXXX.nc" : 3D delectron density, bulk velocity components (Ux,Uy,Uz), electron temperature
• "Read_moment_species_XXXX.dat" : ascii file containing ion species name include in the simulation
• "Hsw_XXXX.nc" : Solar wind proton diagnostic : 3D density, bulk speed and temperature (Dn, Ux,Uy,Uz,T)
• "Hesw_XXXX.nc" : Solar wind alpha particle diagnostic : 3D density, bulk speed and temperature (Dn, Ux,Uy,Uz,T)
• "Hpl_XXXX.nc" : Planetary proton diagnostic : 3D density, bulk speed and temperature (Dn, Ux,Uy,Uz,T)
• "Opl_XXXX.nc" : Planetary O+ diagnostic : 3D density, bulk speed and temperature (Dn, Ux,Uy,Uz,T)
• "O2pl_XXXX.nc" : Planetary O2+ diagnostic : 3D density, bulk speed and temperature (Dn, Ux,Uy,Uz,T)
• "CO2pl_XXXX.nc" : Planetary CO2+ diagnostic : 3D density, bulk speed and temperature (Dn, Ux,Uy,Uz,T)
• "Atmw_XXXX.nc": 3D neutral distribution (Dn for each neutral species)

Concerning ion species (H+pl, O+pl, O2+pl and CO2+pl), information are provided without discrimination with respect to the source of production (photoproduction, electron impact, charge exchange or ionospheric chemistry).

6.1.1 Data format

Archived outputs files are saved in the "netCDF" format (http://www.unidata.ucar.edu/software/netcdf/). This format is well documented and largely adopted in numerical and data archives.

6.1.2 Data descritption

6.1.2.1 Grid and coordinate system

The simulation uses a 3D uniform Cartesian grid with the following coordinate system called "simulation coordinate system".
• For Mars/Mercury's simulation: The x-axis is aligned with the solar wind direction (+Vsw), ie from the Sun toward the planet. The z-axis is perpendicular to Mars' orbital plane and the y-axis

Density	$n_{phys} = n_{simu} * n0$
Magnetic field	$B_{phys} = B_{simu} * B0$
Velocity	$V_{phys} = V_{simu} * V0$
Time	$t_{phys} = t_{simu} * t0$
Length	$x_{phys} = x_{simu} * x0$
Electric field	$E_{phys} = E_{simu} * v0 * B0$
...	

completes the right handed system.
• For Titan/Ganymede's simulation: The x-axis coincides with the ideal magnetospheric plasma flow direction, the z-axis is perpendicular to Titan/Ganymede ecliptic plane and the y-axis completes the right handed system. One has to note that the origin of the coordinate system is the bottom right corner of the simulation box. The position of the planet is indicated in the variable 's_centr'.

However, the output information are provided in physical reference frame (MSO for Mars and Mercury, TIIS for Titan and GPHIO for Ganymede). Information about vector components are in such frame and indication of the grid point position in this reference frame is provided by the array called "X_axis", "Y_axis" and "Z_axis".

6.1.2.2 Simulation unit and normalization

The hybrid model solves a set of equation which has been nondimensionalized. Physical units are therefore removed and all quantities/variables do not have units. Each basic quantity has been parametrized with a combination of characteristical values. The basic quantities are:

- ✘ time t0
- ✘ length x0
- ✘ speed v0
- ✘ Density n0
- ✘ Magnetic field B0
- ✘ Mass m0
- ✘ charge q0

These parameters are constants. To transform from simulation values to physical values, one has to applied the following method
$Z_{simulation} * Z_0 \longrightarrow Z_{physical}$
With $Z_{simulation}$ the simulated quantity, Z_0 the characteristical parameter (or combination of characteristical parameters). The most used transformations are :

6.1.2.2.1 Incident Plasma: Typical Parameters

1. Solar Wind / Magnetospheric Plasma

   Few plasma parameters definition (in reference to ion specie 'i' impigned in a magnetic field B_0) :

   Ion intertial length

   $$\frac{c}{\omega_{pi}} = c\sqrt{\frac{\epsilon_0 m_i}{e^2 n_i}}$$

   ion cyclotron frequency

   $$\Omega_i = \frac{q_i B_0}{m_i}$$

   thermal speed

   $$v_{thi} = \sqrt{\frac{2k_B T_i}{m_i}}$$

   Alfvèn speed of specie 'i'

   $$V_A = \frac{B_0}{\mu_0 m_i n_i}$$

   Sound speed

   $$C_s = \sqrt{\frac{3T_e}{m_i}}$$

   Ion specie 'i' plasma beta :

   $$\beta_i = \frac{n_i k_B T_i}{\frac{B_0^2}{2\mu_0}}$$

   Alfvèn Mach number

   $$M_A = \frac{V_{incplasma}}{V_A}$$

   Sonic Mach number

   $$M_S = \frac{V_{incplasma}}{C_s}$$

   Magnetosonic Mach number

   $$M_{MS} = \frac{M_A M_s}{\sqrt{M_A^2 + M_s^2}}$$

   Mars

   Typical solar wind plasama parameters at the Martian orbit :

   The main ion specie is H^+. Simulation values have been normalized with respect to H^+ parameters. IMF :$B_{IMF} = (1.6, 2.5, 0.)nT$

   Bulk speed : $V_{sw} = (400, 0, 0)km/s$

   Electron temperature : $T_e = 3.10^5 K$.

   The derived quantites are listed in table 6.1.

   Simulation quantites are normalized to the dominant ion species of the incoming plasma. Therfore :

   $$X^{physics} = X_0 \times X^{simulation}$$

   For the Martian solar wind plasma, the characteristics plasma scales are in table 6.2

   Values to set up in the simulation are in table 6.3

H^+	He^{++}
m_{H^+}	$4m_{H^+}$
q_{H^+}	$2q_{H^+}$
$n_{H^+} = 2.5$ cm^{-3}	$n_{He^{++}} = 0.05 n_{H^+}$
$T_{H^+} = 5.10^4$ K	$T_{He^{++}} = 4T_{H^+}$

Table 6.1: Solar wind plasma typical parameters

H^+	He^{++}
$\frac{c}{\omega_{pi}} = c\sqrt{\frac{\epsilon_0 m_i}{e^2 n_i}} = 150.15$ km	$\frac{c}{\omega_{pi}} = c\sqrt{\frac{\epsilon_0 m_i}{e^2 n_i}} = 671.48$ km
$\Omega_i = \frac{q_i B_0}{m_i} = 0.29 rad.s^{-1}$	$\Omega_i = \frac{q_i B_0}{m_i} = 0.14 rad.s^{-1}$
$v_{thi} = \sqrt{\frac{k_B T_i}{m_i}} = 28.7$ km/s	$v_{thi} = \sqrt{\frac{k_B T_i}{m_i}} = 28.7$ km/s
$\beta_i = \frac{n_i k_B T_i}{\frac{B_0^2}{2\mu_0}} = 0.44$	$\beta_i = \frac{n_i k_B T_i}{\frac{B_0^2}{2\mu_0}} = 0.09$
$V_A = \frac{B_0}{\mu_0 m_i n_i} = 39.38$ km/s	

2. Ganymede Typical jovian plasma parameters at the Ganymede orbit :

   The main ion specie is O^+. Simulation values are normalized with respect to O^+ parameters.

   Incoming magnetinc field : $B_j = (0., 0., 120) nT$.

   Bulk speed : $V_j = (180, 0, 0) km/s$

   Elcetron temperature : $T_e = 100. eV$.

   The derived quantites are listed in table 6.4.

   For the jovian magnetospheric plasma at Ganymede, the characteristics plasma scales are in table 6.5

   Values to set up in the simulation are in table 6.6

6.1.2.3 simulation variables

General information concerning each file can be accessed through netcdf functionalities. For instance in a terminal window the command "ncdump -h Magw_28_11_12_t00000.nc" returns information concerning the "Magw_XXXX.nc" file . It is composed of three parts : 1- dimensions of variables of the files, 2- the name of the variables, their data type and dimension, 3- the attribute of the file.

Table 6.2: Normalized values in the simulation

Magnetic field : $B_0 = \|B_{IMF}\| = 3nT$	Electric charge : $q_0 = q_{H^+} \sim 1.602 \times 10^{-19}$ C
Mass : $m_0 = m_{H^+} \sim 1.67 \times 10^{-27}$ kg	Length : $L_0 = c/\omega_{pi}(H^+) = 150$ km
Time : $T_0 = \Omega_i^{-1}(H^+) \sim 3.44s$	Speed :$V_0 = V_A \sim 39.38$ km/s
Density : $n_0 = n_{H^+} = 2.5cm^{-3}$	

Table 6.3: Simulations values for the Martian Solar wind

$\beta_e = 2.79$	$\beta_i(H^+) = 0.44$	$\beta_i(He^{++}) = 0.09$
$V_{incoming} = 400./V_A = 10.15$	$v_{th}(H^+) = 0.728$	$v_{th}(He^{++}) = 0.728$
He_percent = 0.05		

O^+	H^+
m_{O^+}	$1/16m_{O^+}$
q_{O^+}	q_{O^+}
$n_{O^+} = 2.96$ cm^{-3}	$n_{H^+} = 0.2n_{O^+}$
$T_{O^+} = 360$ eV	$T_{H^+} = 1/16T_{O^+}$

Table 6.4: Jovian plasma typical parameters

O^+	H^+
$\frac{c}{\omega_{pi}} = c\sqrt{\frac{\epsilon_0 m_i}{e^2 n_i}} = 529.42$ km	$\frac{c}{\omega_{pi}} = c\sqrt{\frac{\epsilon_0 m_i}{e^2 n_i}} = 264.71$ km
$\Omega_i = \frac{q_i B_0}{m_i} = 0.72 rad.s^{-1}$	$\Omega_i = \frac{q_i B_0}{m_i} = 11.49 rad.s^{-1}$
$v_{thi} = \sqrt{\frac{k_B T_i}{m_i}} = 46.42$ km/s	$v_{thi} = \sqrt{\frac{k_B T_i}{m_i}} = 46.42$ km/s
$\beta_i = \frac{n_i k_B T_i}{\frac{B_0^2}{2\mu_0}} = 0.03$	$\beta_i = \frac{n_i k_B T_i}{\frac{B_0^2}{2\mu_0}} = 0.007$
$V_A = \frac{B_0}{\mu_0 m_i n_i} = 377.39$ km/s	

Table 6.5: Normalized values in the simulation

Magnetic field : $B_0 = \|B_j\| = 120nT$	Electric charge : $q_0 = q_{O^+} \sim 1.602 \times 10^{-19}$ C
Mass : $m_0 = m_{O^+} \sim 26.72 \times 10^{-27}$ kg	Length : $L_0 = c/\omega_{pi}(H^+) = 529$ km
Time : $T_0 = \Omega_i^{-1}(H^+) \sim 1.38s$	Speed :$V_0 = V_A \sim 377.39$ km/s
Density : $n_0 = n_{O^+} = 2.96cm^{-3}$	

Table 6.6: Simulations values for the Ganymede

$\beta_e = 0.021$	$\beta_i(O^+) = 0.03$	$\beta_i(H^+) = 0.007$
$V_{incoming} = 180./V_A = 0.477$	$v_{th}(H^+) = 0.123$	$v_{th}(He^{++}) = 0.123$
H_percent = 0.2		

Chapter 7

Changing input parameters

The user can change some parameter from the script submission file (cf 5.3.1). The most used options are :

- ✘ `-nhm` change the maximum number of iteration
- ✘ `-ncxyz` change the number of cell in X, Y and Z directions
- ✘ `-dx` change the spatial resolution
- ✘ `-dt` change the time step
- ✘ `-nm` no crustal field included (only valid for Martian simulation)
- ✘ `-ssl` fix the solar longitude of the main crustal field (180 for main CF on the dayside, valid onky for Mars)
- ✘ `-le` load an exospheric file
- ✘ `-la` load a thermospheric file

7.1 Changing the incoming plasma parameters

If one wants to change the input condition of the solar wind plasma we have to make the following changes (we take exemple for Mars but it can be applied to other simulated objects).

7.1.1 Solar wind density

In the file "`env_mars.F90`", subroutine "`init_species_mars`":
we have to change the value of
`species%ref%density`

7.1.2 Solar wind velocity

In the file "`env_mars.F90`", subroutine "`init_species_mars`":
we have to change the value of
`species%rS(H)%vxs`

7.1.3 IMF magnitude

In the file "`env_mars.F90`", subroutine "`init_species_mars`":
we have to change the value of
`species%ref%mag`

7.1.4 Solar wind composition (percentage of He++)

In the file "`env_mars.F90`", subroutine "`init_species_mars`":
we have to change the value of
`species%S(He)%percent`

7.1.5 Solar activity

In the file "`env_mars.F90`", subroutine "`photoproduction_mars`":
we have to change the value of
`atmosphere%F107`
`atmosphere%F107_Avg`

7.1.6 IMF orientation

In the file "`defs_parametere.F90`"
we have to change the value of
`psi`
`phi`
psi is the angle between the Z_{sim} axis and the magnetic field (it gives the Bz component). phi is the angle between the X_{sim} axis and the projection ofthe magnetic field in the $X_{sim}Y_{sim}$plane. Please note that $X_{sim} = -X_{MSO}$ and $Y_{sim} = -Y_{MSO}$.