Georgia Tech and Art et Metier ParisTech

Master Thesis

Ablation-Raditation coupling modeling for Hayabusa reentry vehicle

Author:
Valentin Marguet

Supervisors:
Dr. Penelope Leyland
Elise Fahy
Antoine Ducoin

A thesis submitted in fulfilment of the requirements for the degree of Master of Mechanical engineering

Undertaken at

IAG
EPFL

May 8, 2013

GEORGIA TECH AND ARTS ET METIERS PARISTECH

Abstract

Mechanical Engineering
EPFL

Master of Mechanical engineering

Ablation-Raditation coupling modeling for Hayabusa reentry vehicle

by Valentin MARGUET

Reentry vehicles undergo extreme thermal conditions as they reach hypersonic velocities in particular conditions. Thus thermal protection system (TPS) are required to prevent the probe to be damaged. When it comes to Earth's reentry, capsules like `Hayabusa` are equipped with an carbon phenolic TPS which ablates and releases ablation products into the boundary layer during reentry. Besides, as radiation can be a significant component of the overall heat load, chemical reactions may occur between the radiators and the ablative injected species so that the whole aerothermodynamic state is modified.
Therefore, an accurate numerical modeling of the thermal, physical and chemical processes induced by hypersonic reentry , is needed. It implies to couple the fluid solver and the material response code thanks to a partitioned coupling algorithm. As for `Hayabusa`'s reentry simulations performed for the ARC project, ablation-flowfield and radiation-flowfield resulted to be accurate as they were compared to flight data or empirical correlations. Next step of ablation-radiation coupling consists in experimental tests in plasma wind tunnels and full coupling algorithm.

Acknowledgements

Many people have been generous with their thoughts and time during this project. This work would not have been possible without my colleagues' advice, feedback, constructive ideas. My thanks are addressed to :
Penelope Leyland for allowing me to do this Master thesis in his lab, Elise Fahy for her supervision, help, her smile and everything else, Nikhil Banerji, Oleg Kotsur, Jeremy Mora-Monteros for the work we've done and the good times and laughs we shared, Angelo Casagrande, Yann Steiner, Robin Dufour and Pierre Wilhelm for the coffee breaks reliefs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Arc project

The present work describes the contribution of IAG (Interdisciplinary aerodynamics group) at EPFL to the European project called ARC: "Ablation-Radiation coupling". This study is lead by a consortium composed of a leading European Aerospace Research University, the Ecole Polytechnique Federale de Lausanne, a leading European Aerospace Research Establissement CIRA, the IRS Institute of the University of Stuttgart, an experienced Aerospace SME, FGE, together with subcontractors: the University of Queensland (Australia), a well known expert on ablation modelling, Dr. G. Duffa as a consultant, and ASTRIUM-St.
This project addresses the ablation-radiation coupling for high speed reentry and the physical phenomena that are induced by hypersonic flight conditions: ablation and radiation over a thermal protection system (TPS). Therefore, both test campaigns in major European plasma wind tunnels and coupling simulations are lead simultaneously based on data obtained with past Earth return missions like *Stardust* or *Hayabusa*. This particular contribution of IAG deals with the radiation-ablation-flowfield numerical coupled simulation of *Hayabusa* 's reentry.

1.2 Radiation and TPS ablation induced by hypersonic reentry

Earth return missions like *Hayabusa* are associated with high velocities ($12\,km/s$) so that such reentry capsule have to undergo extreme thermal conditions (around 10 and 12 MW/m^2 in particular conditions). Therefore, they are equipped with Thermal Protection Systems (TPS) to avoid the capsule to be damaged and insulate the vehicle's content. High speed reentry velocities lead to a strong shock wave upstream of the vehicle which strongly modifies the state of the fluid behind it and thus induces high heat fluxes. The overall thermal heating results

for the contribution of several heat transfer processes: diffusion, conduction and radiation. All these heat transfer modes are dependent on the probe's surface temperature so that there is a strong interaction between both the fluid and solid domains.

During reentry, chemical species that are in the shock layer zone undergo a high heating so that many thermal non-equilibrium processes can occur like dissociation, recombination, ionization. The hot plasma forming in front of the probe can lead to radiative heating. Thus the probe is exposed to a very severe heating environment. That is why an effective TPS is required to ensure the probe's resistance. The studied ablators are made of phenolic impregnated carbon and are used to withstand the thermal conditions. Besides, they have to be optimized in order to minimize their mass which is a crucial parameter in space missions. As these shields release disintegrated materials and reactive species into the boundary layer, many chemical reactions can occur and lead to the formation of radiators so that there is a strong coupling between ablation and radiation. For a better understanding, figure 1.1 schemes the overall processes occuring during reentry. TPS design is mainly based on numerical tools. A coupled approach between the solid domain and the fluid one has to be lead as they have both a strong thermal, physical, and chemical impact on each other.

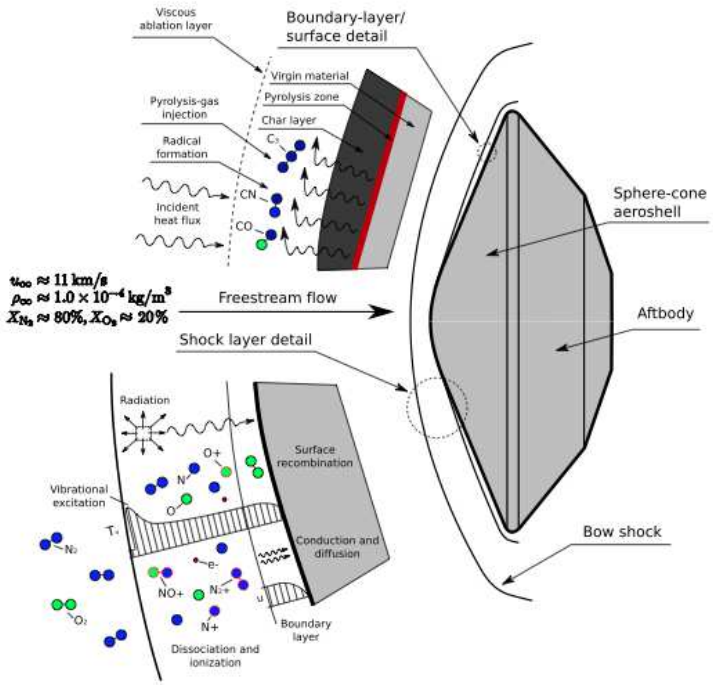


FIGURE 1.1: Illustration of aerothermodynamic processes occurring in the shock-layer and on the surface of a Stardust-type re-entry capsule at peak heating conditions ([1])

1.3 Hayabusa reentry capsule

Hayabusa is a JAXA asteroid sample return mission that re-entered into Earth atmosphere in June 2010. The entry speed exceeded 12 km/s, so that it was equipped with a carbon fiber and resin ablating TPS. The forebody is a 45 degree sphere-cone with a 0.2 m nose radius (figure 1.2).

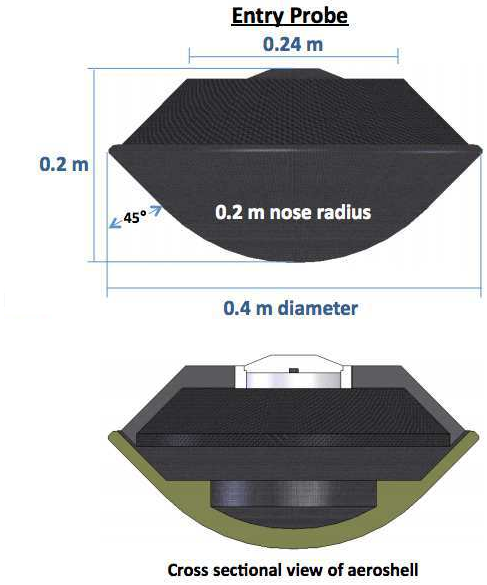


FIGURE 1.2: *Hayabusa* geometry [2]

1.4 EPFL tools

The tools used during this study are validated for aerothermodynamics. A fluid solver developed by University of Queensland called `Eilmer3` was used. It contains radiative transfer models and ablation coupling was implemented in its framework during the present work via an ablating boundary condition. As for the solid domain, a one dimensional material response ablation code developed by Joshi and based on the work of Amar [3] was chosen.

Chapter 2

Fluid flow numerical modeling

2.1 Eilmer3: overview of the code

`Eilmer3` is a compressible flow simulation code being developing at the *Centre for Hypersonics of the University of Queensland*, Australia. This code solves Navier-Stokes equations using a cell-centered and time-dependent finite volume formulation. Thus, it can solve transient compressible flows on planar, axisymmetric and fully three dimensional geometries generated with a block-structured grid. As many features of hypersonic flows are still being developed, this code is a very suitable one for atmospheric reentry. Indeed, it contains several models for chemical reactions and boundary conditions, supports thermal radiation.
The computational core of `Eilmer3` is written in `C/C++` while user-defined functions such as boundary conditions are provided as `lua` scripts. Preprocessing and postprocessing are handled by `Python` programs. All of the simulations of the present work were performed with the OpenMPI version of the code: the flow domain is divided into multiple blocks and each one of them is handled by a singe processor on a cluster computer.

The run of a simulation in `Eilmer3` usually follows the following 3 main steps:

- **Preprocessing**: Edit `Python` file including all the preprocessing information such as the mesh definition and the simulation parameters: Case thermochemical definition, solver parameters, etc. This file is then used by `e3prep.py` to create the grid ans the initial solution.
- **Simulation**: All the files created in the previous step are used by `e3shared.exe` to find a flow solution during the run.
- **Post-Processing**: Once the solutions are written, the program `e3post.py` edits data from the simulation results. Those data are then used in *Paraview* or *Matplotlib*.

2.2 Governing equations

2.2.1 Navier-Stokes equations

During atmospheric reentry, the evolution of a compressible hypersonic fluid flow is described by the Navier-Stokes equations. This set of differential equations express the conservation of mass, momentum and energy. As thermochemical modeling depends on the assumptions made on chemical and thermal equilibrium, we will assume in the following report that:
They can be derived and written in conservative form as follows:

- Chemical nonequilibrium
- Thermal nonequilibrium
- Two-temperatures gas model formulated by Park [4]

Thus chemical and thermal modes cannot be described by a single thermodynamic temperature. The model used in this work considers T_{tr} the translation-rotation energy mode and T_{ve} the vibration-electronic one.

Conservation of mass:

$$\frac{\partial \rho}{\partial t} + \overrightarrow{\nabla}.(\rho \overrightarrow{v} + \overrightarrow{J}) = \dot{\omega} \tag{2.1}$$

Here, $\overrightarrow{v}$ is the velocity vector, $\overrightarrow{J}$ the diffusion vector and $\dot{\omega}$ is the mass production source term.

Conservation of momentum:

$$\frac{\partial \rho \overrightarrow{v}}{\partial t} + \overrightarrow{\nabla}.(\rho \overrightarrow{v} \wedge \overrightarrow{v}) + \overrightarrow{\nabla}.(p\underline{\underline{I}} - \underline{\underline{\tau}}) = \rho \overrightarrow{f} \tag{2.2}$$

Here, E is the total energy, H is the total enthalpy, $\overrightarrow{f}$ is the external force vector and $\underline{\underline{\tau}}$ is the viscous stress tensor defined as follows:

$$\underline{\underline{\tau}} = \mu(\overrightarrow{\nabla}.(\rho \overrightarrow{v} \wedge \overrightarrow{v}) + \overrightarrow{\nabla}.(\rho \overrightarrow{v} \wedge \overrightarrow{v})) + \eta(\overrightarrow{\nabla}.\overrightarrow{v})\underline{\underline{I}} \tag{2.3}$$

Here, μ is the dynamic viscosity and η is the bulk viscosity.

Conservation of energy:

$$\frac{\partial \rho E}{\partial t} + \overrightarrow{\nabla}.(\overrightarrow{v}\rho H - \underline{\underline{\tau}}.\overrightarrow{v}) - = \rho \overrightarrow{f}.\overrightarrow{v} \tag{2.4}$$

The Navier-Stokes equations can be summarized in a compact form as follows:

$$\frac{\partial \overrightarrow{U}}{\partial t} + \overrightarrow{\nabla}.(\overrightarrow{F_i} - \overrightarrow{F_v}) = \overrightarrow{Q} \tag{2.5}$$

Where:

- $\overrightarrow{U} = \begin{pmatrix} \rho \\ \rho \overrightarrow{v} \\ \rho E \end{pmatrix}$: conservative variables
- $\overrightarrow{F_i} = \begin{pmatrix} \rho \overrightarrow{v} \\ \rho \overrightarrow{v} \wedge \overrightarrow{v} + p\underline{\underline{I}} \\ \rho \overrightarrow{v} H \end{pmatrix}$: non-viscous flux vector
- $\overrightarrow{F_v} = \begin{pmatrix} -\overrightarrow{f} \\ \underline{\underline{\tau}} \\ \underline{\underline{\tau}}.\overrightarrow{v} \end{pmatrix}$: viscous flux vector
- $\overrightarrow{Q} = \begin{pmatrix} \dot{\omega} \\ \rho \overrightarrow{f} \\ \rho \overrightarrow{f}.\overrightarrow{v} \end{pmatrix}$: source term

The following sections refer to the formulation of the code for axisymmetric flows using a finite-rate chemistry and multi-species gas.

2.2.2 Implemented equations

In this code, the physical processes such as inviscid gas dynamics, viscous effects, finite-rate chemistry and thermal energy exchange are formulated using a operator-split approach. The Navier-Stokes equations are thus written so that the flux vector and source terms are applied in a loosely couple way:

$$\frac{\partial}{\partial t}\int_V U dV = -\int_S \left(\overline{F}_i - \overline{F}_v\right) \cdot \hat{n}\ dA + \int_V Q dV \tag{2.6}$$

Where U is the vector of conserved quantities.

$$U = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho E \\ \rho e_{ve} \\ \rho Y_s \end{bmatrix} \tag{2.7}$$

Here, ρ is density, u is velocity, E is total energy, e_{ve} is the vibration-electron-electronic energy and Y_s is the species s mass-fraction. The flux vector is divided into two parts: an inviscid one $\overline{F}_i$ and a viscous one $\overline{F}_v$. The inviscid contribution $\overline{F}_i$ for a two-temperature model is a follows:

$$\overline{F}_i = \begin{bmatrix} \rho u_x \\ \rho u_x^2 + p \\ \rho u_y u_x \\ \rho E u_x + p u_x \\ \rho e_{ve} u_x + p_e u_x \\ \rho Y_s u_x \end{bmatrix} \hat{i} + \begin{bmatrix} \rho u_y \\ \rho u_x u_y \\ \rho u_y^2 + p \\ \rho E u_y + p u_y \\ \rho e_{ve} u_y + p_e u_y \\ \rho Y_s u_y \end{bmatrix} \hat{j} \tag{2.8}$$

Here, p_e is the electron pressure. The viscous contribution $\overline{F}_v$ for a two-temperature model is as follows:

$$\overline{F}_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{xx} u_x + \tau_{yx} u_y + q_x \\ q_{x,ve} \\ J_{x,s} \end{bmatrix} \hat{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{xy} u_x + \tau_{yy} u_y + q_y \\ q_{y,ve} \\ J_{y,s} \end{bmatrix} \hat{j} \tag{2.9}$$

Here, τ refers to the axisymmetric viscous stress components, q denotes the heat-flux and J is the diffusion flux. So as to apply th operator-splitting approach, the vector of source terms is separated into geometric, chemical kinetic, thermal energy exchange and radiation contributions:

$$Q = Q_{geom.} + Q_{chem.} + Q_{therm.} + Q_{rad.} \tag{2.10}$$

$Q_{geom.}$ is the source term for axisymmetric geometries:

$$Q_{geom.} = \begin{bmatrix} 0 \\ 0 \\ (p - \tau_{\theta\theta})A_{xy}/V \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.11}$$

Here, A_{xy} is the projected area of the cell in the (x,y) plane. $Q_{chem.}$ is the chemistry source term:

$$Q_{chem.} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \sum_{i}^{N_{mol}} \Omega_i^{VC} + \sum_{j}^{M_{ion}} \Omega_j^{EC} \\ M_s \dot{\omega}_s \end{bmatrix} \tag{2.12}$$

Here, Ω^{VC} is the vibration-chemistry energy exchange source term, Ω^{EC} is the electron-chemistry energy exchange source term, M_s is the molecular weight and $\dot{\omega}_s$ is the mass production source term. $Q_{therm.}$ is the thermal energy exchange source term:

$$Q_{therm.} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \sum_{i}^{N_{mol}} \Omega_i^{VT} + \sum_{j}^{M_{species}} \Omega_j^{ET} \\ 0 \end{bmatrix} \tag{2.13}$$

Here, Ω^{VT} is the vibration-translation energy exchange source term, Ω^{ET} is the electron-translation energy exchange source term. $Q_{rad.}$ is the radiation source term:

$$Q_{rad.} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\nabla.q_{rad} \\ -\nabla.q_{rad} \\ 0 \end{bmatrix} \tag{2.14}$$

2.3 Numerical methods

2.3.1 Finite volume

To solve partial differential equations `Eilmer3` uses a finite volume method. This kind of space discretization is suitable for fluid problems thanks to its conservative aspect and its adaptability to unstructured meshes. This method is based on integral formulation of conservative equations which are applied to straight-edged quadrilateral cells defining a grid. The boundaries of the control volumes are labeled North, East, South, West and the computational node lies at the center of the cell.

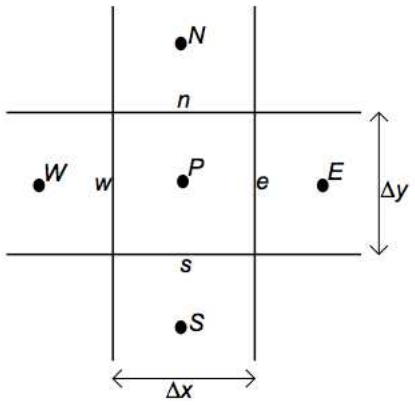


FIGURE 2.1: Control volume

The net flux through the control volume boundary is the sum of integrals over the four control volume faces (six in 3D) and the integral conservation equation is approximated with the following algebraic expression:

$$\frac{dU}{dt} = -\frac{1}{V} \sum_{NESW} (\overline{F}_i - \overline{F}_v).\hat{n}dA + Q \tag{2.15}$$

Where U and Q represent cell-averages values. Then, an operator-splitting approach (as described by Oran and Boris [5]) is used to perform the time-integration of this ordinary differential equation (2.15). Indeed, the physic phenomenon are handled in a decoupled manner and the boundary conditions are imposed. So that we end up with a system of algebraic equations that is then solved by an iterative method.

$$\int_{\delta t} \frac{dU}{dt} dt = \int_{\delta t} \left(\frac{dU}{dt}\right)_{inv.} dt + \int_{\delta t} \left(\frac{dU}{dt}\right)_{visc.} dt + \int_{\delta t} \left(\frac{dU}{dt}\right)_{chem.} dt + \int_{\delta t} \left(\frac{dU}{dt}\right)_{therm.} dt \quad (2.16)$$

Where,

$$\left(\frac{dU}{dt}\right)_{inv.} = -\frac{1}{V} \sum_{NESW} (\overline{F}_i).\hat{n}dA + Q_{geom.} + Q_{rad.} \quad (2.17)$$

$$\left(\frac{dU}{dt}\right)_{visc.} = -\frac{1}{V} \sum_{NESW} (-\overline{F}_v).\hat{n}dA \quad (2.18)$$

$$\left(\frac{dU}{dt}\right)_{chem.} = Q_{chem.} \quad (2.19)$$

$$\left(\frac{dU}{dt}\right)_{therm.} = Q_{therm.} \quad (2.20)$$

2.3.2 Time discretization procedure

As said in the previous section, the set of ODE is obtained by an operator-splitting approach. Thus, each kind of physical process can be handled by the more efficient integration scheme, which is very useful for large chemical kinetic system (see [4] for the chemical model). Indeed, the four incremented terms in 2.16 use the following integration method:

- $\left(\frac{dU}{dt}\right)_{inv.}$: Predictor-corrector method
- $\left(\frac{dU}{dt}\right)_{visc.}$: Explicit Euler method
- $\left(\frac{dU}{dt}\right)_{chem.}$: α -QSS method
- $\left(\frac{dU}{dt}\right)_{therm.}$: 4th order Runge-Kutta-Fehlberg method

For a better understanding, let's consider $\overrightarrow{U}$ containing all the conserved quantities previously described. $\overrightarrow{U}$ is time and space-dependent and can be written as $\overrightarrow{U}(\overrightarrow{x}, t)$ for a one-dimensional problem. Once discretized in space and time, we end up with:
$U_i^n = U(x = x_i, t = t_n)$. The next value of $\overrightarrow{U}$, can be computed via the second order upwind scheme as: $U_i^{n+1} = f(U_i^n, U_{i-1}^n, U_{i-2}^n)$.
Thus, to compute the value of the considered conservative variable in a cell at the next time step (t_{n+1}), we need to know the past values (t_n) in the current and the two previous cells.

2.3.2.1 Time-stepping procedure

Here is a typical sequence of operations needed to update a time-step in `Eilmer3` (written by Potter [1]):

1. compute gas transport due to inviscid flux:
   (a) apply inviscid boundary conditions or exchange data at boundaries for each block as appropriate
   (b) reconstruct the flow field sate on both sides of each interface
   (c) compute the inviscid fluxes $\overline{F_i} \cdot \hat{n}$
   (d) compute the radiative source term $-\nabla \cdot q_{rad}$ for each cell
   (e) integrate Eq. 2.17 over the timestep
   (f) decode the conserved quantities and update the gas-state
   (g) repeat for corrector update
2. compute gas transport due to viscous flux:
   (a) apply viscous boundary conditions at solid walls
   (b) compute the viscous fluxes as $\overline{F_v} \cdot \hat{n}$
   (c) integrate Eq. 2.18 over the timestep
   (d) decode the conserved quantities and update the gas-state
3. compute change of gas state due to chemical reactions:
   (a) compute all chemical source terms
   (b) integrate Eq. 2.19 over the timestep
   (c) decode the conserved quantities and update the gas-state
   (d) redo via smaller subcycles if failed and apply call to equation-of-state more frequently
4. compute change of gas state due to thermal energy exchange:
   (a) compute all chemical source terms
   (b) integrate Eq. 2.20 over the timestep
   (c) decode the conserved quantities and update the gas-state
   (d) redo via smaller subcycles if failed and apply call to equation-of-state more frequently

FIGURE 2.2: Sequence of operations for a time-step update in `Eilmer3` [1]

2.4 Boundary conditions

So as to compute the temperature, pressure and heat-flux on the wall of the probe or the subscale model forebodies, the choice of a suitable boundary condition is very important as it will govern the thermodynamics and chemistry of the simulation.

2.4.0.2 Wall catalicity

In `Eilmer3`, the catalicity can be modeled as follows:

- Non-catalytic wall: This extreme condition makes the wall absolutely indefferent to kinetics. This boundary condition only applies the chemical composition in the closer cell to the wall. Thus, there no production or depletion on the wall interface.
- Super-catalytic wall: It considers the wall is an infinitely efficient catalyst as it applies the freestream chemical composition to the wall interface.

Those two cases are the complete opposite catalicity models, so that the ideal catalicity should be modeled in between those limits: it should consider the reaction kinetics due to the solid surface.

2.4.0.3 Wall Temperature

During this work the wall temperature has been modeled in both following ways:

- Fixed temperature: This boundary condition called `FixedTBC` simply applies a constant temperature at the wall.
- Energy balance: This variable temperature boundary condition implies a dynamic evolution of the wall temperature governed by a surface energy balance at the wall interface. Along a direction normal to the wall interface, this energy balance is expressed as follows:

$$q_{in} + q_{out} = 0 \tag{2.21}$$

The incident heat flux is made of a convection component and a diffusion one, while a radiative equilibrium allows to express the re-radiated heat-flux.

$$q_{in} = q_{conv} + q_{diff} + q_{rad} \tag{2.22}$$

$$q_{out} = \epsilon \sigma T_w^4 \tag{2.23}$$

Where σ is the Boltzmann constant and ϵ is the emissivity. Thus equation 2.21 gives:

$$T_w = \left(\frac{-q_{conv} - q_{diff} - q_{rad}}{\epsilon \sigma} \right)^{1/4} \tag{2.24}$$

Besides, as the heat flux depends on the wall temperature, the radiative equilibrium wall temperature is evaluated via an iterative procedure.

- Ablating boundary condition: This particular boundary condition was implemented in `Eilmer3` during the present work. It models the behavior of the ablator via a pyrolysis gases mass flow at the wall. The implementation and validation of this boundary condition is widely discussed in chapter 4.

2.5 Chemistry model

2.5.1 Chemical species

As said before, the present work is based on Park's chemical model developed in 2000 [4]. It considers 20 species (Table 2.1) divided into:

1. *Air species*: species that are already present in the Earth's atmosphere.
2. *Ablative species*: extra species that are present in the ablative gases.
3. *Boundary layer species*: As the ablative species are likely to be broken down as they travel through the boundary layer and the post-shock layer, one needs also to add these species.

TABLE 2.1: List of chemical species involved according to [4]

Air species	N_2	O_2	NO	N	O	N_2^+	NO^+	N^+
	O^+	e^-						
Ablative species	H_2	CO	C_2H	C_2H_2	C_3	CN		
Boundary layer species	H	C	C^+	H^+				

Note that some species as HCN, CNO, NH or O_2^+ are missing though they play a significant role in particular reentry conditions. Their concentration is very small and O_2^+ has a no significant impact on chemical reactions and radiation. Once the species present in the atmospheric air and the ablated gases are identified, the chemical reactions and their chemical kinetics have to be identified.

2.5.2 Chemical kinetics

For an accurate characterization of the shock layer in atmospheric reentry, a good knowledge of the species mass created by chemical reaction taking place in the considered chemical model is required. The 20 species identified in Park's model follow a 24 reaction scheme (5 dissociation reactions, 12 exchange reactions, 4 electron impact ionization reactions, and 2 associative ionization reactions). All those reactions are governed by the chemical kinetics theory which deals with the reaction rates corresponding to the speed of reaction. `Eilmer3` has to compute these reaction rates so as to describe the evolution of the gases mixture composition and thus the chemical behavior of the flow field.

An usual set of n chemical reactions involving Ns species needs to take into account a parameter r to distinguish the different reaction. It can be described by:

$$\sum_{i=1}^{N_s} \nu_{i,r}[x_i] \rightleftharpoons \sum_{i=1}^{N_s} \nu''_{i,r}[x_i] \tag{2.25}$$

Where x_i is the species mole fraction and $\nu_{i,r}$ and $\nu''_{i,r}$ are the stoechiometric coefficient. Equation 2.25 can be split into two individual relations: one for the reaction going from left to right and the other for that going from right to left. Let us denote by $k_{f,r}$ the forward reaction rate coefficient and $k_{b,r}$ the backward one. Taking into account this transformation and summing over all reactions we end up with the Law of Mass action expressed in terms of mass production for a particular species α:

$$\dot{\omega_s} = M_\alpha \sum_{r=1}^{n} (\nu_{\alpha,r} - \nu''_{\alpha,r}).\{k_{f,r} \prod_{i=1}^{N_s} [x_i]^{\nu_{i,r}} - k_{b,r} \prod_{i=1}^{N_s} [x_i]^{\nu''_{i,r}}\} \tag{2.26}$$

In equation 2.26 the forward and backward reaction rate coefficients can be expressed using the equilibrium reaction rate constant k_c:

$$k_{c,r} = \frac{k_{f,r}}{k_{b,r}} = A.T^{\eta} \exp\left(\frac{-E_A}{RT}\right) \tag{2.27}$$

k_c follows the extended Arrhenius law where E_A is the activation energy , η is the temperature exponent and R is the gas constant. In Park's model, the Arrhenius law was defined as:

$$k(T) = C\ T^n\ exp(-\frac{T_a}{T}), \tag{2.28}$$

where C, n and T_a (activation temperature) have to be defined (see following paragraph).

2.5.2.1 Park 2001

As mentioned before, the model used in this work involves 24 reactions. The following table summarized their features. Note that reactions involving ablative species will be widely discussed in chapter 3.

TABLE 2.2: Chemical reactions considered in Park's model

Reactions	M	C	$T_a[K]$	n	Ref.	**Reactions**	M	C	$T_a[K]$	n	Ref.
Dissociation reactions											
$N_2 + M \rightleftharpoons N + N + M$	All	7.0^{21}	113200	-1.6	[4]	$O_2 + M \rightleftharpoons O + O + M$	All	2.0^{21}	59360	-1.5	[4]
	C	3.0^{22}	113200	-1.6	[4]		C	1.0^{22}	59360	-1.5	[4]
	O	3.0^{22}	113200	-1.6	[4]		O	1.0^{22}	59360	-1.5	[4]
	N	3.0^{22}	113200	-1.6	[4]		N	1.0^{22}	59360	-1.5	[4]
	H	3.0^{22}	113200	-1.6	[4]		H	1.0^{22}	59360	-1.5	[4]
	e^-	3.0^{24}	113200	-1.6	[4]	$H_2 + M \rightleftharpoons H + H + M$	All	2.2^{14}	48300	0.0	[4]
$C_2 + M \rightleftharpoons C + C + M$	All	3.7^{14}	69900	0.0	[4]		H_2	5.5^{14}	48300	0.0	[4]
$CN + M \rightleftharpoons C + N + M$	All	2.5^{14}	87740	0.0	[4]						
Neutral exchange reactions											
$N_2 + O \rightleftharpoons NO + N$	–	5.7^{12}	42938	0.42	[4]	$CN + O \rightleftharpoons NO + C$	–	1.6^{13}	14600	0.10	[4]
$NO + O \rightleftharpoons O_2 + N$	–	8.4^{12}	19400	0.0	[4]	$CN + C \rightleftharpoons C_2 + N$	–	5.0^{13}	13000	0.0	[4]
$CO + C \rightleftharpoons C_2 + O$	–	2.0^{17}	58000	-1.0	[4]	$CO + C_2 \rightleftharpoons C_3 + O$	–	1.0^{12}	41200	0.0	[4]
$CO + O \rightleftharpoons O_2 + C$	–	3.9^{13}	69200	-0.18	[4]	$C_3 + N \rightleftharpoons CN + C_2$	–	1.0^{12}	34200	0.0	[4]
$CO + N \rightleftharpoons CN + O$	–	1.0^{14}	38600	0.0	[4]	$C_3 + C \rightleftharpoons C_2 + C_2$	–	1.0^{12}	16400	0.0	[4]
$N_2 + C \rightleftharpoons CN + N$	–	1.1^{14}	23200	-0.11	[4]	$C_2H + H \rightleftharpoons C_2 + H_2$	–	1.0^{12}	16770	0.0	[4]
Electron impact ionization reactions											
$O + e^- \rightleftharpoons O^+ + e^- + e^-$	–	3.9^{33}	158500	-3.78	[4]	$C + e^- \rightleftharpoons C^+ + e^- + e^-$	–	3.7^{31}	130720	-3.00	[4]
$N + e^- \rightleftharpoons N^+ + e^- + e^-$	–	2.5^{34}	168200	-3.82	[4]	$H + e^- \rightleftharpoons H^+ + e^- + e^-$	–	2.2^{30}	157800	-2.80	[4]
Associative ionization reactions											
$N + O \rightleftharpoons NO^+ + e^-$	–	5.3^{12}	31900	0.0	[4]	$N + N \rightleftharpoons N_2^+ + e^-$	–	4.4^{7}	67500	1.5	[4]

2.5.3 Transport properties and chemistry model

The choice of the chemistry model influences many parameters needed to solve the Navier-Stokes equations as the gas mixture's viscosity, thermal conductivity, and other transport properties. The example of viscosity is a suitable one because its computation for hypersonic flows is not easy. Indeed, in a given gas mixture, all the different particles (atoms, molecules, ions, electrons) collide so that there is an induced momentum transfer. This amount of momentum transfer depends on the electromagnetic forces and thus to the shape and size of the particles involved. The kinetic gas theory implies to model particles as hard-spheres and that is how Maxwell defined the following proportionality relation for the viscosity [6]:

$$\eta_i \propto \frac{(M_i T)^{1/2}}{\sigma_i^2} \tag{2.29}$$

where M, T and σ denote respectively the molecular mass, absolute temperature in K and hard-sphere diameter in $\mathring{A}$.

When the collisions between particles are taken into account, the model of hard-spheres is no longer suitable so that a correction factor is required in equation 2.29. This factor denoted by Ω_V is called collision integral [6]:

$$\eta_i = 26.69 \frac{(M_i T)^{1/2}}{\sigma_i^2 \Omega_V}. \tag{2.30}$$

In a similar way thermal conductivity and mass diffusion can be expressed [7].

Conductivity:

$$\lambda_i = \frac{15}{4} \left(\frac{\eta_i * R_0}{M_i} \right) \left(\frac{4}{15} \frac{C_{pi} M_i}{R_0} + \frac{1}{3} \right) \tag{2.31}$$

Mass diffusivity:

$$D_{ij} = \frac{0.0188 T^{3/4} \sqrt{(M_i + M_j)/M_i M_j}}{p \sigma_{ij}^2 \Omega_{D_{ij}}} \tag{2.32}$$

Thus, those microscopic interactions between particles govern the transport properties of the fluid which are defined as linear combinations of the collision integrals. A variety of terminology and nomenclature is used in literature to describe collision integrals:

- Ω is the dimensional collision integral in m^3/s (SI).
- $\Omega*$ is the dimensionless reduced collision integral, which is obtained by dividing by rigid sphere value.
- $\sigma^2 \Omega*$ is the collision cross-section in m^2 (SI).

Therefore, collision integrals are fundamental inputs for most of the computations during a simulation with `Eilmer3`. Moreover, as there is a collision integral for each pair of chemical species, a N_s-species model requires $1/2 \cdot N_s * (N_s - 1)$ computations. So the present work needs 190 collision integrals as it is based on Park's 20-species chemistry model.In `eilmer3`, collision integrals are entered as 4 coefficients of an order 3 polynomial interpolation in $\log T$ as:

$$\log(\pi \Omega) = A \log^3 T + B \log^2 T + C \log T + D. \tag{2.33}$$

For more details on how to implement collision integrals in `Eilmer3`, the authors refer to [8].

2.6 Radiation transport

High temperatures induced during re-entry flights are responsible for the dissociation of gases. Indeed, the bonds holding the molecules together are broken. With even higher temperatures, the particles are ionized so that they release free electrons. Thus, dissociation and ionization make these particles radiate energy in the flow (radiative heating) and also transfer energy to the surface when entering in collision with it (convective heating). The radiating gas can be either absorptive and gain energy or transparent and emit energy in the flowfield.

2.6.1 Radiation source term

In Equation 2.10 the radiation source term is expressed as the negative divergence of the radiative heat flux vector $-\nabla.\overrightarrow{q_{rad}}$. It can be related to the radiative intensity which is defined by Anderson [9] as: *radiative energy (dE) transferred in the r direction crossing the unit area (dA) orthogonal to r, perunit frequency (dν), per unit time (dt), per unit solid angle (dω).* For a better understanding, see equation 2.34 and figure 2.3 below.

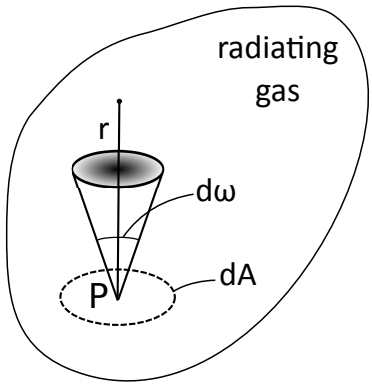


FIGURE 2.3: Radiative intensity

$$I_\nu = \lim_{dAd\omega d\nu dt \to 0} \left[\frac{dE_\mu}{dAd\omega d\nu dt} \right] \tag{2.34}$$

Thus the radiation source term is as follows:

$$-\nabla.\overrightarrow{q_{rad}} = -\nabla. \int_0^\infty \overrightarrow{I_\nu} d\nu \tag{2.35}$$

Equation 2.36 can be expressed with local emission and absorption so that it's more convenient for computational grid applications:

$$-\nabla.\overrightarrow{q_{rad}} = \int_0^\infty \int_{4\pi} \kappa_\nu I_\nu d\omega d\nu - 4\pi \int_0^\infty j_\nu d\nu \tag{2.36}$$

Where κ_ν is the spectral absorption coefficient and j_ν is the spectral emission coefficient.

2.6.2 Implemented models in `Eilmer3`

They are many models available in `Eilmer3` to evaluate the radiative source term. During the present work, simulations were run with the following models:

- Tangent-slab
- Discrete tranfer

2.6.2.1 Tangent-slab

In this approximation, the radiation is supposed to be emitted along each line-of-sight normal to the probe's wall. Thus the problem is reduced to the radiation of an infinitely thin parallel plan to the surface. A correction factor can be applied so as to take into account the curvature of the surface. Moreover, when using a single block for the computational domain this model can use the parallelism of the code which makes it very time efficient.

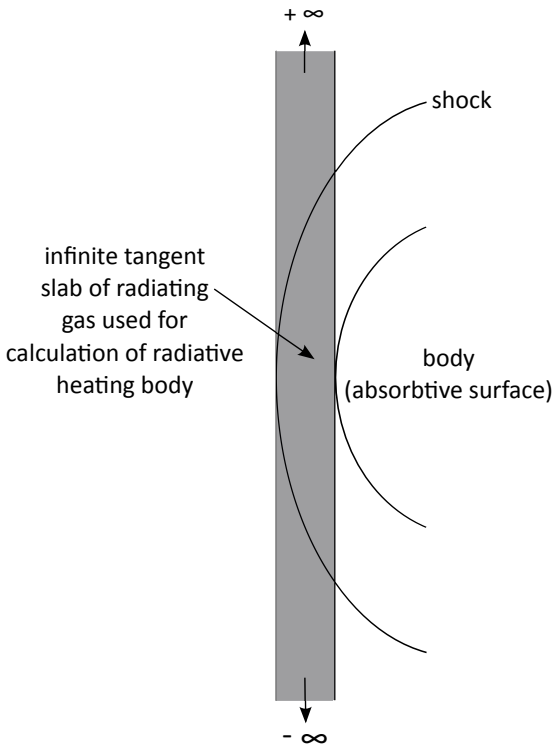


FIGURE 2.4: Tangent-slab approximation scheme [10]

2.6.2.2 Discrete transfer

This model is based on numerical integration of the radiant energy field over direction and space. It uses a 'radiation sub-grid' which is generated via a mapping of the CFD grid. This sub-grid is made of isodirectionally distributed rays over the flowfield. The main parameter to set up is the number of rays. It was a used as an investigation parameter during this work.

2.7 Verification and validation

During the present work, simulations were run considering *Hayabusa* full scale and low scale models with different freestream conditions. This section of validation will deal with a full scale model at a particular reentry trajectory point (H1) corresponding to the radiative peak heating (Table 2.3).

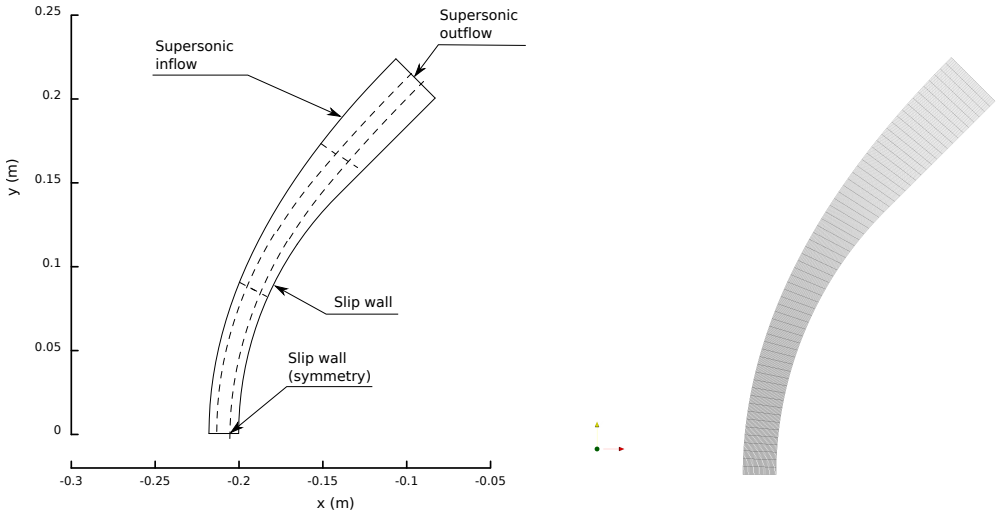
TABLE 2.3: Freestream conditions for H1 (radiative peak heating)

Velocity $[m/s]$	$10.52 \cdot 10^3$
Density $[kg/m^3]$	$4.966 \cdot 10^{-4}$
Temperature $[K]$	257.65
Y_{N2}	0.767
Y_{O2}	0.233

To run the H1 simulation, several modification on the geometry of *Hayabusa* have been made so as to reduce the computational expense. Indeed, the capsule is modeled in two dimensions and the computational domain is reduced to the forebody (the wake region is ignored). Moreover,the case is build as axis-symmetric.

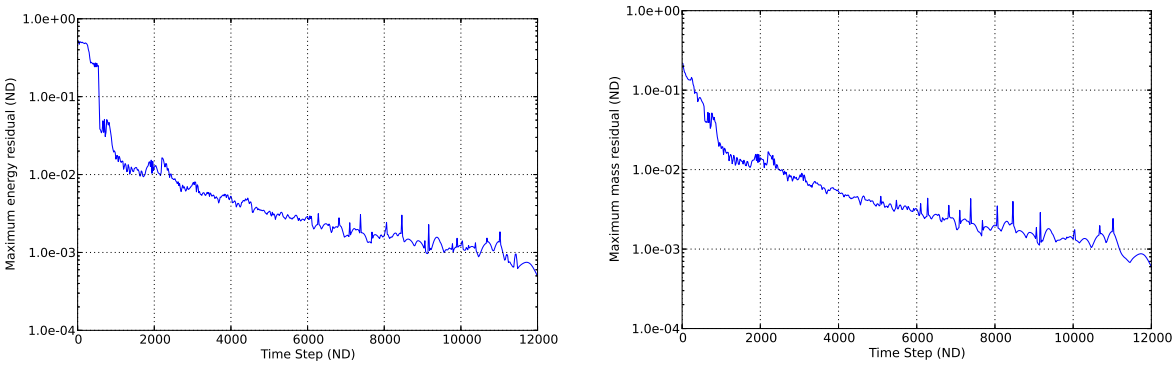
2.7.1 Inviscid simulations

The viscous simulations are initialized with the converged inviscid case run with a coarse 30×30 mesh so as to prevent the solution to diverge when inserting the viscous effect (figure 2.5). To compute the initial inviscid simulation, a 5 body lengths was run on a 9-sub-blocks domain to make use of the parallelism of the code. The convergence of this simulation has been verified via a `Python` file called `get_residuals.py` that computes the maximum energy residuals as a function of time. Those residuals show good convergence as they drop to more than $1 \cdot 10^{-4}$ after the 5 body lenghts (Figure 2.9).



(a) Computational domain (b) One-half resolution grid (30×30 cells)

FIGURE 2.5: Computational domain and grid for inviscid simulations of the subscale Hayabusa model *without* the wake region include [1]



(a) Maximum total energy residuals (b) Maximum total mass residuals

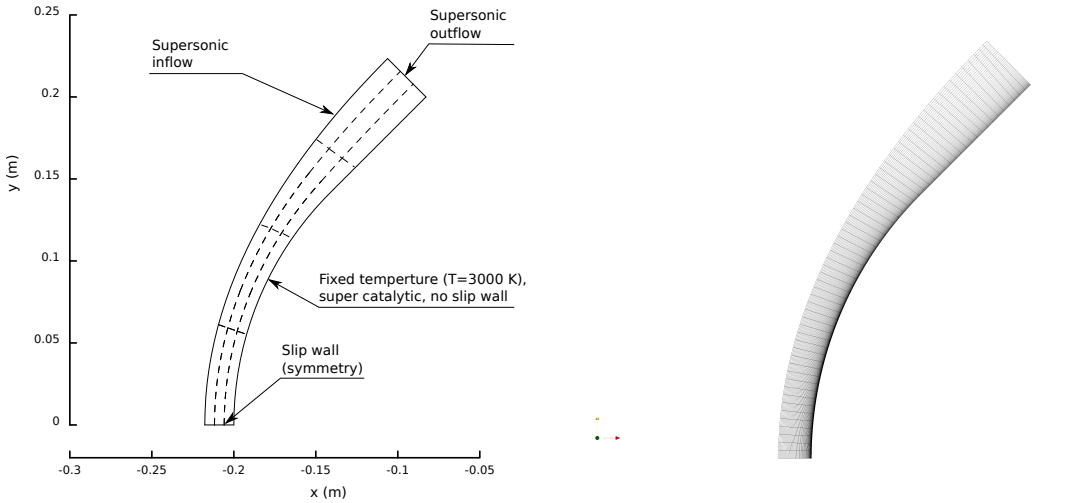
FIGURE 2.6: Maximum energy and mass residuals for inviscid simulation on Hayabusa full scale model

2.7.2 Viscous simulations

Once the flow is well established via the inviscid solution, the viscous effect is added incrementally. The transport properties are incremented at each time step. Thus the viscosity computed in the following way for more stability:

$$\mu_i^{n+1} = \mu_i^n \times F_{inc} \tag{2.37}$$

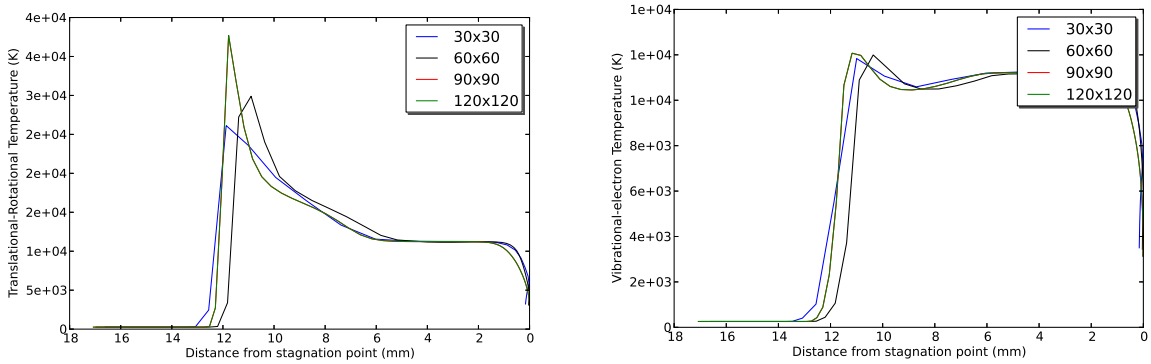
Where F_{inc} is the incremental factor. Here, this factor was set to $1 \cdot 10^{-5}$ and the simulation run over 5 body lengths.



(a) Computational domain (b) One-half resolution grid (60×60 cells)

FIGURE 2.7: Computational domain and grid for inviscid simulations of the subscale Hayabusa model *without* the wake region include [1]

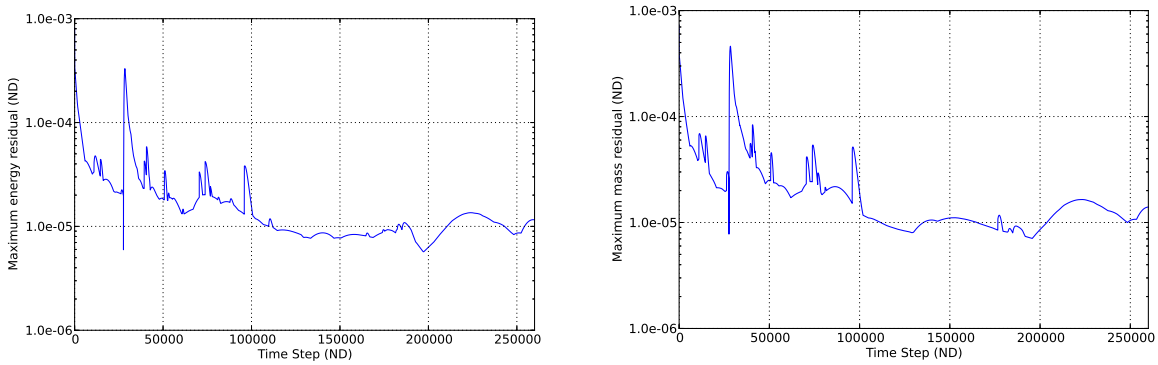
A grid refinement study was lead where the translation-rotation temperature profiles and electron number density profiles along the stagnation streamline where compared for the following four meshes: 30×30, 60×60, 90×90, 120×120 (Figure 2.8). The location of the shock front, the peak in the translational temperature and the peak in the electron number density are suitable resolution criteria. Thus the 60×60 was chosen as it proved to be both efficient and accurate. Again the computational domain was divided into 12 sub-blocks for computational efficiency.



(a) T_{tr} (b) T_{ve}

FIGURE 2.8: T_{tr} and T_{ve} profiles along stagnation line for different meshes

This 60×60 shows good convergence results, again the maximum energy and mass residuals drop to about $1 \cdot 10^{-5}$ after 5 body lengths.



(a) Maximum total energy residuals

(b) Maximum total mass residuals

FIGURE 2.9: Maximum energy and mass residuals for viscous simulation on *Hayabusa* full scale model

2.7.2.1 Gas composition

Figure 2.10 shows the mass fraction of air species during a viscous simulation of *Hayabusa* reentry with H1 conditions. Nitrogen and oxygen are the most represented species after the shock which is located approximately 11 mm before the probe's wall. Moreover the mass fraction of nitrogen is about 4 times higher than the one of oxygen. Besides Y_O and Y_N are at least 10 times higher than any other mass fraction. Note that with a mass fraction of less than $\cdot 10^{-5}$, electrons can be considered as non-existent.

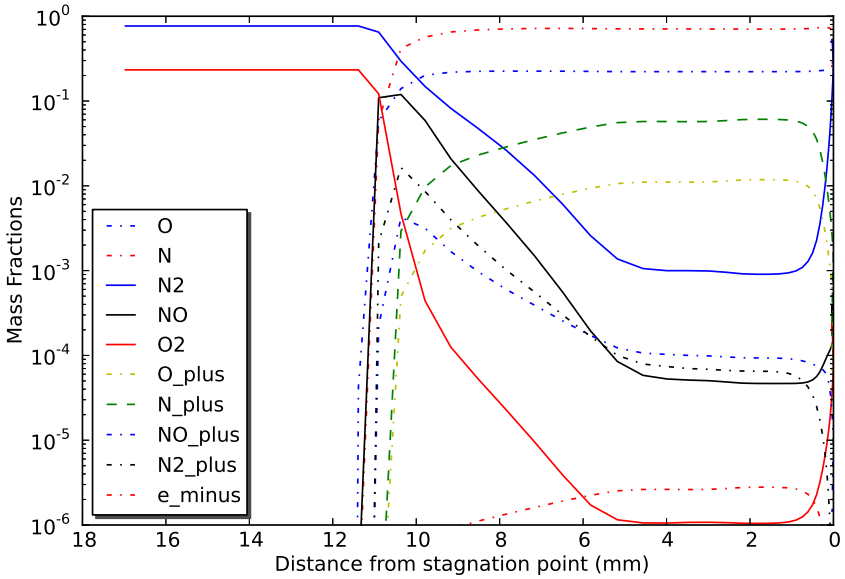


FIGURE 2.10: Air species mass fractions using Park's model (viscous flow, H1)

2.7.2.2 Conductive heat flux

The evolution of conductive heat flux is plotted in the following figure (2.11). As expected, it drops away from the stagnation point. It is due to the shock wave induced by the hypersonic flow. Indeed, the shock wave is vertical and so the most intense in front of the stagnation point.

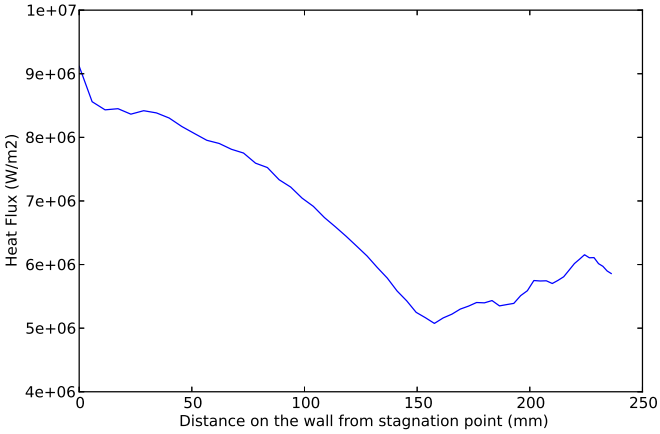


FIGURE 2.11: Conductive heat flux along on *Hayabusa*'s wall (viscous flow, H1)

2.7.3 Diffusion simulations

To take into account mass diffusion, simulations were run with the viscous case (described in the former section) as a starting point. Many models of mass diffusion are implemented in `Eilmer3` but many of them lead to numerical difficulties during the present work. Thus, the constant Lewis number model implemented by Moss, Candler and Suzuki et al was used for the following calculations. In this model, the Lewis number L_e is assumed to be constant so that the effective diffusion coefficients D_i are expressed as follows.

$$D_i = \frac{L_e \mu}{P_r} \tag{2.38}$$

Where P_r is the Prandtl number defined as:

$$P_r = \frac{\mu c_p}{K} \tag{2.39}$$

Where c_p is the total specific heat and K the total conductivity. Note that L_e was set to 1.4 during the present work, as advocated by Moss for reacting air.

2.7.3.1 Gas composition

The gas composition is almost the same than in the viscous case. See figure 2.10 for the air species mass fractions.

2.7.3.2 Convective heat flux

Before, analyzing the evolution of heat fluxes, it is worth clarifying the terminology used during the present work as there is apparently no common agreement on the use of terms when it deals with conductive and convective heat fluxes. A viscous flow will induced a conductive heating denoted by Q_{cond}. Once the effect of diffusion is added, Q_{diff}, we end up with a convective heating, Q_{conv} taking into account both previous heat fluxes (2.40):

$$Q_{conv} = Q_{cond} + Q_{diff} \tag{2.40}$$

The following figure (2.12) shows the evolution of both conductive heating Q_{cond} and convective heating Q_{conv} along *Hayabusa*'s wall under H1 conditions. The convective heat flux is more than twice higher than the conductive one, which would mean that the diffusion would double the total heat flux. Those results are obviously not consistent and it is due to the catalicity boundary condition used during those calculations. Indeed a `SuperCatalyticWBC` was used here so that the freestream chemical conditions are applied to the probe's wall. To be the more accurate, a finite-rate chemistry boundary condition should be chosen when diffusion is taken into account but this boundary condition is still under development.

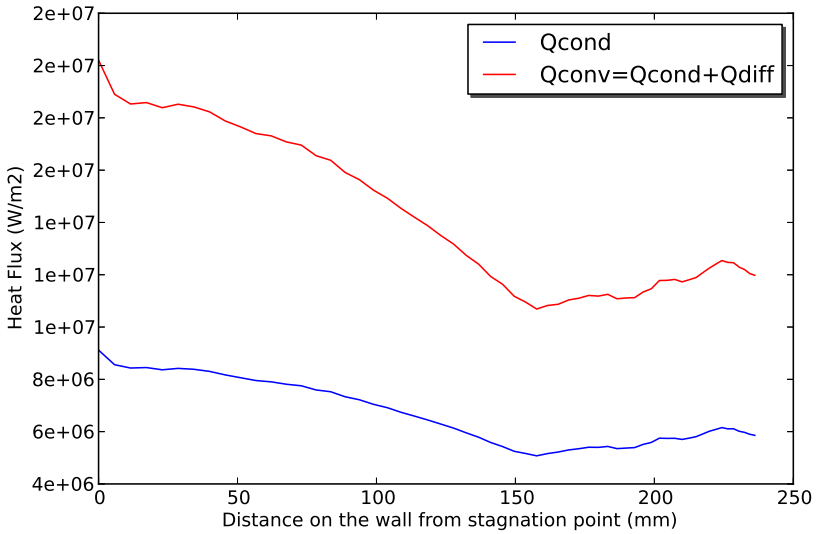


FIGURE 2.12: Conductive and convective heat fluxes on *Hayabusa*'s wall

2.7.4 Empirical correlations

Certain studies [11–13] have been conducted with the aim of formulating an empirical solution to 2.40 for stagnation point convective heat transfer. The ones used in the present study are presented in the following section. Each correlation is based on equation 2.41, with characteristic length L typically taken as the blunt body nose radius, R_N. Equation 2.41 displays a significant increase in convective heat transfer with flight velocity (proportional to total enthalpy) and the square root of the density. There is however a decrease with the square root of nose radius. It is thus evident both fluid properties and vehicle size have a significant effect on surface convective heat transfer.

$$Q_{conv} \propto \sqrt{\frac{\rho}{L}} U_{\infty}^{3} \tag{2.41}$$

A basic correlation was used with a constant of proportionality K equal to $1.74153 \cdot 10^{-4}$.

2.7.4.1 Fay-Riddel

In 1958, Fay and Riddel developed what is now the most commonly used empirical correlation for the prediction of stagnation point heat transfer [11]. In this study, correlations were developed for the convective heat transfer to a blunt body for the case of a reacting gas and no ablation[14]. The conditions considered included either fully catalytic or noncatalytic walls and either frozen or equilibrium boundary layers.

A boundary layer is said to be in chemical equilibrium when the time taken for the control volume to completely cross the boundary layer (τ_f) is significantly greater than the time for the chemical reactions/vibrational energy to approach equilibrium (τ_c)[10]. It is considered to be frozen when the time to achieve chemical relaxation is much longer than that taken by the control volume to traverse the boundary layer ($\tau_f << \tau_c$). Wall catalicity refers to the ability of the wall material to speed up or enhance chemical reactions at the surface, resulting in an increase in aerodynamic heating [10]. A fully catalytic wall catalyses chemical reactions at an infinite rate, thus rendering mass fractions at their local equilibrium values. A catalytic wall provides an effective collision partner to gas constituents such that chemical equilibrium is attained at wall temperature and bulk gas pressure [10]. A noncatalytic wall does not allow for recombination to occur.

Equations 2.42, 2.43 and 2.44 below formulate the empirical solutions for stagnation point convective heat transfer developed by Fay and Riddell in W/m^2.

For an equilibrium boundary layer:

$$(Q_{conv})_0 = 0.76Pr^{-0.6}(\rho_e\mu_e)^{0.4}(\rho_w\mu_w)^{0.1}\sqrt{\left(\frac{dU_e}{ds}\right)_0}(h_0 - h_w)\left[1 + (Le^{0.52} - 1)\left(\frac{h_D}{h_0}\right)\right] \quad (2.42)$$

For a frozen boundary layer with a fully catalytic wall,

$$(Q_{conv})_0 = 0.76Pr^{-0.6}(\rho_e\mu_e)^{0.4}(\rho_w\mu_w)^{0.1}\sqrt{\left(\frac{dU_e}{ds}\right)_0}(h_0 - h_w)\left[1 + (Le^{0.63} - 1)\left(\frac{h_D}{h_0}\right)\right] \quad (2.43)$$

For a frozen boundary layer with a noncatalytic wall,

$$(Q_{conv})_0 = 0.76Pr^{-0.6}(\rho_e\mu_e)^{0.4}(\rho_w\mu_w)^{0.1}\sqrt{\left(\frac{dU_e}{ds}\right)_0}(h_0 - h_w)\left(\frac{h_D}{h_0}\right) \quad (2.44)$$

In the above equations, Pr is the Prandtl number, ρ and μ are the density and viscosity, h is the enthalpy and Le is the Lewis number, which is assumed constant. The Lewis number is estimated to lie in the range of 1.0 - 1.5[15]. The subscripts e and w refer to conditions at the boundary layer edge and the wall respectively while the subscript 0 refers to the stagnation values. The term h_D is the dissociation enthalpy and is calculated as follows,

$$Le = \frac{\rho D_{12} c_p}{k} \quad (2.45)$$

$$h_D = \sum_i Y_i(\Delta h_f)_i^0 \quad (2.46)$$

Where Y_i is the mass fraction of species i and $(h_f)_i^0$ is the enthalpy of formation of species i.

The velocity gradient, $\left(\frac{dU_e}{ds}\right)$ has a significant effect on the solution to the above equations. Therefore the method used to calculate it is extremely important. Fay and Riddell use the one-dimensional momentum equation for a near stagnation point streamline and Newtonian theory to calculate it [11], resulting in,

$$\left(\frac{dU_e}{ds}\right)_0 = \frac{1}{R_N}\sqrt{\frac{2(p_0 - p_\infty)}{\rho_0}} \quad (2.47)$$

where R_N is the nose radius, p_0 and ρ_0 are stagnation pressure and density respectively and p_∞ is the flow pressure.

2.7.4.2 Zoby

Zoby [12] also formulated an empirical correlation for stagnation point convective heat transfer in W/m^2 given in equation 2.48. This expression was developed for air, Argon, Carbon dioxide, Hydrogen and Nitrogen and is valid to an enthalpy potential of $117MJ/kg$[12].

$$(Q_{conv})_0 = K\sqrt{\frac{p_0}{R_{eff}}}(h_0 - h_w) \tag{2.48}$$

where R_{eff} is an *effective* nose radius (explained below) and K is a specific gas constant. For air and Nitrogen, the value of constant, K, is 3.88×10^{-4} and $3.63 \times 10^{-4} m^{-0.5} sPa^{0.5}$ respectively.

2.7.4.3 Tauber and Sutton

Tauber and Sutton[13] formulated an analytical expression, equation 2.49 to calculate stagnation-point radiative heating for re-entry into Terrestrial and Martian atmospheres. In this project, only Earth re-entry has been studied and hence only parameters associated with this case are given below.

$$(Q_{rad})_0 = CR_N^a \rho_\infty^b f\left(U_\infty\right) \tag{2.49}$$

where radiative heating is in W/m^2, nose radius (R_N) in m and density (ρ_∞) in kg/m^3. For Terrestrial cases, the values of C, a and b are as follows,

$$\begin{aligned} C &= 4.736 \times 10^4 \\ a &= 1.072 \times 10^6 U_\infty^{-1.88} \rho_\infty^{-0.325} \\ b &= 1.22 \end{aligned} \tag{2.50}$$

and the velocity function $f(U_\infty)$ is found from table 1 in [13]. Equation 2.49 is valid for earth entries for flight speeds between 10-16km/s, altitudes of 72 to 54km and nose radii varying between $0.3 - 3.0m$.

2.7.4.4 *Effective* Nose Radius

Effective nose radius, R, plays an important role in the above correlations, as convective heating is proportional to $R_N^{-1/2}$. A Newtonian approximation is used to relate stagnation point velocity

gradient to the pressure gradient. This works well if the sonic line sits on the nose of a spherically capped cone but does not if the sonic line sits over the aft shoulders of a blunt body. In this case, *effective* nose radius approaches the radial distance between the stagnation point and sonic line attachment point[14].

2.7.4.5 Results analysis

These correlations have been used to validate our results for both H1 and H2 conditions. A `Python` script has been created during this work in order to evaluate heat fluxes in a given range of reentry velocities according to the former correlations. Note that the Zoby model is still under development so that it was not used as comparison. The following picture shows the values of convective and radiative heat flux for H1 and H2 conditions so that they can be compared to the different correlations.

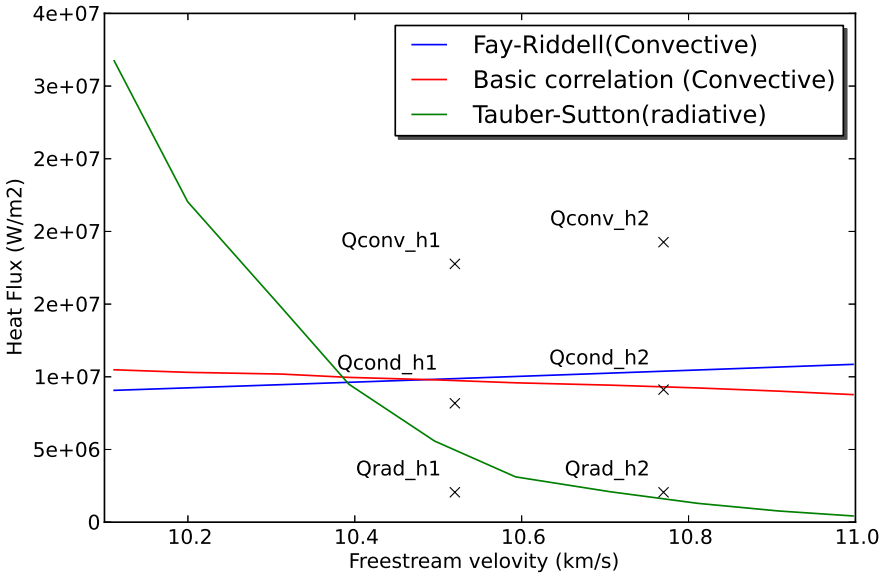


FIGURE 2.13: Conductive and radiative heat fluxes for H1 and H2 conditions

Values for conductive heat flux are a bit lower than those predicted by both basic (equation 2.41) and Fay-Riddell correlations. Indeed, they give an approximation of convective heat flux and not conductive so that the diffusive contribution is missing in our results. Thus, values obtained under H1 and H2 conditions are consistent. Again, the diffusion model used during this work is not relevant: values of convective heating are twice as high conductive ones.
As for the radiation heating, values obtained for H1 and H2 freestream conditions are a bit lower than what is predicted by Tauber and Sutton but still consistent.

Chapter 3

Material response numerical modeling

3.1 `Sacram`: Overview of the code

`Sacram` is a one-dimensional thermal response/ablation code developed by Joshi and based on the work of Amar [3]. Written in `python`, it solves the mass, momentum and energy equations. The mass loss is modeled using Arrhenius law while the mixture energy, gas phase continuity and solid phase continuity are solved using the following laws:

- Fourier's law to model conduction
- Darcy's law to model porous flow
- ideal gas law to model state of pyrolysis gases

Besides, the governing equations that will be widely discussed in the following sections are solved using:

- Control volume finite element spatial discretization method (CVFEM)
- Newton iterative method to solve the non linearities.
- Euler implicit time integrator
- Contracting grid scheme

3.2 Governing equations

The severe thermochemical re-entry conditions that `Hayabusa` has to undergo lead to the use of a carbon-phenolic TPS that is meant to absorb a part of the overall heating. This high heat flux induces an important increase in temperature inside the material which strongly modifies the ablator through two different physical processes: pyrolysis and ablation. Pyrolysis is the high temperature induced reaction that transforms the pyrolysing component of the polymer matrix into gas. This gas reaches the fluid/solid interface as it is free to move toward the porous structure and is injected in the boundary layer depending on in-depth pressure and temperature gradient.

Ablation consists in the removal of the residue that is left once the polymer matrix has been pyrolysed. Indeed, decomposition of carbon fibers can be due to heterogeneous chemical reactions such as oxidation and nitridation, phase change as sublimation or mechanical erosion like spallation. For a better understanding of the overall phenomena, figure 3.1 schemes the decomposition of a TPS.

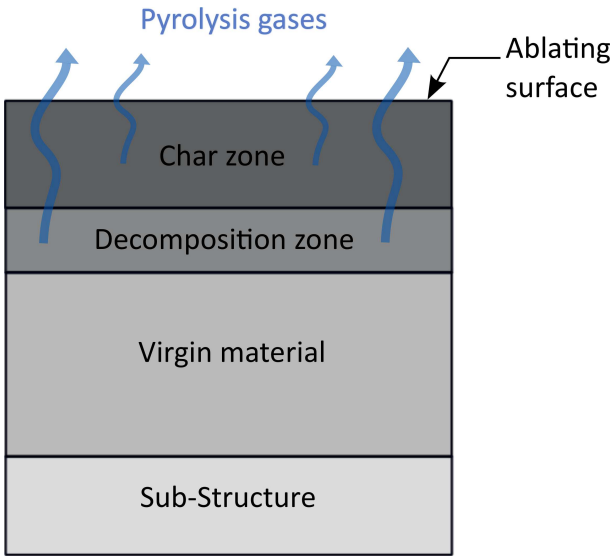


FIGURE 3.1: Decomposing ablative material

3.2.1 Conservation equations

The solid/gas system that represents the ablator, is governed by energy and mass conservation equations. During this work, we assume that pyrolysis gases are in thermochemical equilibrium while the solid and gas are in thermal equilibrium. The integral forms of the governing equations are as follows.

3.2.1.1 Mixture energy

Both solid and gas energy equations can be reduced to the following equation:

$$\frac{d}{dt}\int_V \rho e dV = \int_S \dot{Q}\cdot dA + \int_S \rho h v_m \cdot dA - \int_S \phi \rho_g h_g v_g \cdot dA \tag{3.1}$$

Where the first term is the energy content, the second one is the conduction component, the third one is the grid convection term and the last one is the gas flux. In this equation, subscripts g and m respectively correspond to the gas and the considered mesh cell. Besides, ϕ is the porosity of the material.

3.2.1.2 Solid mass conservation

$$\frac{d}{dt}\int_V \rho_s dV = -\int_V \dot{\omega}_s dV + \int_S \rho_s v_m \cdot dA \tag{3.2}$$

3.2.1.3 Gas mass conservation

Assuming that all the solid that is removed is transformed into gas ($\dot{\omega}_g = -\dot{\omega}_s$) and that the gas fills the entire free space in the porous material, we end up with:

$$\frac{d}{dt}\int_V \phi \rho_g dV = \int_V \dot{\omega}_g dV - \int_S \phi \rho_g v_g \cdot dA + \int_S \phi \rho_g v_m \cdot dA \tag{3.3}$$

3.2.1.4 Mixture mass conservation

Adding equations 3.4 and 3.3 gives a mixture continuity equation:

$$\frac{d}{dt}\int_V \rho dV = \int_S \rho v_m \cdot dA - \int_S \phi \rho_g v_g \cdot dA \tag{3.4}$$

3.2.2 Darcy's law

To be complete, the former set of equations needs a momentum conservation equation. However, the detailed pore structure is not known so that Darcy [16] advocated to express the volumetric flow rate of a fluid through a fully saturated medium as follows:

$$Q = -A\frac{\kappa}{\mu}\Delta P \tag{3.5}$$

Where A is the cross sectional area of the medium, κ is the permeability, μ is the dynamic viscosity and ΔP is the local pressure gradient. Note that Darcy's law is only valid for laminar flows, but the notion of low Reynolds numbers is not that straightforward.

$$Re = \frac{|v'|\rho D_H}{\mu} \tag{3.6}$$

Where, v' is the superficial velocity expressed in equation 3.7 and D_H is the hydraulic diameter. Thus, Darcy's law is meant to model pyrolysis gases flow through the ablator as the velocities are low enough. If the velocity was higher, a quadratic term could be added to the Darcy's law and give Darcy-Forchheimer relationship [17].

$$v' = \frac{Q}{A} = -\frac{\kappa}{\mu}\Delta P \tag{3.7}$$

3.2.3 Implemented equations

The one-dimensional form of the mixture energy conservation equation 3.1 can be expressed in a semi-discrete form on a nodal control volume as:

$$\begin{aligned} 0 = {} & (\dot{Q}A)_{i+1/2} - (\dot{Q}A)_{i-1/2} - (\phi\rho_g v_g A)_{i-1/2} + (\phi\rho_g v_g A)_{i+1/2} + (\rho v_m A)_{i-1/2} - (\rho v_m A)_{i+1/2} \\ & + \frac{d}{dt}\int_{z_{i-1}}^{z_i} \rho e A dz + \int_{z_i}^{z_i} \rho e A dz \end{aligned} \tag{3.8}$$

Here, we assumed that the source terms was the difference between the outflow terms and the inflow terms plus the rate of change of content. Each term of the semi-discrete form of the mixture energy equation is discretized in the following sections. For a better understanding, see picture below:

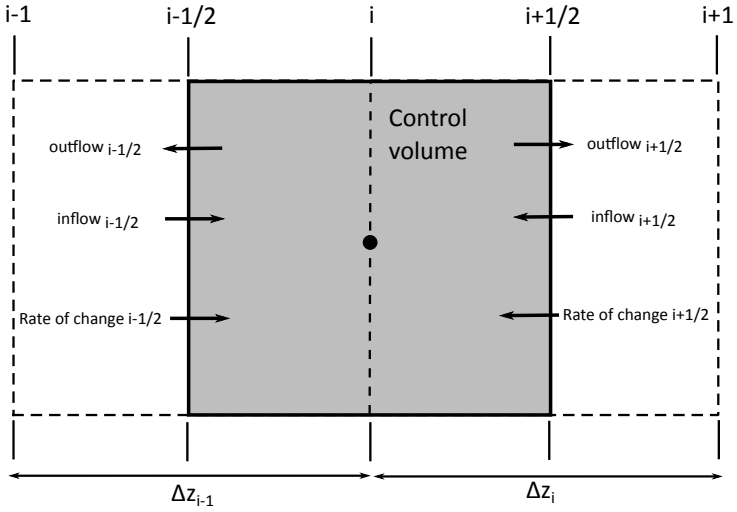


FIGURE 3.2: Mixture energy balance on a control volume

3.3 Numerical methods

3.3.1 Space discretization

The one-dimensional ablation code used in during this work, uses a control volume finite element spatial discretization method (CVFEM). This methods consists in defining a control volume for each node of the one dimensional domain. Besides, boundaries always coincide with the center element as each nodal control volume has two sub-control volumes (left and right) denoted as SV_{i-1} and SV_i in the following scheme. In this kind of method, physics properties are associated with a node. Besides, `Sacram`'s framework contains a contracting grid scheme to model the moving grid because of ablation, but it is still under development so that it was not used during this work and recession was always set to zero. Therefore, ablation does not refers to mechanical removal and recession in the present work, as no residue removal was taken into account yet.

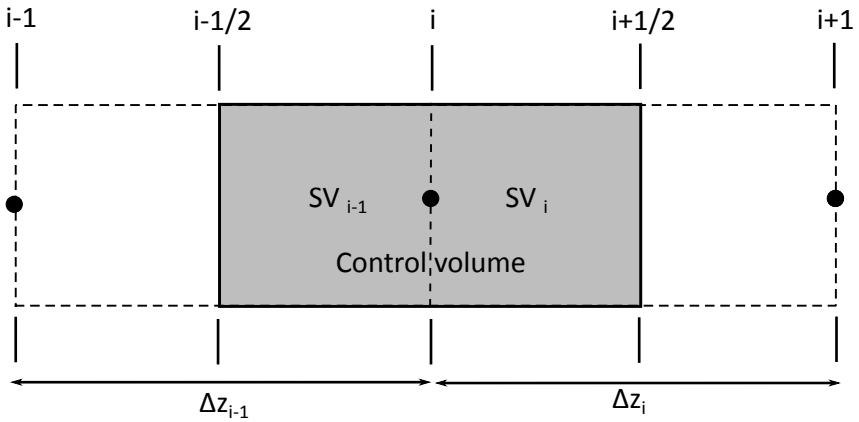


FIGURE 3.3: Space discretization scheme according to CVFEM

3.3.2 Solution procedure

Governing equations detailed in the former sections, are solved via a sequential method. Indeed, each conservation equation is solved independently at each time step. For example, while the semi-discrete mixture energy equation is solved, temperatures are dependent variable whereas values of solid and gas density and gas velocity are constant. As advocated by Amar [3], Joshi used the following procedure:

1. Calculate nodal temperatures from the iterative solution of the energy equations while keeping constant nodal values of solid and gas density and gas velocity

2. Calculate nodal solid densities solving the solid phase continuity and mass loss equations while keeping constant nodal values of solid and gas temperatures.

3. Calculate nodal gas densities solving the gas phase continuity equation while keeping constant nodal values of solid and gas temperatures and densities.

FIGURE 3.4: `Sacram` solution procedure

3.3.2.1 Newton's method

As mixture energy and gas phase continuity equations are nonlinear, a Newton's iterative method was used here to solve the set of equations. It can be expressed via a residual formulation as:

$$[J]\,\Delta T = R \tag{3.9}$$

Where $[J]$ is the linear system's Jacobian matrix also called sensitivity matrix and is tridiagonal, ΔT is the correction vector containing the nodal temperatures and R is the residual matrix. The system being tridiagonal, an iterative method is lead until convergence of residuals. Here is an example of a four nodes one dimensional domain at iterative step n:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{bmatrix}^n \cdot \begin{bmatrix} \Delta T_1 \\ \Delta T_2 \\ \Delta T_3 \\ \Delta T_4 \end{bmatrix}^{n+1} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}^n \tag{3.10}$$

3.4 Boundary conditions

3.4.1 Surface mass balance

Assuming that there is no mechanical removal (spallation or liquefaction), for a given species i, we have the following mass balance at the interface:

$$\rho D_{ij} \frac{\partial T_i}{\partial n}|_w + \dot{m}_{c,i} + \dot{m}_{g,i} = (\rho_i v_n)_w + \sum_{r=1}^{N_s species} \dot{\omega}_{i,r} \tag{3.11}$$

Where $\dot{m}$ is the mass blowing rate which has two component: $\dot{m}_c$ which is due to the char and $\dot{m}_g$ which is due to the pyrolysis gases. Besides, $\dot{\omega}_{i,r}$ corresponds to the mass production of species i during surface reaction r. At the interface, mass fluxes are leaving (right and side of equation 3.11) because of blowing and chemical reaction while other mass fluxes are entering the solid domain (left hand side) because of diffusion, injection of ablation species and injection of pyrolysis gases.

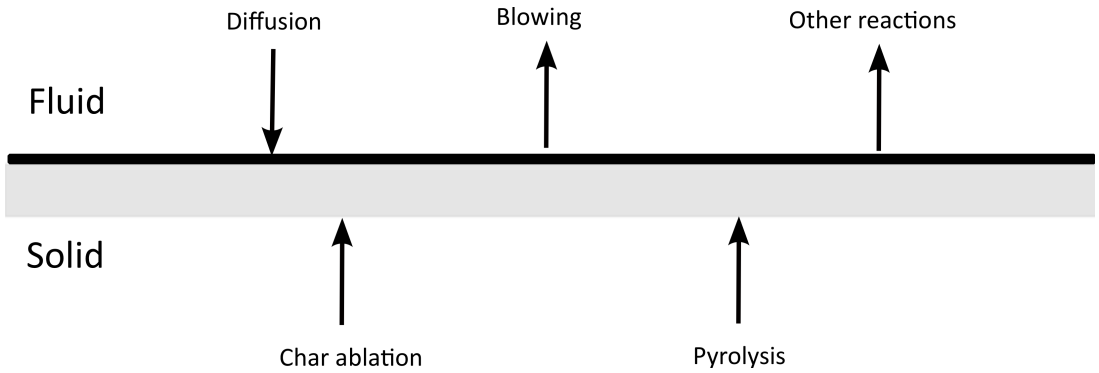


FIGURE 3.5: Surface mass balance at the solid/fluid interface

Note that the sum of equation 3.11 over all species gives the following mass conservation equation. It was used in the ablating boundary condition file called `udf-ablation.lua` during this work.

$$\dot{m}_c + \dot{m}_g = \rho_w v_w \tag{3.12}$$

3.4.2 Surface energy balance

At the interface, considering the balance between entering and leaving energy fluxes, we end up with:

$$Q_{cond} + Q_{diff} + Q_{rad,in} + \dot{m}_c h_c + \dot{m}_g h_g = Q_{rad,out} + \rho_w v_w h_w \tag{3.13}$$

Where the blowing energies are computed by multiplying the mass flow rate $\dot{m}_i$ by the enthalpy h_i.

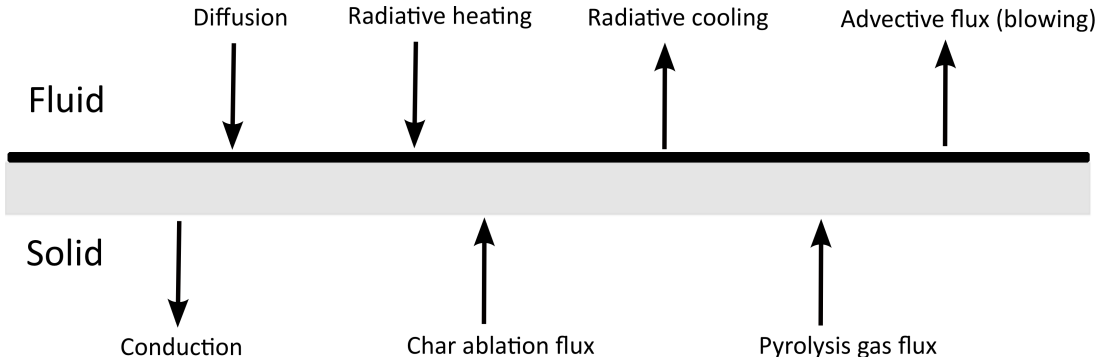


FIGURE 3.6: Surface energy balance at the solid/fluid interface

3.5 Material model

To make the set of equations complete, a modeling of the studied material is required to predict the gas/solid mixture's behavior. As the TPS material is porous,the gas is free to flow through it so that we can express the overall density as:

$$\rho = \phi\rho_g + \rho_s \tag{3.14}$$

Where ϕ represents the porosity, and the subscripts g and s correspond to the gas and the solid. The porosity represents the gas volume fraction: ratio between the pore volume and the mixture volume. Moreover, as said in the overview of `Sacram`, the thermodynamic state of pyrolysis gases is defined by the perfect gas law:

$$P = \rho_g \frac{R}{M_g} T_g \tag{3.15}$$

The solid material is made of carbon phenolic and has two main components: the resin and the binder. Thus it is usual to make the following modeling assumption for the resin/binder composite: it is composed of three different "materials", A, B and C. They do not correspond to actual materials but this decomposition comes from experimental results (thermogravimetric analyses: TGA). Indeed, the phenolic resins tend to follow a two stages decomposition represented by A and B. The binder is modeled by material C. Therefore, the solid density is as:

$$\rho_s = \Gamma(\rho_A + \rho_B) + (1 - \Gamma)\rho_C \tag{3.16}$$

Where Γ is the resin volume fraction in the virgin composite. Besides, the material model is based on the fact that all decomposed solid is transformed into gas so that we usually admit the decomposition model:

$$Virgin \Longrightarrow Char + Gas \tag{3.17}$$

Note that, during reentry, the composite is in an intermediate state between the virgin state and the fully charred one. Moreover, the mass loss is irreversible and can be modeled by an Arrhenius law:

$$\frac{\partial \rho_i}{\partial t} = -k_i \rho_{v_i} \left(\frac{\rho_i - \rho_{c_i}}{\rho_{v_i}} \right)^{\psi} e^{-E_i/RT} \tag{3.18}$$

Where, i corresponds to either A, B or C. k is the pre-exponential factor, E is the activation energy, R is the universal gas constant and subscripts c and v respectively correspond to char and virgin. TGA only allow to know thermophysical properties for virgin and fully charred states so that transitional states are obtained by interpolation using the extent of reaction β defined as:

$$\beta = \frac{\rho_v - \rho_s}{\rho_v - \rho_c} \tag{3.19}$$

Here the virgin and char densities are set as constant. Thus it is possible to express the solid density as:

$$\rho_s = (1 - \beta)\rho_v + \beta \rho_c \tag{3.20}$$

`Sacram` uses the following parameters in order to determine the mixture properties like specific heats or thermal conductivities: the virgin and char mass fractions are given as:

$$y_v = \frac{\rho_v}{\rho_v - \rho_c} \left(1 - \frac{\rho_c}{\rho_s} \right) = \frac{\rho_v}{\rho_s}(1 - \beta) \tag{3.21}$$

$$y_c = 1 - y_v = \frac{\rho_c}{\rho_s}\beta \tag{3.22}$$

3.5.1 Model properties

Virgin and char mass fractions allows us to define thermal and pysical properties for a given intermediate state. Indeed, let us denote by X a given thermophysical property of the solid phase and X_v and X_c its values for the virgin and char. It is possible to expre X as:

$$X = y_v X_v + (1 - y_v) P X_c \tag{3.23}$$

A given property Z of the mixture can be expressed in the same way:

$$Z = y_g Z_g + \frac{\rho_v}{\rho}(1 - \beta) Z_v + \frac{\rho_c}{\rho} \beta Z_c \tag{3.24}$$

3.6 Material response calculations with `Sacram`

3.6.1 Using `Sacram`

The one dimensional material response code used during this work is all written in `python` and uses three different scripts and two different properties tabs.

- `gas/solid.py`
- `mixture_energy.py`
- `sacram.py`
- `gas_props.data`
- `solid_props.data`

An overview of `Sacram` is given in figure 3.7 for a better understanding. The `gas/solid` module contains the properties of the gas and solid different phases. The initial values are set with those of a full virgin composite material (TACOT) and are modified at each time step of the integration according to the material model discussed in the former section. The `mixture_energy` script calculates the terms to be input in the jacobian tridiagonal matrix for each term of the semi-discrete mixture energy equation (3.8). For further details, see Amar's work [3]. The main module, `sacram` integrates in time the mass loss and mixture energy equations calling the `mixture_energy` module and getting the data from the `gas/solid` module.

Here is a list of inputs and parameters to set so as to launch a simulation on:

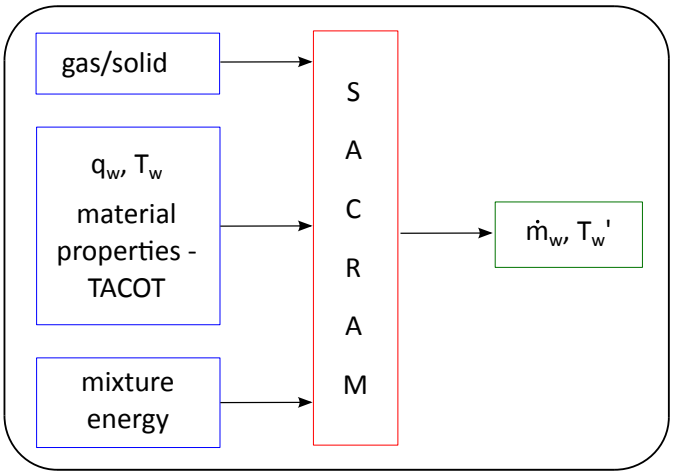


FIGURE 3.7: `Sacram` scheme

- Material properties: they are set according to TACOT database.
- One dimensional geometry: Set the length of the solid domain and the number of nodes.
- Boundary conditions: Set T_w coming from `Eilmer3` and final Q_w. The heat flux values are actual values coming from flight data (JAXA) and are set a each time step.
- Convergence criteria: Set the number of maximum iterations and precision criteria for the temperature to add in the jacobian system's term.

3.6.2 TACOT

`Sacram` was validated on the TACOT test case which is a code-to-code comparison case formulated by NASA. TACOT stand for Theoretical Ablative Composite for Open Testing and is not a real material but rather a set of material properties of a carbon-phenolic composite that enables to compare different codes. Note that many of TACOT's properties are similar to the available data on PICA wich is a low density carbon-phenolic ablator with high porosity. The properties used during this work are summarized in the following tables:

TABLE 3.1: Material properties of TACOT

Parameter	Value
L_0	$0.05\ m$
ρ_c	$220\ kg/m^3$
ρ_v	$280\ kg/m^3$
K_v	$1.6 \cdot 10^{-11} m^2$
K_c	$2 \cdot 10^{-11} m^2$

TABLE 3.2: Arrhenius law constant for A, B and C

Component	A	B	C
$\rho_{v,i}$ (kg/m^3)	300.0	900.0	1600.0
$\rho_{c,i}$ (kg/m^3)	0.0	600.0	1600.0
k_i (s^{-1})	$1.3 \cdot 10^4$	$1.2 \cdot 10^4$	0
E_i/R (-)	3.0	3.0	0.0
T_{reac} (K)	333.33	555.56	5555.56

3.7 Verification and validation

As the version of `Sacram` written by Joshi was modified during this work, calculations were launched on the test case that was used to validate it during Joshi's work. This NASA test case was performed using the properties of TACOT (tables 3.1 and 3.2). In this test case,a known heat flux is assigned to the ablating surface whose temperature is set constant to 300 K. The heat flux is ramped up from 0 to $0.45 \cdot 10^6\ W \cdot m^{-2}$ in 0.1 s, maintained constant up to 60 s, then ramped down to 0 W/m2 in 0.1 s, and finally the material is left to cool down up to 120 s. There is no surface recession in this test case and pyrolysis gas and solid material are assumed to be in thermal equilibrium.

Temperatures probes are located at different depths: x0 =0m, x1 =0.001m, x2 =0.002 m, x3 =0.004 m, x4 =0.008 m. A rise in temperatures is observed for all thermocouples. Their evolution is accurate as it is similar to what Joshi obtained with its version. As for the mass flow rate of pyrolysing gases, there is peak at the beiginning due to the temperature shock but it then tapers down to a constant value of $1 \cdot 10^{-2}\, Kg \cdot m^2 \cdot s^{-1}$ which is relevant before dropping to a non-significant value. Thus, though changes were made in the framework of `Sacram` during this work, it is still accurate so that is can be used for the coupled approach.

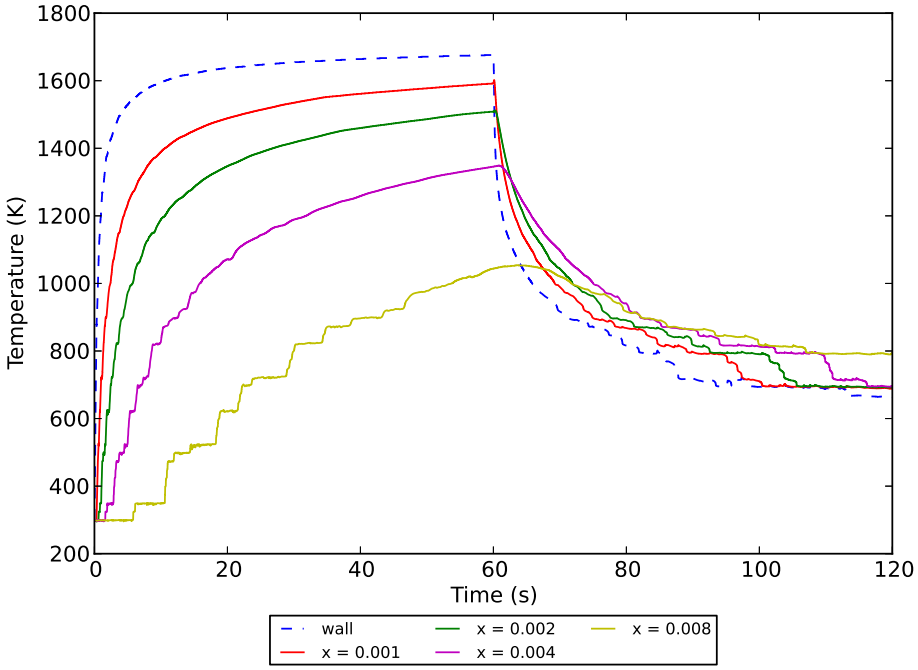


FIGURE 3.8: Temperatures evolution at different depths for `Sacram` ablation test case

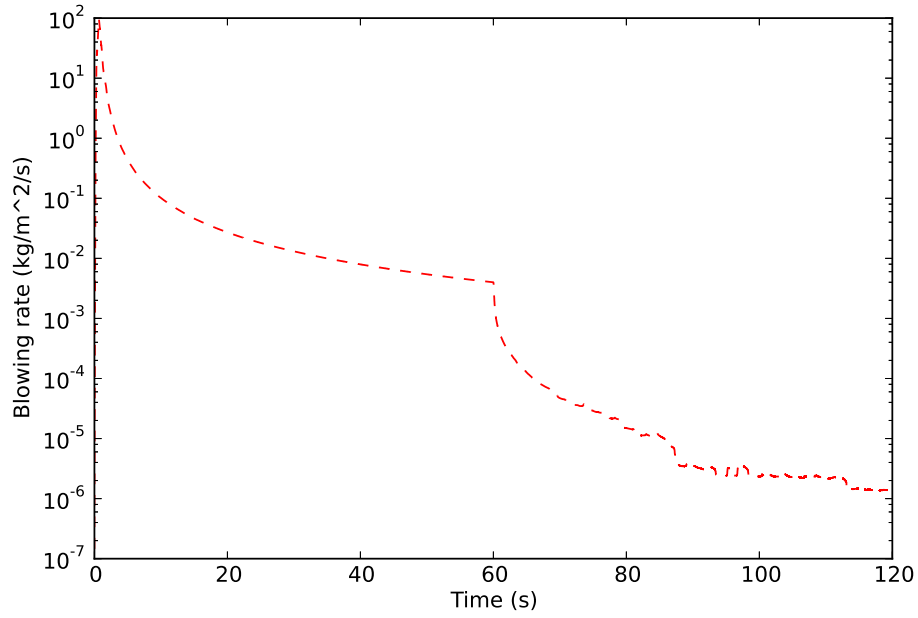


FIGURE 3.9: Pyrolysing gases mass flow rate during `Sacram` ablation test case

Chapter 4

Multiphysics coupling

The present work is a preliminary study of the ablation/radiation/flowfield coupling for atmospheric reentry vehicles. Thus it focuses on the thermal exchanges between the solid body and the fluid domain so as to predict the behavior of the TPS. As this thermal exchange can be a combination of convection, diffusion and radiation, a multiphysics coupling approach has to be chosen so as to solve the set of partial differential equations taking into account the boundary conditions.

4.1 Coupled approach

During this study, a coupling of the CFD code `Eilmer3` and the ablation material response code `Sacram` have been used for the study of *Hayabusa* reentry. There are several kind of multiphysics coupling algorithms, divided into two main categories:

- Monolithic algorithm: Here, all physics processes are handled in a single numerical problem. Both set of partial differential equations are solved simultaneously in the same domain. Thus, this kind of algorithm needs the time steps of each physics problem to be equal.
- Partitioned algorithm: This kind of coupling method uses different domain for each physical process. Thus, each physical problem is solved separately and the data are exchanged through boundary conditions at the interface. Therefore, different discretization method and grid refinement can be applied in each domain which makes this method suitable for problems which involve different time steps.

In atmospheric reentry applications, the typical time step to compute the flow state is much smaller than the time step needed to solve the heat conduction into the solid. Thus, a partitioned algorithm seems to be suitable to handle this problem even if it implies the application of the smaller time-step in both continua, which highly increases the computational time. That is why a modified version of a usual partitioned algorithm has been used during this work.

4.1.1 Partitioned algorithms

As the discretization used in `Eilmer3` is explicit and the one used in `Sacram` is implicit, the coupling algorithm is considered as explicit during this work. Moreover, for a better understanding, a coupling will be qualified as 'strong' if the algorithm ensures energy conservation at the interface and 'loose' if it does not.

4.1.1.1 Loose coupling

This kind of algorithm is based on the simple exchange of data between the different domains which makes it straightforward. It can be schemed as follows:

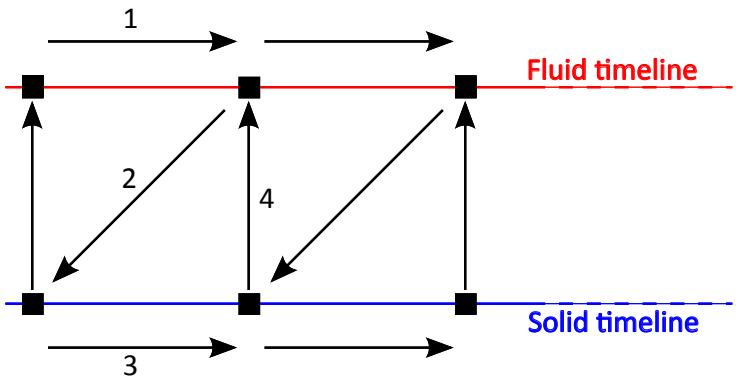


FIGURE 4.1: Loose partitioned coupling algorithm

Assuming the state of both the fluid and the solid domain and the boundary conditions, $BC_{s,n}$ for the solid and $BC_{f,n}$ for the fluid, are known at time t_n, this algorithm can be divided into four steps:

1. The state of the fluid is computed at time t_{n+1} using the boundary condition at the wall $BC_{s,n}$
2. The new boundary condition $BC_{f,n+1}$ is passed to the solid solver
3. The state of the solid is computed at time t_{n+1} using $BC_{f,n+1}$.
4. Pass the value of $BC_{f,n+1}$ to the fluid solver

This kind of algorithm is considered as 'staggered' as both sets of partial differential equations are integrated separately. A said before, the time steps of both domain being different, as staggered algorithm may not be stable. Thus a modified version known as 'sub-iterated loose coupling' was used here. It takes into account the characteristic time difference by performing several sub-iterations with the fluid solver while only one is performed in the solid domain. Therefore, step 1 in figure 4.1 is now considered as a multiple sub-iterations step.

4.1.2 Hypersonics reentry application

During the reentry in the earth atmosphere at hypersonic velocity, *Hayabusa* capsule has to undergo a very high heat load which can be increased by a shock layer radiation under particular flight conditions. This global heating impacts the Thermal Protection System (TPS) which induces ablation processes (e.g. pyrolysis) that also have an influence on the flowfield. Thus, the TPS has to be efficient enough to prevent the payload to be damaged, but at the same time, its weight should be optimized. Therefore, numerical simulations of heat fluxes have to be performed. For this simulation to be accurate, a coupling approach has to be chosen, so that both the response of the solid and the problem in the fluid domain interact.

4.1.2.1 Ablation/flowfield coupling

The ablation of the TPS is governed by the heat fluxes it has to undergo. Processes responsible for this ablation (surface chemistry and in-depth thermal decomposition) also have an impact on convective and radiative heating as ablative products injected in the flowfield tend to thicken the boundary layer and thus reduce the heat flux to the surface. As said before, this work is based on an innovative partitioned algorithm developed by Joshi (ref). The trajectory of *hayabusa* reentry is discretized into trajectory points. At each one of them steady-state simulations are run. Meanwhile, the material response of the TPS is time-accurate during the whole trajectory. Note that, as `Sacram` is a one-dimensional code, it has to be run at each cell of *Haybusa*'s wall. Here are the numerical tools used for this coupled approach:

- `Eilmer3`: Flowfield solver described in chapter 2.
- `Sacram`: One dimensional material response code described in chapter 3.
- `cea2PT`: `Python` script which computes gas composition of a complex mixture.
- `udf-ablation`: Ablating boundary condition developed during this work and embedded in the framwork of `Eilmer3`

Figure 4.2 shows a diagram of the procedure followed for a single trajectory point. For a better understanding of this procedure, table 4.3 gives the detailed steps for a particular trajectory point TP_n:

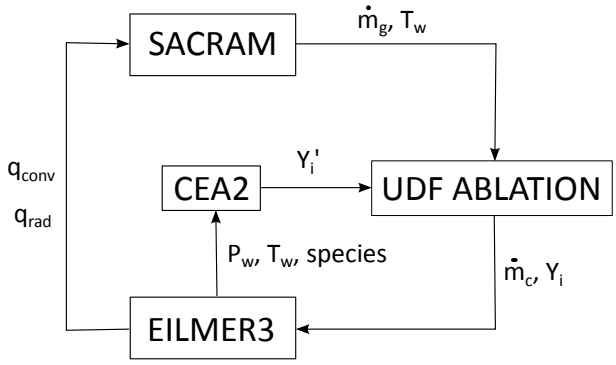


FIGURE 4.2: `Eilmer3-Sacram` coupling diagram. This procedure has to be run until convergence is achieved

- `for` $TP_n = 1, ..., N$
  - `k=0`
  - `while do`
    - Calculate the steady-state solution of the fluid domain with `Eilmer3` at trajectory point TP_n
      - if $k = 0$: use a radiative equilibrium boundary condition with no finite rate chemistry ablation reactions.
      - if $k > 0$: use $\dot{m}_{g,i}$ and T_w computed at iteration `k-1` on the boundary.
    - Pass values of $\overrightarrow{q_{tot}}$ and p_w on the interface to `Sacram`
    - Integrate in time the solid response domain from time t_0 to t_n using time-accurate conditions for $\overrightarrow{q_{tot}}$ and p_w from flight data.
    - Pass the values of P_w and T_w calculated by `Sacram` to `cea2PT` and the chemical species list to calculate the gas composition (Y_i').
    - Pass the values of Y_i' ,T_w and $\dot{m}_g$ to `udf-ablation` to compute values of Y_i and $\dot{m}_c$.
    - Pass the value of $\dot{m}_c$ and Y_i and T_w to `Eilmer3`
- end

FIGURE 4.3: Ablation/flowfield partitioned algorithm

4.1.2.2 Radiation/flowfield coupling

In particular reentry conditions, radiation can be important so that, there is a strong coupling between the flowfield and the radiating gas mixture. One main phenomenon impacts the probe's wall heat fluxes: radiation cooling. As the shock layer is not adiabatic, it can absorb radiant energy so that the flow is cooled and its enthalpy is reduced. Thus, the radiation cooling modifies all components of the overall heat transfer to the vehicle surface. Radiative heat flux decreases as it depends on the temperature which is highly reduced in the shock layer while convective heat flux is reduced as the temperature of the boundary layer is also lower.

When radiation energy source term is important enough to consider it in the total energy flux incident on the vehicle, the Goulard number Γ is a relevant parameter to compute. It gives an estimation of the radiation/flowfield coupling:

$$\Gamma = \frac{2q_{rad}}{\frac{1}{2}\rho_\infty u_\infty^3} \tag{4.1}$$

where q_{rad} is the incident radiative heat flux at the stagnation point, ρ_∞ is the freestream density and v_∞ is the freestream velocity. Thus, the Goulard number corresponds to the ratio between the radiative flux through the shock layer and the total freestream energy. For a given set of reentry conditions, there are two possibilities:

- $\Gamma > 0.01$: Radiation/flowfield has to be taken into account.
- $\Gamma < 0.01$: Radiation/flowfield phenomena are not important enough to be taken into account.

4.1.2.3 Ablation/radiation coupling

The injection of ablatives species in the boundary layer during reentry tends to increase the thickness of the boundary layer and its absorption. Therefore, the a part of the radiative heat flux is absorbed. This physico-chemical process is called 'blockage effect'.

4.2 Equivalent ablative boundary condition

In order to model and simulate the reentry of a vehicle with ablator, both the flow solver and the material response code have to be coupled. The main step of this coupling procedure consists in solving the flow in `Eilmer3` with an implemented boundary condition that accounts for ablation.

4.2.1 Methodology

This equivalent boundary condition consists in blowing a gas mixture representing the ablated material into the flow domain, at constant mass flow rate and wall temperature [18]. Though we assumed chemical nonequilibrium in the flow domain in the present work, this boundary condition is based on a chemical equilibrium at the wall. The injected mass flow is set as normal to the wall while the tangent boundary condition is an usual slip condition.
The following method is specific to both the flow solver and the material response code used during this work, so that it is inherent to the discretization scheme described in the previous section.

4.2.1.1 Ghost cells

Solving the flow in a cell with the finite-volume methods implies a balance of fluxes so that the boundary conditions also have to be set in terms of fluxes. Those boundary conditions come from the neighboring cells. Thus, "ghost cells" are introduced. They are considered as "ghost" as they don't belong to the studied flow domain but are located behind its boundaries. The use of ghost cells is needed here to ensure the required flux between the boundary and its first neighbor called "control cell". As said in the previous chapter, the discretization scheme is a second order one so that two ghost cells are required to determine the values in the control cell. The following figure (4.4) shows that there is no flux in the orthogonal direction to the boundary of ghost cells.

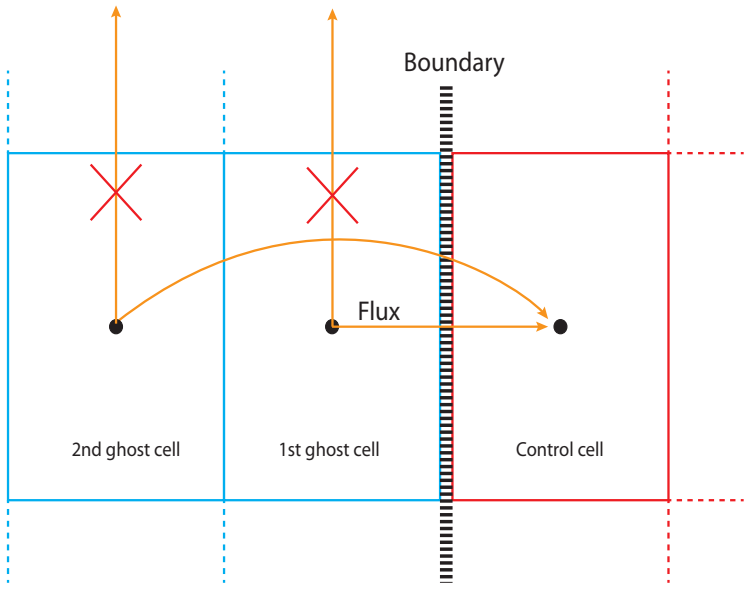


FIGURE 4.4: Set-up of the ghost cells

The ablative boundary condition is based on the mass, momentum and energy balances as discussed in the previous section. To computes the values in the ghost cells, several quantities have to be imposed on the wall: the temperature T_w and the pyrolysis gas blowing rate per species (see CEA2pt). Once those values are known, it is possible to compute density, pressure and velocity at the wall as follows [19]. Conservation of mass assuming that there is no loss is given by:

$$P_{tot} = P_{cc} + \rho_{cc} v_{cc}^2 = P_w + \rho_w v_w^2. \tag{4.2}$$

In this equation, subscripts w and cc respectively refer to the wall and the control cell. The problem requires two more equations to be solved. The first one is given by the expression of the mass flow rate:

$$\dot{m}_w = \rho_w v_w. \tag{4.3}$$

The second consists in the ideal gas equation:

$$\frac{P_w}{\rho_w} = R \cdot T_w. \tag{4.4}$$

This set of equations is solved via a second order equation and gives:

$$\begin{cases} \rho_w = \frac{P_{tot} + \sqrt{P_{tot}^2 - 4RT_w \dot{m}_w^2}}{2RT_w} \\ v_w = \frac{2RT_w \dot{m}_w}{P_{tot} + \sqrt{P_{tot}^2 - 4RT_w \dot{m}_w^2}} \\ P_w = \frac{P_{tot} + \sqrt{P_{tot}^2 - 4RT_w \dot{m}_w^2}}{2}. \end{cases} \tag{4.5}$$

To make this computation easier to understand, let's take the example of a finite-volume problem of first order (the second order case can be obtained in a similar way). As said before, only one ghost cell is required here. This method is illustrated in the figure 4.5. Let us denote F one particular conservative variable and "0", "1" and "w" the respective subscripts for the ghost cell, the control cell and the wall. In this case we have the following fluxes balance:

$$F_w \cdot (\Delta x_0 + \Delta x_1) = F_0 \Delta x_0 + F_1 \Delta x_1. \tag{4.6}$$

When considering that both the control and ghost cells have the same width (i.e. $\Delta x_0 = \Delta x_1$), F is computed as follows:

$$F_0 = 2F_w - F_1. \tag{4.7}$$

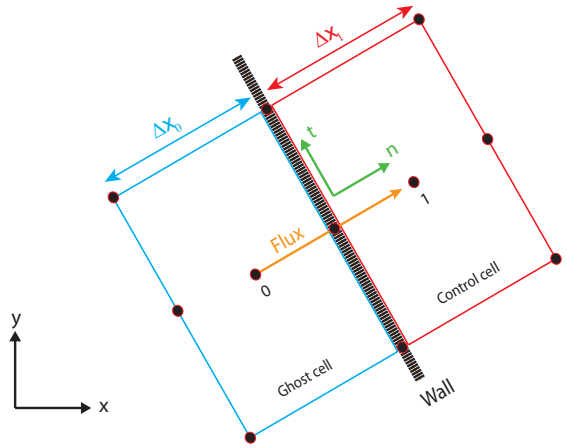


FIGURE 4.5: Arrangement of control and ghost cell

4.2.2 Implemented ablative BC in `Eilmer3`

The studied boundary condition was implemented during this work in `Eilmer3` thanks to a `lua` file called `udf-ablation.lua`. These are the main parameters used in this file:

$\dot{m}_{w,f}$: *Final mass flow* $(Kg \cdot m^{-2} \cdot s^{-1})$

$\dot{m}_{w,i}$: *Initial mass flow* $(Kg \cdot m^{-2} \cdot s^{-1})$

T_{wall}: *Temperature at the wall* (K)

u_{inf} : *Velocity at infinity* $(m \cdot s^{-1})$

Rn : *Blunt body radius* (m)

d_{inc} : *Factor for the increment of* $\dot{m}(m)$

Rg : *gas constant* $(J \cdot Kg^{-1} \cdot s^{-1})$

nsp : *Number of species in the mixture*

ntm : *Number of thermal degrees of freedom*

4.2.2.1 Mass fractions of the mixture injected

The first values to input in `udf-ablation.lua` are the mass fractions of the 20 species injected at the wall [4]. They are computed via a `python` module called `cea2PT.py`. This `Python` script is based on the NASA Computer program CEA (Chemical Equilibrium with Applications) that computes chemical equilibrium compositions and properties of complex mixtures. Figure 4.6 gives an overview of how `cea2PT.py` works:

FIGURE 4.6: `cea2PT.py scheme`

Once computed with `cea2PT.py`, the mass fractions are set in the ablative boundary condition as follows via a table called `massf_wall`

4.2.2.2 Flow with blowing gas rate

The following function called `flow_with_massflux` computes all the values needed to set up the mass flow with injection of pyrolysis gases. This function creates two different tables: the first one called `Q` contains the values of mass fractions and temperature for each species and the second one called `flow` includes values at the wall required for the balance of fluxes. The values of density, pressure and velocity are computed with equation 4.5. They use values returned by a function called `sample_flow`, already implemented in `eilmer3` code. It returns values of flow variables of any cell that can be localized by the parameters `block_id`, `args.i`,`args.j` and `args.k`. For more information about cell localization in `eilmer3`, authors refer to [20]. Note that:

- `p_eta` is the total pressure defined as: $p_{eta} = p + \rho u_n^2$
- `u_n_nc` is the velocity computed in the local reference of the control cell.

In this function the blowing mass flow rate is increased progressively in order to avoid any numerical instability. At every iteration step, a certain constant amount of mass flow rate is added linearly to the mass flow rate until the final mass flow rate, $\dot{m}_{w,f}$, defined by the modeling, is reached. The parameter d_{inc} is used to define the characteristic length until the ramp reaches the final mass flow rate.

Denoting $t_{inc} = \frac{d_{inc}}{u_{inf}}$, the linear ramp is described by following equation:

$$\dot{m} = \dot{m}_{w,i} + (\dot{m}_{w,f} - \dot{m}_{w,i}).\frac{t}{t_{inc}} \tag{4.8}$$

4.2.2.3 Values at the wall

The function called `interface` returns the values of velocity, temperature and mass fraction of each species at the wall via a table calle `face`.

4.2.2.4 Values in the ghost cell

The function `ghost_cell` is similar to the former one. It now works on the ghost cells.

4.2.2.5 Injected mass flux

The last function called `flux` computes values of mass flux, momentum, energy and mass fractions.

4.2.3 Verification and validation

Before using this ablating boundary condition on the Hayabusa reentry capsule in different trajectory points, it has to be validated. As a gas mixture in injected a the wall of the probe, the evolution of the velocity, density, and heat flux should be coherent with the blowing mass flow rate. The reference case is a viscous simulation without injected mass flux. If this boundary condition is efficient, the density and the conductive heat flux should decrease as the mass flow increases and the velocity has to be directed outgoing from the ablative layer. These verification were done in the present work with a blunt wedge geometry using the 20 species Park model.

4.2.3.1 Set up of the test case

The blunt wedge case is part of the examples of `Eilmer3` and was lead by Chen and Milos [21]. It is an axisymmetric blunt body with a 19.05 mm nose radius (figure 4.7). In table 4.1, the characteristics of the mesh are summarized. The geometry has been divided into two blocks to use the parallelism of the code and thus optimize the computation time.

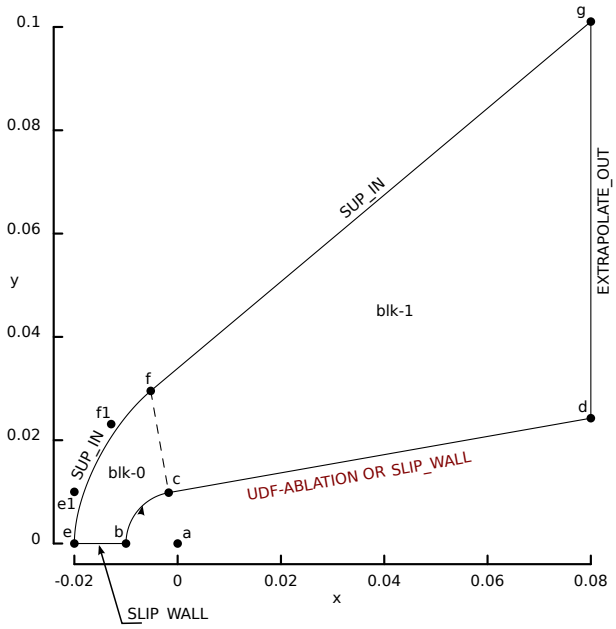


FIGURE 4.7: Blunt wedge geometry

TABLE 4.1: Characteristics of the mesh

mesh type	structured quad
number of blocks	2
mesh $x * y$ (block 0)	$40 * 40$
mesh $x * y$ (block 1)	$70 * 40$

4.2.3.2 Run simulations

For a finite-volume method, the values in each cell must be initialized before the first iteration. Convergence of the simulation may depend, among others, on this choice. Far from the wall, one can assume that the influence of blowing with a small mass flow rate (blowing boundary condition) will be rather low. Hence, the converged viscous simulation without ablation is a suitable starting point for the simulation with ablation. Besides, having computed the simulation without ablation can prove to be useful when comparing the results with the case with ablation in order to identify the effectiveness of the ablator. In this validation test case, chemical diffusion has been neglected to reduce the computation time.

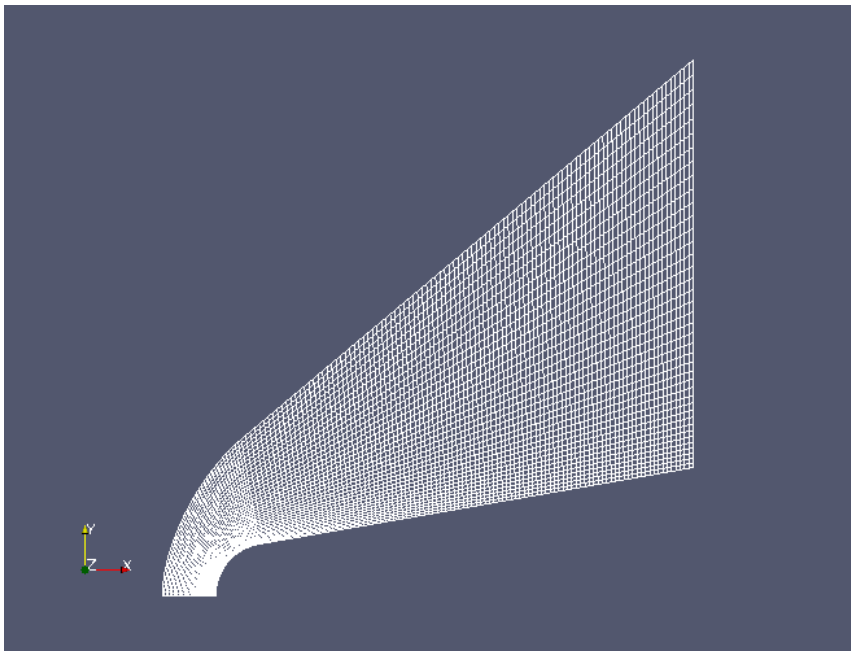


FIGURE 4.8: Blunt wedge mesh

In the present work, the freestream conditions for the reference case without ablation were as follows [21]:

TABLE 4.2: Freestream conditions for reference case (no ablation)

Velocity $[m/s]$	5354
Density $[kg/m^3]$	0.003
Temperature $[K]$	1428
Y_{N2}	0.6169
Y_{NO}	0.0046
Y_N	0.1212
Y_O	0.2573

For the ablative simulations, only carbon was injected at the wall (table 4.4). Two different simulations were run with the following properties:

TABLE 4.3: Ablative Wall properties

Case	$T_w[K]$	$\dot{m}_w[kg/m^2s]$
1	3000	$2.5 \cdot 10^{-1}$
2	3000	$5 \cdot 10^{-1}$

TABLE 4.4: Chemical composition of pyrolysis gases at the wall

Y_{N2}	0.6169
Y_{NO}	0.0046
Y_N	0.1212
Y_O	0.2573
Y_C	0.2573

4.2.3.3 Results analysis

In this section, the evolution of density, conductive heat flux,and mass fractions will be studied. The evolution of the density ρ_w is consistent as the density decreases as the mass flow increases (Figure 4.9), which is coherent with equation 4.9.

$$\rho_w(\dot{m}_w = 0) = \frac{P_{tot}}{RT_w} \geq \frac{P_{tot} + \sqrt{P_{tot}^2 - 4RT_w\dot{m}_w^2}}{2RT_w} \tag{4.9}$$

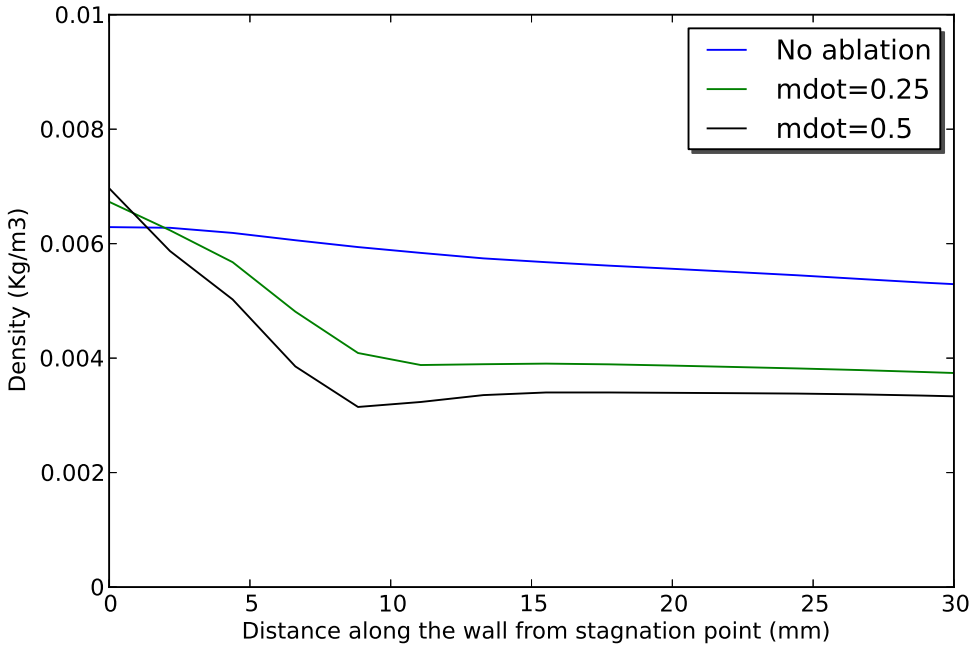


FIGURE 4.9: Density along the blunt wedge's wall with different mass flows

As for conductive heating, the two cases are consistent as conductive heat flux drops away from the stagnation point. It is due to the shock wave induced by the hypersonic flow. Indeed, the shock wave is vertical and so the most intense in front of the stagnation point (Figure 4.10).

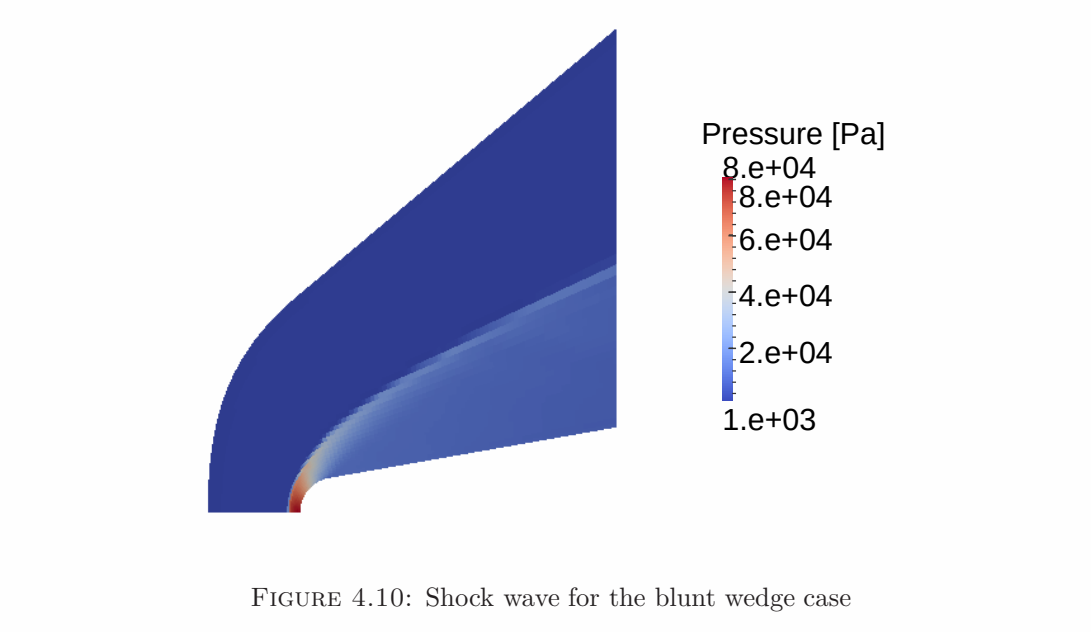


FIGURE 4.10: Shock wave for the blunt wedge case

Thus, when the distance from this point increases the wave becomes skewer, weaker and distant from the probe's wall. Besides the heat flux decreases as the mass flow increases, which is what we expected as it is the main goal of ablative material (Figure 4.11). Moreover, this reduction in conductive heat flux becomes smaller during the re-entry. Indeed, the TPS made of ablative species is getting thinner during re-entry so that the reduction of heat flux to the surface is smaller.

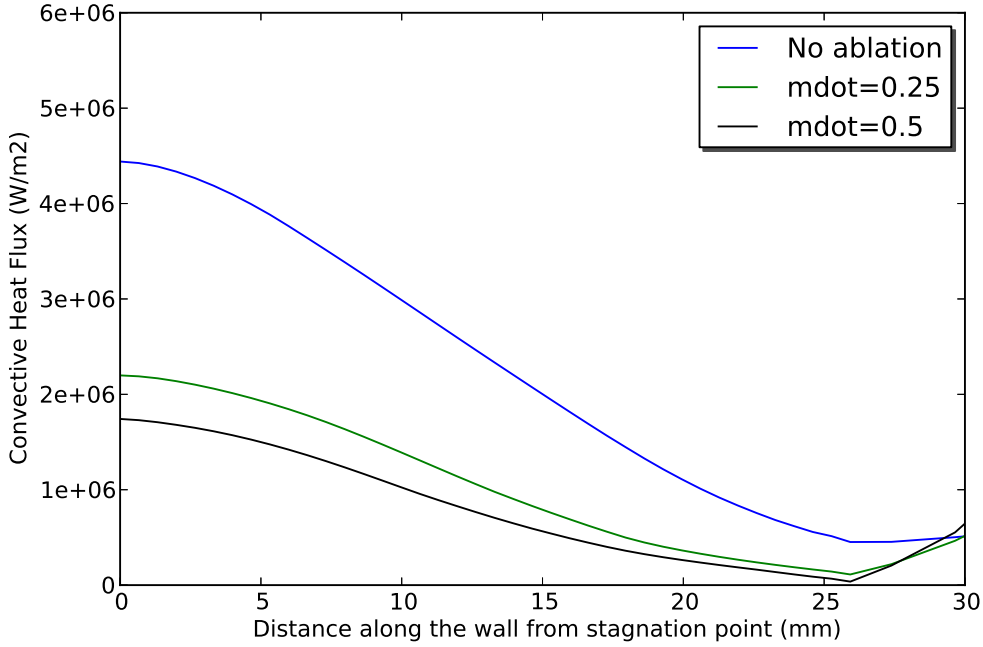


FIGURE 4.11: Conductive heat flux along the blunt wedge's wall with different mass flows

This reduction in heat flux is due to the injection of ablative species at the wall. As said before, only carbon was injected at the wall during cases 1 and 2 (with different mass flows). The following figure shows the evolution of carbon mass fraction Y_C along the probe's wall. For the viscous case with a fixed temperature boundary condition, there is no carbon at the wall as expected. For cases 1 and 2 the evolution of carbon mass fraction is relevant as it is higher when the mass flow rate is bigger.

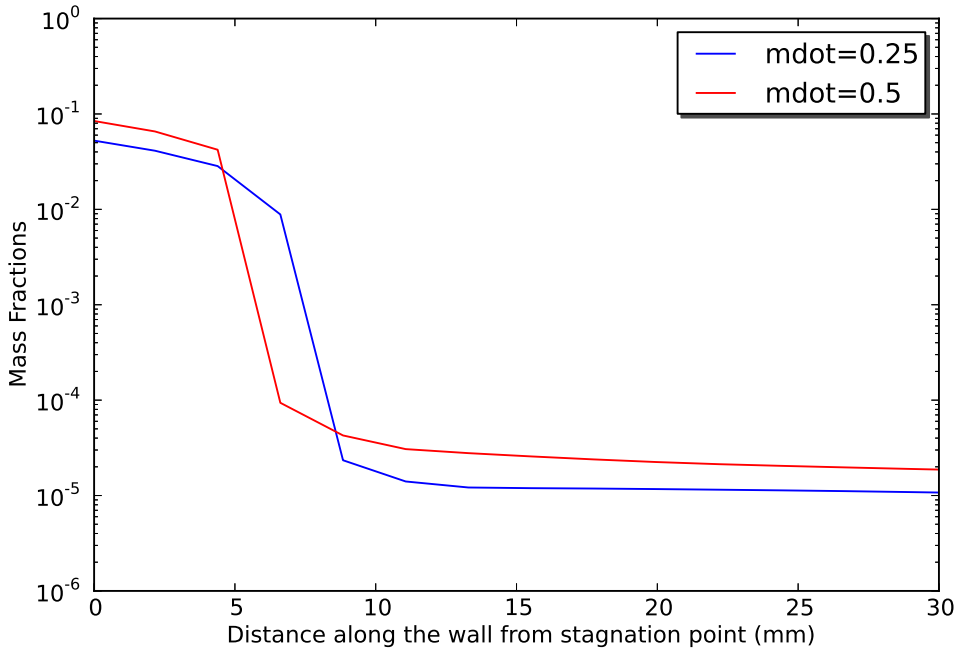
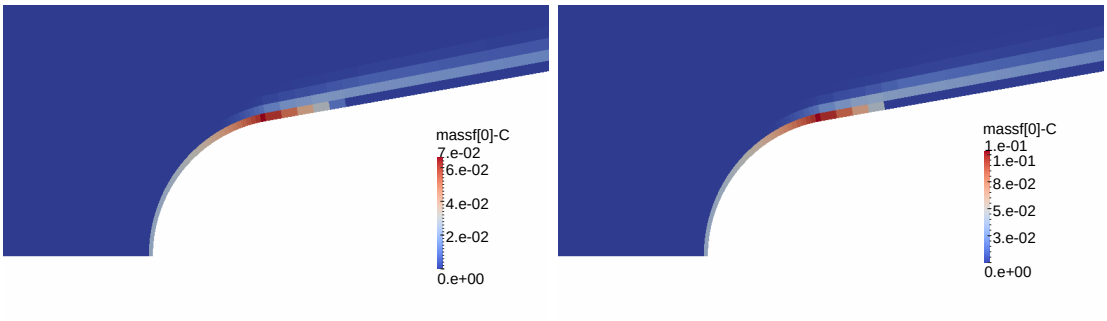


FIGURE 4.12: Carbon mass fraction along the blunt wedge's wall with different mass flows



(a) $\dot{m} = 0.25$ (b) $\dot{m} = 0.50$

FIGURE 4.13: Carbon mass fraction at the blunt wedge's wall for different mass flow rates

4.3 Hayabusa reentry coupled simulations

This section provides the results obtained during *Hayabusa*'s coupled simulations. Thus, it widely refers to the former sections and it is based on the coupled approach discussed earlier. During this work, many cases were studied as the vehicle's reentry was simulated for two different trajectory points and three different scales (full, 1/5 and 1/10). Note that the low-scale models were designed and studied for the next step of this ESA project which consists in an experimental campaign: ablation-radition coupling ground testing in expansion tubes.
In this report, only one trajectory point for the full scale model will be widely studied so that it's consistent with the results of the first part of this report. Thus, refer to tab 2.3 for the freestream conditions and the entire chapter 2 for the uncoupled simulations.

4.3.1 Ablation/flowfield coupling

As said in the first chapter, the diffusion model needs the finite-rate chemistry boundary condition to give consistent results. Thus, ablation simulation will be initialized with the viscous simulation of *Hayabusa*'s reentry under H1 conditions. As a grid resolution study has been lead, the 60×60 mesh is kept for the following simulations.

During this work, only the first iteration of the loose coupling procedure described in table 4.3 was lead. It can be schemed as follows (figure 4.15):

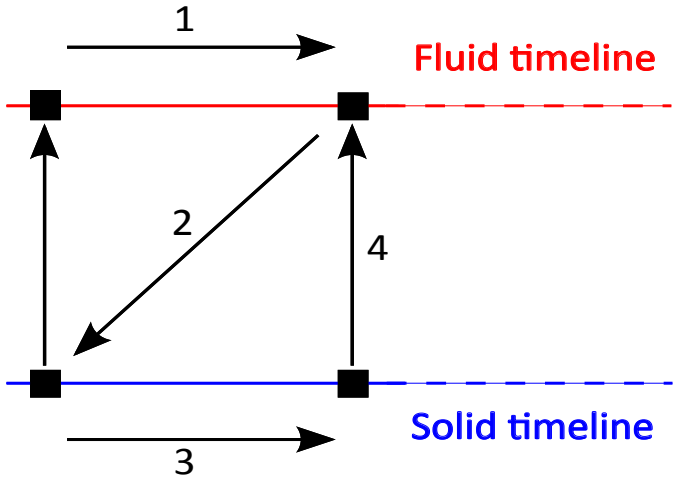


FIGURE 4.14: First iteration of the loose partitioned algorithm

For a better understanding, a detailed list of the steps followed during this first loop of the ablation/flowfield coupling is given in figure 4.16. Note that each step is located on the coupling diagram (figure 4.15) and will be widely discussed in the coming sections.

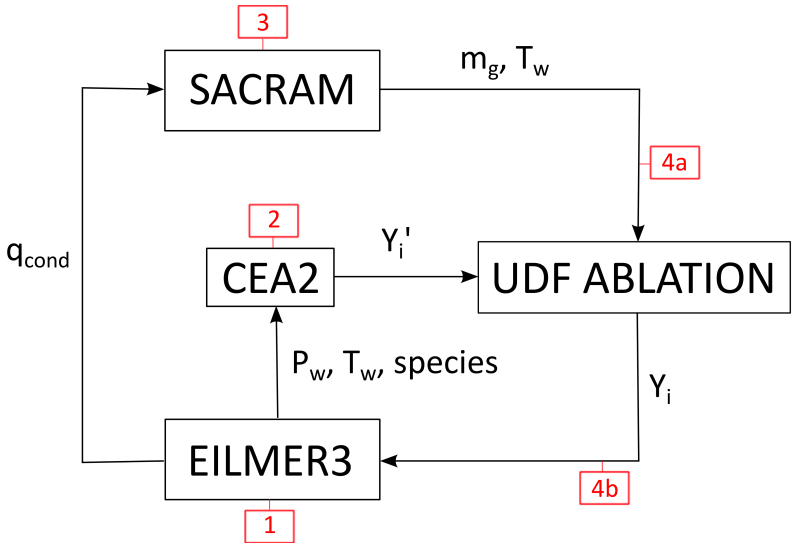


FIGURE 4.15: First iteration of the coupling procedure diagram

- TP_n = Peak radiative heating

  1. Calculate the steady-state solution of the fluid domain with `Eilmer3` H1 freestream conditions.

  2. Use `cea2PT` as a radiative equilibrium boundary condition with no finite rate chemistry ablation reactions to find gas composition: Pass the values of P_w and T_w calculated by `Eilmer3` to `cea2PT` and the chemical species list to calculate the gas composition (Y_i').

  3. Integrate in time the solid response domain from time t_0 to t_{H1} using time-accurate conditions for $\overrightarrow{q_{tot}}$ and p_w from flight data.

  4. (a) Pass the values of Y_i' ,T_w and $\dot{m}_g$ to `udf-ablation` to compute values of Y_i.

     (b) Pass the values Y_i and T_w to `Eilmer3`

FIGURE 4.16: First iteration of ablation/flowfield partitioned algorithm

4.3.1.1 Step 1: Viscous simulation with `Eilmer3`

The first step of the loop consists in a viscous simulation of *Hayabusa* full scale model reentry under H1 conditions. This was done at the beginning of this report, refer the "Verification and validation" part of chapter 1.

4.3.1.2 Step 2: Radiative equilibrium boundary condition

This step consists in determining the gas composition at the probe's wall. Thus, values of P_w and T_w are needed (See figure 4.6) so as to determine the mass fractions of each species Y_i'. The viscous case gave the following values:

TABLE 4.5: cea2PT input values

$P_w[Pa]$	$5.48 \cdot 10^4$
$T_w[K]$	3140

This is done via the `cea2PT.py` python file which also requires the species list which corresponds here to the 20 species of Park's model and their mole fractions. The command line is as follows:

```
-------------------------------------------------------------------------------
python cea2pT.py --species="C  O  N  H  CO  C2  N2  CN  NO  O2  H2  C3  C2H \\
C+  O+  H+ N+  NO+ N2+ e-" --input=moles --fractions="0.206 0.115 0.0 0.679 \\
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0" --output=moles --p=54580.0 --T=3140.0
-------------------------------------------------------------------------------
```

We end up with this gas mixture composition:

TABLE 4.6: Gas composition at the wall

C	$4.79 \cdot 10^{-4}$
O	$4.07 \cdot 10^{-9}$
H	$1.69 \cdot 10^{-2}$
CO	$6.44 \cdot 10^{-1}$
$C2$	$2.38 \cdot 10^{-3}$
$H2$	$1.17 \cdot 10^{-1}$
$C3$	$1.44 \cdot 10^{-1}$
$C2H$	$7.48 \cdot 10^{-2}$

Note that we considered a threshold of $1e^{-10}$ for mass fraction, every species having a lower value is being considered as non-existent. Thus, table 4.6 represents the actual gas mixture

which will be injected at the wall with a given mass flow rate $\dot{m}_g$. The latter is an output of `Sacram`.

4.3.1.3 Step 3: Material solid response

The third step of the loop corresponds to the work done in the "Validation and verification" part of chapter 3. The TPS undergoes the conductive heat flux which corresponds to H1 conditions (following flight data from JAXA). The material response simulation gives the following results:

TABLE 4.7: Sacram output values

$T_w[K]$	3000
$\dot{m}_g[kg/m^2s]$	$3.5 \cdot 10^{-2}$

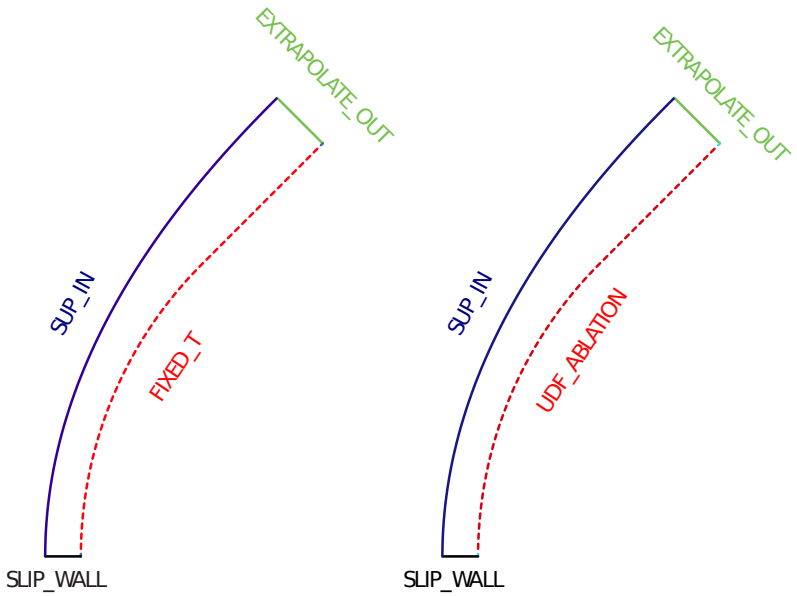
4.3.1.4 Step 4: `Eilmer3` simulation with ablating boundary condition

This last step is divided into two different sub-steps as both `udf_ablation.lua` and `Eilmer3` are given new inputs but it is done in the same framework. Indeed, values obtained with `cea2PT` and `Sacram` are passed to the boundary condition `python` file which is used during the fluid simulation. Note that simulation using this kind of ablating boundary condition has been validated and verified in at the beginning of this chapter. The global parameters used during this calculation were set as follows:

$$
\begin{aligned}
&\dot{m}_{w,f} = 3.5 \cdot 10^{-2} && : Final mass flow (Kg \cdot m^{-2} \cdot s^{-1}) \\
&\dot{m}_{w,i} = 3.5 \cdot 10^{-4} && : Initial mass flow (Kg \cdot m^{-2} \cdot s^{-1}) \\
&T_{wall} = 3000 && : Temperature at the wall (K) \\
&u_{inf} = 10.52 \cdot 10^{3} && : Velocity at infinity (m \cdot s^{-1}) \\
&d_{inc} = 2 && : Factor for the increment of \dot{m}(m) \\
&Rg = 8.314 && : gas constant (J \cdot Kg^{-1} \cdot s^{-1}) \\
&nsp = 20 && : Number of species in the mixture \\
&ntm = 2 && : Number of thermal degrees of freedom
\end{aligned}
$$

Even if only one iteration of the loose coupling algorithm was performed here, the results might be relevant enough to be compared with those obtained in the viscous simulation. Indeed,

a comparison of both sets of results will help identify the effectiveness of the ablator. The following figure (figure 4.17) summarizes the boundary conditions used in both cases. Note that this simulations were run with a 63×63 mesh subdivided into 49 sub-blocks to reduce computational effort. Once converged, the calculation will be studied in terms of temperature, density, normal velocity, heat fluxes and gas composition.

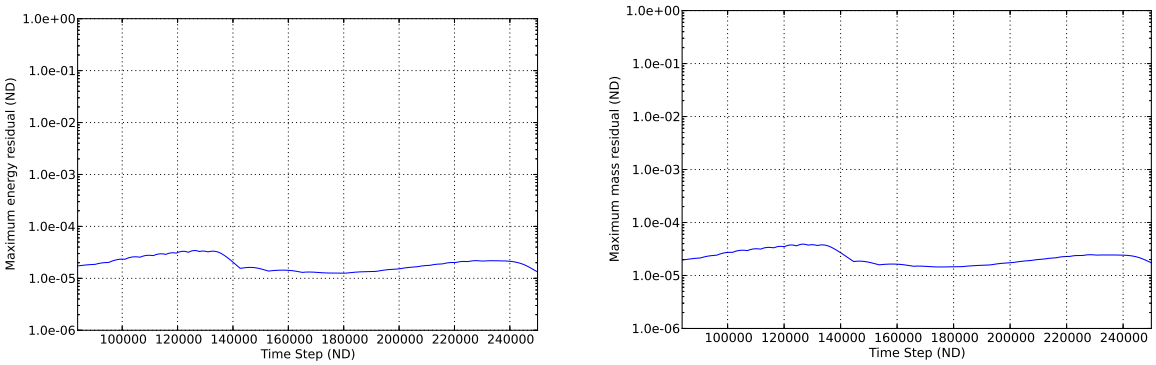


(a) Case without ablation (b) Case with ablation

FIGURE 4.17: Boundary conditions used for the simulation of *Hayabusa* under H1 conditions

4.3.1.5 Results analysis

This 63×63 shows good convergence results, again the maximum energy and mass residuals keep a very low value during the whole calculation: $1 \cdot 10^{-5}$. Besides the Reynold's number keeps almost constant to a very low value (around 2.0) so that the flow is laminar, as expected (picture 4.19).



(a) Maximum total energy residuals

(b) Maximum total mass residuals

FIGURE 4.18: Maximum energy and mass residuals for ablation simulation on Hayabusa full scale model under H1 conditions

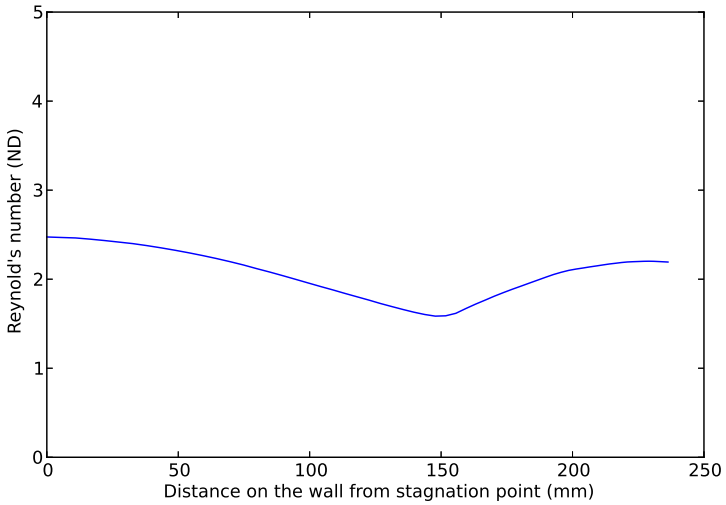


FIGURE 4.19: Reynold's number along the wall from stagnation point

As ablative species were injected at the wall, the gas mixture composition has been strongly modified for both air species and ablative species. The following plots show the evolution of air and ablative species mass fractions along the stagnation line. They have to be compared with the equivalents plots for the viscous simulation (Figure 2.10).

With this model, the variation of mass fractions of air species are not more very significant. As for the ablative species, the mass fraction of H is significant. This is the main ablative species, as its mass fraction (about 0.3) at the wall is about ten times as much as the following highest mass fraction Y_C, Y_{CN}, Y_{CO} (between 0.04 and 0.02). Other species are much less significant at the wall. At a certain distance from the wall, mass fractions become more balanced but not significant compared against air species. It can also be noticed that ablative species do not go

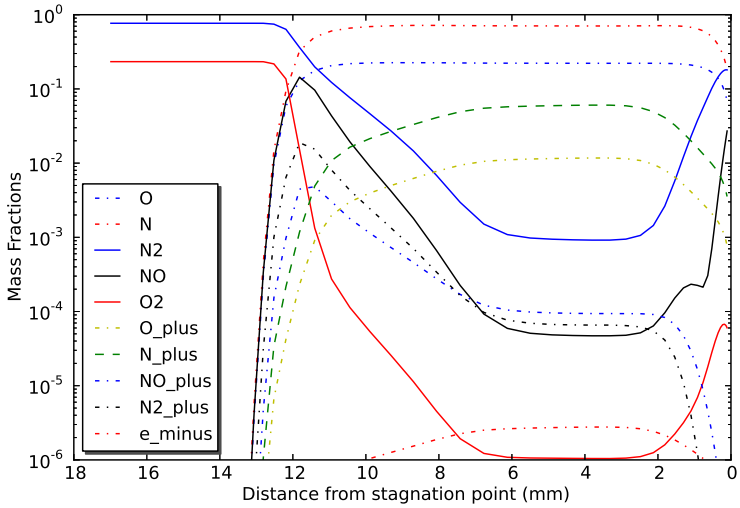


FIGURE 4.20: Air species mass fractions using Park's model (ablative flow, H1)

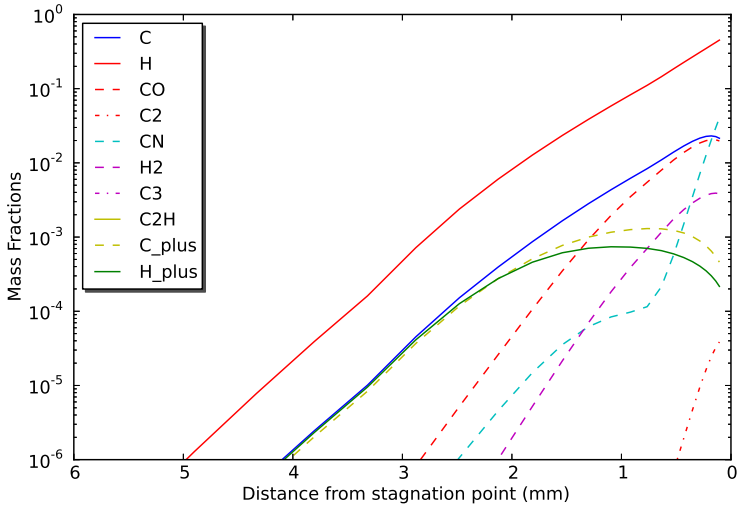
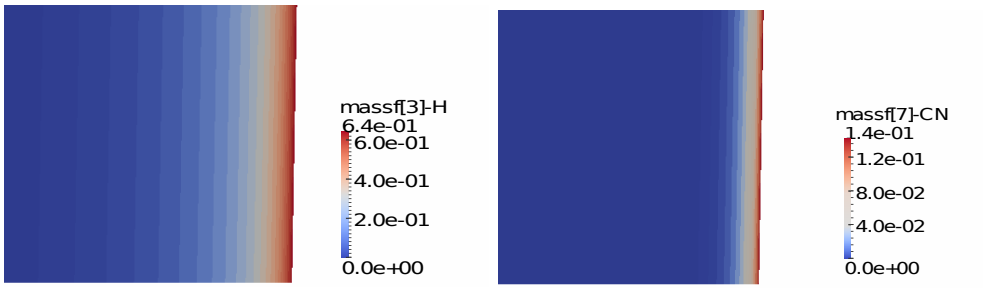


FIGURE 4.21: Ablative species mass fractions using Park's model (ablative flow, H1)

further than $5mm$ away from the wall, which is small compared with the situation of the shock (about $11mm$). It can be verified with the following Paraview screenshots showing the H and C mass fraction field:



(a) Y_{H} (b) Y_{CN}

FIGURE 4.22: H and CN mass fractions at the probe's wall

On figures 4.23, temperature profiles are plotted along the stagnation streamline for Park's model, with and without ablation. At approximately a distance of $11mm$ from the stagnation point, one can observe a sudden raise of temperature for the case without ablation. This raise reaches a peak and then relaxes slowly. This phenomena is typical for a fluid crossing a shock wave. Between the cases with and without ablation, one can notice a shift in the peak of temperature. A shift to the left means that the shock wave occurs further away from the probe's wall, which, as stated in the previous subsection, results in a smaller convective heat flux at the wall. This shift in the temperature peak is really significant as the shock-off distance is now about $12.5mm$. Note that the peak T_{tr} is higher in the case with ablation.

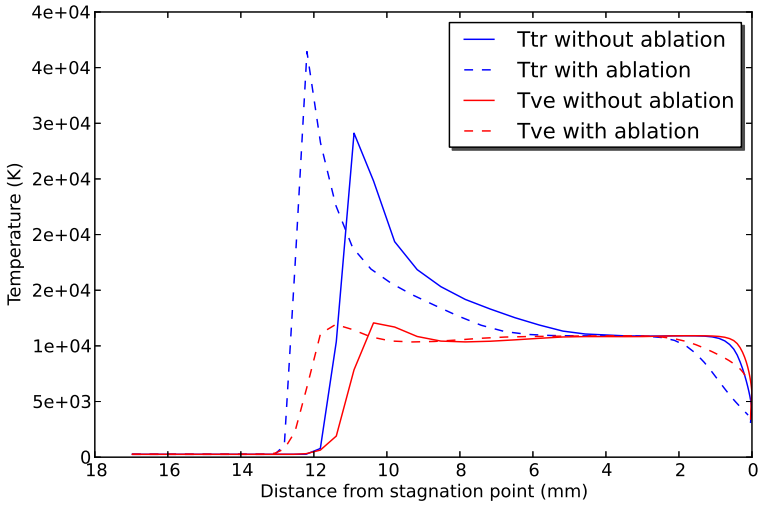


FIGURE 4.23: Temperature profiles along the stagnation streamline (ablative flow, H1)

Figure 4.24 shows the conductive heat flux at the probe's wall (starting from the stagnation point) with and without ablation. In both cases, convective heat flux drop away from the stagnation point. The conductive heat flux is smaller in the case of an ablative wall that it is for a non-ablative wall. Indeed, for a reduction of the maximum conductive heat flux of 31% can be reached thanks to the ablator. Thus, in this first case (ablative wall) less heat is transferred to the probe, which is exactly the expectation of such a TPS. This observation is a result of the shift of the shock wave due to the injected ablative gas at the wall. In the next subsection will this phenomena be explained.

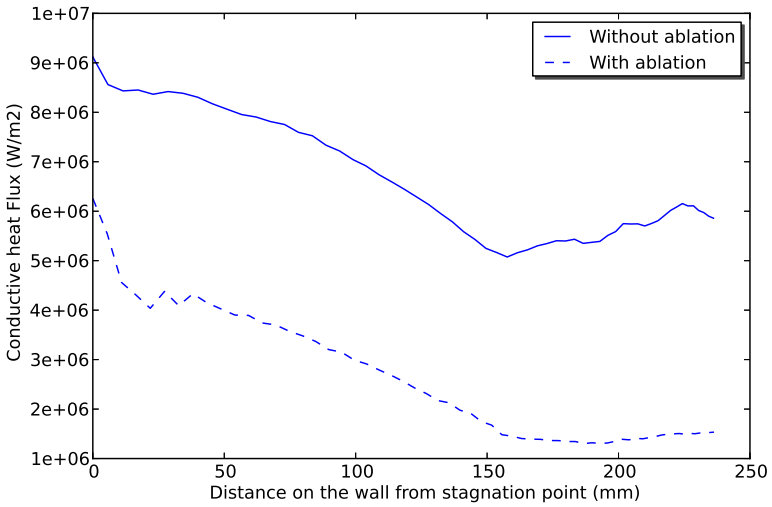


FIGURE 4.24: Conductive heat flux along the stagnation streamline (ablative flow, H1)

4.3.2 Radiation/flowfield coupling

Again the viscous simulation run with 60×60 under H1 freestream conditions is used as a starting point. Calculations were run with two different electronic level populations models: an equilibrium one (Boltzmann) and a non-equilibrium one (QSS: Quasi Steady State). Besides three different radiation transport models were used: Tangent slab, Discrete transfer and Monte Carlo (See chapter 1 for further explanations). Here is a table summarizing the studied cases:

TABLE 4.8: Radiation/flowfield simulations models

Case	Electronic level population	Radiation transport
1	Boltzamnn	Tangent slab
2	QSS	Tangent slab
3	QSS	Discrete transfer with 32 rays
4	QSS	Discrete transfer with 64 rays
5	QSS	Monte-Carlo

The spectral range considered is $50 \leq \lambda \leq 1200nm$ and the radiators considered are:

$$C,\ O,\ N,\ H,\ CO,\ C_2,\ N_2,\ CN,\ NO,\ O_2,\ H_2,\ O^+,\ H^+,\ N^+,\ NO^+,\ N_2^+,\ e^-$$

4.3.2.1 Results analysis

As shown in figure 4.25, the radiative divergence $\nabla.q_{rad}$ along the stagnation streamline is highly perturbed between 11 and 8 mm to the stagnation point. Indeed, it drops to $-2 \cdot 10^9\ W/cm^3$, which means re-absorption is strong through-out the whole shock layer. Besides, the values of radiative flux incident at the stagnation point given in table 4.9 show the radiation-coupling is significant for *Hayabusa* under H1 conditions, especially in the shock layer area, even if the Goulard numbers are less than 0.01. Besides, the Boltzmann solution over-predicts the radiative divergence in the shock layer while the tangent slab solution slightly under-predicts it because of the strong curvature of the shock layer. Note that *case 5* using the Monte-Carlo model was not used for the following comparison as it did not proved to be stable enough during this work (See figure 4.26).

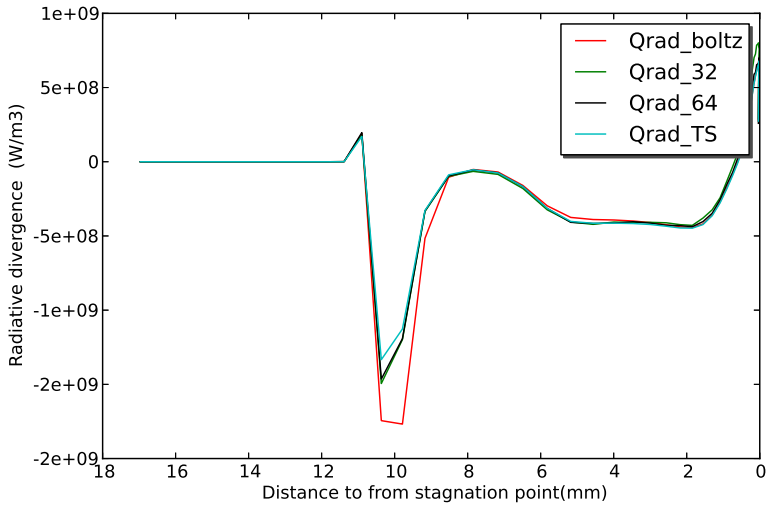
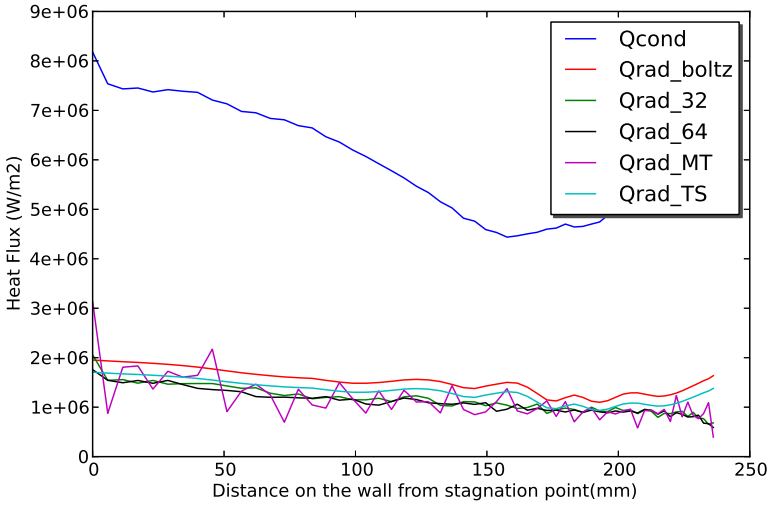


FIGURE 4.25: Radiative divergence along stagnation streamline

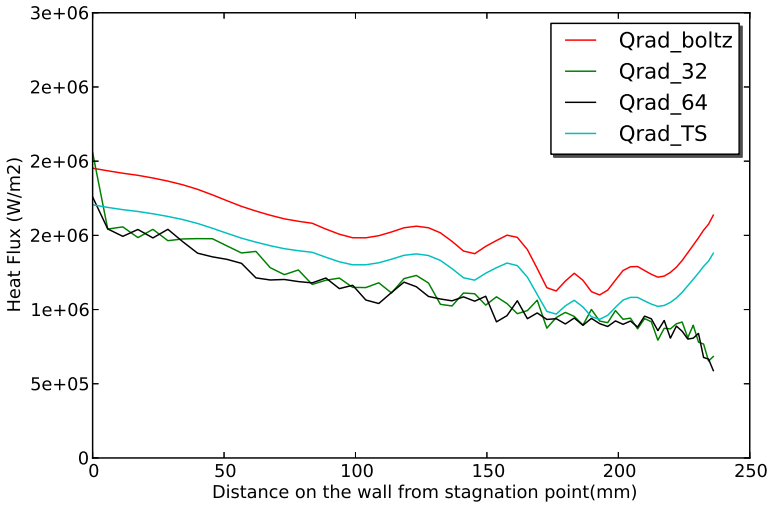
TABLE 4.9: Radiative heat fluxes at stagnation point and Goulard number

Case	$q_{rad}[MW/m^2]$	q_{rad}/q_{cond}	Γ
1	1.95	0.24	0.0113
2	1.71	0.21	0.0095
3	2.06	0.25	0.0047
4	1.76	0.22	0.0041

Thus, the radiative heat flux being approximately 20 percents of the overall heat flux (Figure 4.26), the use of radiative coupled simulations was required and relevant. The following plots show the evolution of both T_{tr} and T_{ve} along the stagnation line with and without radiation. When adding the effect of radiation, the shock detachment slightly decreases while the peak in T_{tr} increases and the peak in T_{ve} becomes a bit lower.



(a) Conductive and radiative heat fluxes



(b) Radiative heat flux

FIGURE 4.26: Heat fluxes along the probe's wall for radiation/flowfield coupled simulations of *Hayabusa* under H1 conditions

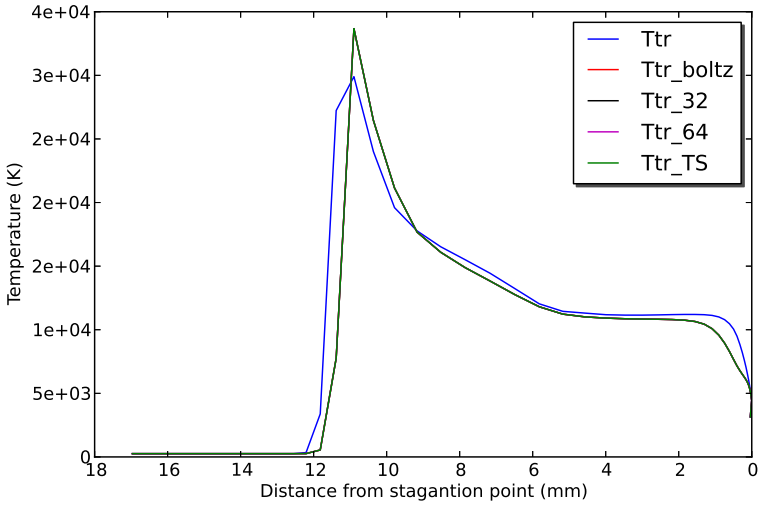


FIGURE 4.27: Translational-Rotational temperature along stagnation streamline under H1 conditions

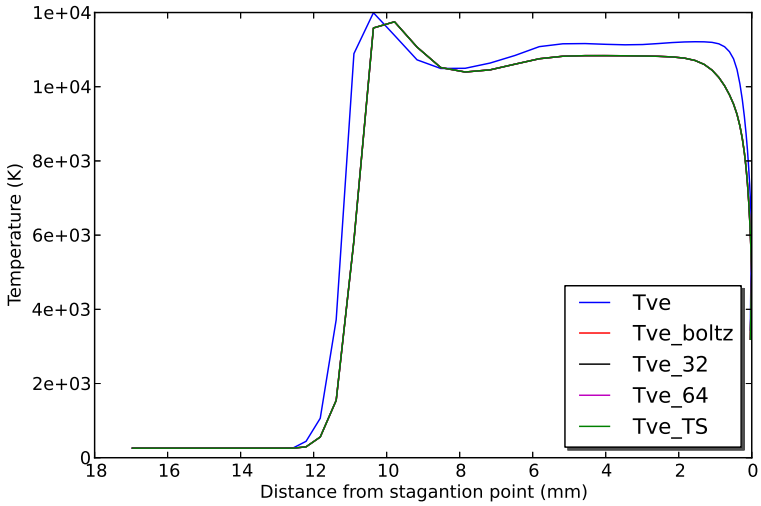


FIGURE 4.28: Vibrational-electron temperature along stagnation streamline under H1 conditions

Chapter 5

Conclusion

As part of the global European project ARC (Ablation-Radiation Coupling), the lab IAG (Interdisciplinary Aerodynamics Group) at EPFL is in charge of the numerical coupled approach of `Hayabusa`'s reentry. The severe thermal conditions that it undergoes during its hypersonic flight induces an extreme heat load composed of an important radiative part. To cope with the overall heating, a carbon phenolic TPS is used. The ablative species released during reentry react with the radiators present in the boundary layer so that the aerothermodynamic behavior of the probe is quite difficult to simulate. Therefore a coupled approach is lead between the fluid solver and the material response code thanks to a partitioned algorithm.
All the numerical tools used and created during this work were implemented in the framework of both `Eilmer3` and `Sacram` so that the future work on ablation-radiation coupling is ready to be performed. All the results obtained for *Hayabusa*'s reentry were compared to flight data and empirical correlations and proved to be accurate and relevant. To be even more accurate, a full coupling algorithm needs to be implemented and a test campaign in plasma wind tunnels has to be lead. Both steps are already scheduled by the consortium leading ARC project.

Appendix A

Ablating boundary condition script

```
-- udf-ablation-ideal.lua
-- User-specified mass-flux at a wall boundary.
-- Part 1 of development of ablation BCs for eilmer3:
-- pyrolysis only, no ablation, ideal case
-- from sources including Martin & Boyd (2009).
-- EJF, 22-Mar-2013.

nsp = 20                        -- number of species
ntm = 2                         -- number of thermal modes
T_wall = 3000.0                 -- temperature at the wall, K
mdot_i = 0.00035                -- initial mass flow rate, kg/m^2-s
mdot_f = 0.035                  -- final mass flow rate, kg/m^2-s
mdot_increment = 2.0            -- a factor to increment the effect of mdot
u_inf = 10.770e3                -- inflow velocity, m/s

tf_increment = mdot_increment / u_inf

-- Specify mass fractions at wall
-- Order of species:
--  0: C     1: O     2: N      3: H      4: CO
--  5: C2    6: N2    7: CN     8: NO     9: O2
-- 10: H2   11: C3   12: C2H   13: C+    14: O+
-- 15: H+   16: N+   17: NO+   18: N2+   19: e-

massf_wall = {}
for i=0,nsp-1 do
      massf_wall[i] = 0.0
end
massf_wall[7] = 0.5       -- test values (not realistic)
massf_wall[12] = 0.5      -- test values (not realistic)
--print (massf_wall)

function flow_with_massflux(nc, args)
--   print('hello from flow w mf')
   mdot = mdot_i + (mdot_f-mdot_i) * args.t / tf_increment
   if args.t >  tf_increment then
      mdot = mdot_f
```

```
   end

   Q = create_empty_gas_table(nsp,ntm)
   Q.T[0] = T_wall
   Q.T[1] = T_wall
   Q.massf[0] = massf_wall[0]
   Q.massf[7] = massf_wall[7]
   Q.massf[12] = massf_wall[12]

   nc = sample_flow(block_id,args.i,args.j,args.k)
   u_n_nc = nc.u * args.csX + nc.v * args.csY
   p_eta = nc.p + nc.rho * u_n_nc^2.0
   R_gas = eval_R(Q)
   gamma = eval_gamma(Q)
   Cv = eval_Cv(Q)

   rho_wall=(p_eta+math.sqrt(p_eta^2.0-4.0*R_gas*T_wall*mdot^2.0))/(2.0*R_gas*T_wall)
   u_wall=(2.0*R_gas*T_wall*mdot)/(p_eta+math.sqrt(p_eta^2.0-4.0*R_gas*T_wall*mdot^2.0))
   p_wall=(p_eta+math.sqrt(p_eta^2.0-4.0*R_gas*T_wall*mdot^2.0))/ 2.0

   flow = {}
   for i=0,6 do
      flow[i] = 0.0
   end
   flow[0] = p_wall
   flow[1] = T_wall
   flow[2] = rho_wall
   flow[3] = u_wall
   flow[4] = R_gas
   flow[5] = gamma
   flow[6] = Cv
--   print (flow[0], flow[1], flow[2], flow[3], flow[4], flow[5], flow[6])
   return flow
end

function interface(args)
--   print('hello from interface')
   nc = sample_flow(block_id,args.i,args.j,args.k)
   flow = flow_with_massflux(nc, args)
--   print('flow = ', flow)
   face = {}
   face.u = flow[3] * args.csX
   face.v = flow[3] * args.csY
   face.w = 0.0
   face.T = {} -- temperatures, K (as a table)
   face.T[0] = T_wall
   face.T[1] = T_wall
   face.massf = {}
   for isp=0,nsp-1 do
      face.massf[isp] = massf_wall[isp]
   end
 --  print(face)
   return face
end
```

```
function ghost_cell(args)

   nc = sample_flow(block_id,args.i,args.j,args.k)
   flow = flow_with_massflux(nc, args)
--   print(flow[0])

   ghost = {}
   ghost.p = flow[0] -- pressure, Pa
   ghost.T = {} -- temperatures, K (as a table)
   ghost.T[0] = flow[1]
   ghost.T[1] = flow[1]
   ghost.u = flow[3]  -- x-velocity, m/s
   ghost.v = 0.0      -- y-velocity, m/s
   ghost.w = 0.0
 --  print('ghost = ', ghost)
   ghost.massf = {} -- mass fractions to be provided as a table
   for isp=0,nsp-1 do
      ghost.massf[isp] = massf_wall[isp]
   end
--   print('now ghost = ', ghost.massf[1])
   return ghost, ghost
end
--print(debug.getinfo(interface))

function flux(args)
  -- print('hello from flux')
   nc = sample_flow(block_id,args.i,args.j,args.k)
   flow = flow_with_massflux(nc, args)
   vt = 0.0           -- tangential velocity, m/s
--   print(vt)
   p = flow[0]
   rho = flow[2]
   Cv = flow[6]
   u = flow[3] * args.csX - vt * args.csY
   v = flow[3] * args.csY + vt * args.csX
   w = 0.0
 --  print(p, rho, Cv, u, v, w)
   -- Assemble flux vector
   F = {}
   F.mass = rho * (u * args.csX + v * args.csY) -- kg/s/m**2
   F.momentum_x = p * args.csX + u * F.mass
   F.momentum_y = p * args.csY + v * F.mass
   F.momentum_z = 0.0
   F.total_energy = F.mass * (Cv*T_wall + 0.5*(u*u+v*v) + p/rho)
   F.species = {}
   for isp=0,nsp-1 do
      F.species[isp] = F.mass * massf_wall[isp]
   end
   F.renergies = {}
   F.renergies[0] = F.mass * (Cv*T_wall)
   return F
end
```

Appendix B

Sacram scripts

Here are the Sacram scripts used during this work, they are based on Joshi's work and were modified and commented during this work.

B.1 gas/solid.py

```
from scipy.interpolate import interp1d
from numpy import *

## gas.py interpolates material properties to set parameters of the gas, solid and mixture

##################################### FUNCTIONS ################################################

def interpolate_gas_property(T): # using gas_props.data
    rho = interp1d(gas_prop_tab.T, gas_prop_tab.rho)(T)
    M = interp1d(gas_prop_tab.T, gas_prop_tab.M)(T)
    Cp = interp1d(gas_prop_tab.T, gas_prop_tab.Cp)(T)
    Cv = interp1d(gas_prop_tab.T, gas_prop_tab.Cv)(T)
    h = interp1d(gas_prop_tab.T, gas_prop_tab.h)(T)
    mu = interp1d(gas_prop_tab.T, gas_prop_tab.mu)(T)
    return [rho, M, Cp, Cv, h, mu]

def interpolate_solid_property(T,yV,yC): # using solid_props.data
    f1 = interp1d(solid_prop_tab.T, solid_prop_tab.Vir_Cp)
    f2 = interp1d(solid_prop_tab.T, solid_prop_tab.Vir_k)
    f3 = interp1d(solid_prop_tab.T, solid_prop_tab.Vir_h)
    f4 = interp1d(solid_prop_tab.T, solid_prop_tab.Char_Cp)
    f5 = interp1d(solid_prop_tab.T, solid_prop_tab.Char_k)
    f6 = interp1d(solid_prop_tab.T, solid_prop_tab.Char_h)
    Cp = yV * f1(T) + yC * f4(T)
    k = yV * f2(T) + yC * f5(T)
    h = yV * f3(T) + yC * f6(T)
    return [Cp, k, h]
```

```
def calculate_dkdT(solid):
    [Cp1, k1, h1] = interpolate_solid_property(solid.T + 0.01*solid.T, solid.yV,solid.yC)
    [Cp2, k2, h2] = interpolate_solid_property(solid.T - 0.01*solid.T, solid.yV,solid.yC)
    m = (k1 - k2) / (0.02*solid.T) # 0.02*solid.T = delta(solid.T)
    return m

############################ CLASS DEFINITIONS ################################################

##---------------------------------------GAS---------------------------------------##
class gas_prop_tab(object):
    def __init__(self):
        self.T = []
        self.rho = []
        self.M = []
        self.Cp = []
        self.Cv = []
        self.h = []
        self.mu = []

class gas_data(object):
    def __init__(self, T=0, V=0, R=8.31):
        self.T = T
        [self.rho, self.M, self.Cp, self.Cv, self.h, self.mu] = interpolate_gas_property(self.T)
        self.R = R
        self.p = self.rho * self.R / self.M * self.T
        self.e = self.h - self.p / self.rho
        self.V = V

##---------------------------------------SOLID---------------------------------------##
class solid_prop_tab(object):
    def __init__(self):
        self.T = []
        self.Vir_Cp = []
        self.Vir_k = []
        self.Vir_h = []
        self.Char_Cp = []
        self.Char_k = []
        self.Char_h = []

class solid_data(object):
    def __init__(self, T, gamma, phi, rhov, rhoc, rhoABC, rhos):
        self.T = T
        self.gamma = gamma # Resin volume fraction in the virgin composite
        self.phi = phi # Virgin porosity
        self.rhov = rhov # Overall virgin density (see TACOT_1.5.xls for formula)
        self.rhoc = rhoc # Overall char density (see TACOT as above)
        self.rhoABC = rhoABC # See Amar's thesis for definition, p.17
        self.rhos = (1-self.phi)*((self.gamma*(self.rhoABC[0]+self.rhoABC[1]))
                                  +((1-self.gamma)*(self.rhoABC[2]))) # rho_solid
        self.beta = (self.rhov - self.rhos)/(self.rhov-self.rhoc) # extent of reaction
        self.yV = (self.rhov/self.rhos)*(1-self.beta)
        self.yC = 1.0 - self.yV
        self.kappa = self.yV * 1.60e-11 + self.yC * 2.00e-11 # permeability, See TACOT_1.5.xls
        [self.Cp, self.k, self.h,] = interpolate_solid_property(self.T,self.yV,self.yC)
```

```
        self.e = self.h
        self.Cv = self.Cp

##--------------------------------------MIXTURE--------------------------------------##
class mix_data(object):
    def __init__(self, T, rhoABC, V, rhos, rhov, rhoc, gamma, phi):
        self.T = T
        self.solid = solid_data(self.T, gamma, phi, rhov, rhoc, rhoABC, rhos)
        self.gas = gas_data(self.T, V)
        self.epsilon = self.solid.phi+
            0.1*(1-(sum(rhoABC)-self.solid.rhoABC[2])/(self.solid.rhoABC[0]+self.solid.rhoABC[1]))
        self.rho = self.epsilon * self.gas.rho + self.solid.rhos
        self.yg = self.epsilon * self.gas.rho / self.rho
        self.e = self.yg * self.gas.e + (1.0 - self.yg) * self.solid.e
        self.h = self.yg * self.gas.h + (1.0 - self.yg) * self.solid.h
        self.Cp = self.yg * self.gas.Cp + (1.0 - self.yg) * self.solid.Cp
        self.Cv = self.yg * self.gas.Cv + (1.0 - self.yg) * self.solid.Cv
        self.k = self.solid.k
```

B.2 mixture-energy.py

```
from scipy.interpolate import interp1d
from numpy import *
from gas import *

## mixture_energy calculates the terms to be input in the jacobian function (for further
    details see Amar's thesis)
## which resolves the semi discrete mixture energy conservation equation
## -> outflow terms - inflow terms + rate of change of content = 0

## The values for "e" are commented out in this file as recession is not being taken into
     account in this case.

######################################## FUNCTIONS
    ################################################

# -- Conductive term

def calculate_mix_Q(gdata, grid, mix):
    Q = zeros(gdata.npoints)
    dx = grid.dx
    for j in range(gdata.npoints-1):
        Q[j]    += 1.0 / dx * 0.5 * (mix[j].k+mix[j+1].k) * (mix[j].T - mix[j+1].T)
        Q[j+1]  += -1.0 / dx * 0.5 * (mix[j].k+mix[j+1].k) * (mix[j].T - mix[j+1].T)
    return Q

def calculate_mix_Q_jacobian(gdata, grid, mix):
    dx = grid.dx
    dQdT = zeros((gdata.npoints,gdata.npoints))
    a = zeros(gdata.npoints-1)
    b = zeros(gdata.npoints)
    c = zeros(gdata.npoints-1)
    #e = zeros(gdata.npoints)
    for j in range(gdata.npoints-1):
        b[j]    += 0.5 / dx * (mix[j].k + mix[j+1].k) #+ 0.5 / dx * calculate_dkdT(mix[j
    ].solid) * (mix[j].T - mix[j+1].T)
        b[j+1]  += 0.5 / dx * (mix[j].k + mix[j+1].k) #- 0.5 / dx * calculate_dkdT(mix[j
    +1].solid) * (mix[j].T - mix[j+1].T)
        c[j]    += -0.5 / dx * (mix[j].k + mix[j+1].k) #+ 0.5 / dx * calculate_dkdT(mix[j
    ].solid) * (mix[j].T - mix[j+1].T)
        a[j]    += -0.5 / dx * (mix[j].k + mix[j+1].k) #- 0.5 / dx * calculate_dkdT(mix[j
    ].solid) * (mix[j].T - mix[j+1].T)
        #e[j]    += -DdxDsdot / dx * 0.5 * (mix[j].k+mix[j+1].k) * (mix[j].T - mix[j+1].T
    )
        #e[j+1]  += DdxDsdot * epsilon / dx * 0.5 * (mix[j].k+mix[j+1].k) * (mix[j].T -
    mix[j+1].T)

        ## 2nd part of above equations commented out due to negligiable influence on
    output & visible gain in computational efficiency
    dQdT += diag(b) + diag(a,-1) + diag(c,1)
    return dQdT
```

```
# -- Grid convective term

def calculate_mix_G(gdata, grid, mix):
    G = zeros(gdata.npoints)
    for j in range(gdata.npoints-1):
        Ux      = 0.5 * (grid.nodes[j].Ux + grid.nodes[j+1].Ux)
        G[j]    += -0.5 * (mix[j].rho*mix[j].h*Ux + mix[j+1].rho*mix[j+1].h*Ux)
        G[j+1]  += 0.5 * (mix[j].rho*mix[j].h*Ux + mix[j+1].rho*mix[j+1].h*Ux)
    return G

def calculate_mix_G_jacobian(gdata, grid, mix):
    dGdT = zeros((gdata.npoints,gdata.npoints))
    a = zeros(gdata.npoints-1)
    b = zeros(gdata.npoints)
    c = zeros(gdata.npoints-1)
    #e = zeros(gdata.npoints)
    for j in range(gdata.npoints-1):
        Ux      = 0.5 * (grid.nodes[j].Ux + grid.nodes[j+1].Ux)
        b[j]    += -0.5 * (mix[j].rho * mix[j].Cp * Ux)
        b[j+1]  += 0.5 * (mix[j+1].rho * mix[j+1].Cp * Ux)
        c[j]    += -0.5 * (mix[j+1].rho * mix[j+1].Cp * Ux)
        a[j]    += 0.5 * (mix[j+1].rho * mix[j+1].Cp * Ux)
        #e[j]    += -epsilon * 0.5 * (gas[j].rho*gas[j].h + gas[j+1].rho*gas[j+1].h) *
    0.5 * (grid.eta[j] + grid.eta[j+1])
        #e[j+1]  += epsilon * 0.5 * (gas[j].rho*gas[j].h + gas[j+1].rho*gas[j+1].h) * 0.5
     * (grid.eta[j] + grid.eta[j+1])
    dGdT += diag(a,-1) + diag(c,1) + diag(b)
    return dGdT


# -- Convective term (Gas Flux in Amar's thesis)

def calculate_gas_F(gdata, grid, mix):
    dx = grid.dx
    F = zeros(gdata.npoints)
    for j in range(gdata.npoints-1):
        V       = mix[j].gas.V
        F[j]    += mix[j].epsilon * V * 0.5 * (mix[j].gas.rho*mix[j].gas.h + mix[j+1].gas
    .rho*mix[j+1].gas.h)
        F[j+1]  += -mix[j].epsilon * V *0.5 * (mix[j].gas.rho*mix[j].gas.h + mix[j+1].gas
    .rho*mix[j+1].gas.h)
    return F

def calculate_gas_F_jacobian(gdata, grid, mix):
    dx = grid.dx
    dFdT = zeros((gdata.npoints,gdata.npoints))
    a = zeros(gdata.npoints-1)
    b = zeros(gdata.npoints)
    c = zeros(gdata.npoints-1)
    #e = zeros(gdata.npoints)
    for j in range(gdata.npoints-1):
        V       = mix[j].gas.V
        b[j]    += mix[j].epsilon * V * 0.5 * (mix[j].gas.rho * mix[j].gas.Cp)
        b[j+1]  += -mix[j+1].epsilon * V * 0.5 * (mix[j+1].gas.rho * mix[j+1].gas.Cp)
```

```
        c[j]    += mix[j+1].epsilon * V * 0.5 * (mix[j+1].gas.rho * mix[j+1].gas.Cp)
        a[j]    += -mix[j].epsilon * V * 0.5 * (mix[j].gas.rho * mix[j].gas.Cp)
        #e[j]    += 0.0
        #e[j+1]  += 0.0
    dFdT += diag(a,-1) + diag(c,1) + diag(b)
    return dFdT


# -- Total energy content term

def calculate_mix_E(gdata, grid, mix):
    E = zeros(gdata.npoints)
    dx = grid.dx
    for j in range(gdata.npoints-1):
        E[j]    += dx / 8.0 * (3.0 * mix[j].rho * mix[j].e + mix[j+1].rho * mix[j+1].e)
        E[j+1]  += dx / 8.0 * (mix[j].rho * mix[j].e + 3.0 * mix[j+1].rho * mix[j+1].e)
    return E

def calculate_mix_E_jacobian(gdata, grid, mix):
    dx = grid.dx
    DdxDsdot = gdata.dt * grid.deta
    dEdT = zeros((gdata.npoints,gdata.npoints))
    a = zeros(gdata.npoints-1)
    b = zeros(gdata.npoints)
    c = zeros(gdata.npoints-1)
    #e = zeros(gdata.npoints)
    for j in range(gdata.npoints-1):
        b[j]    += 1.0 / gdata.dt * dx * (3.0/8.0 * (mix[j].rho * mix[j].Cp))
        b[j+1]  += 1.0 / gdata.dt * dx * (3.0/8.0 * (mix[j+1].rho * mix[j+1].Cp))
        c[j]    += 1.0 / gdata.dt * dx * (1.0/8.0 * (mix[j+1].rho * mix[j+1].Cp))
        a[j]    += 1.0 / gdata.dt * dx * (1.0/8.0 * (mix[j].rho * mix[j].Cp))
        #e[j]    += 1.0 / gdata.dt * DdxDsdot * epsilon / 8.0 * (3.0 * gas[j].rho * gas[j
    ].h + gas[j+1].rho * gas[j+1].h)
        #e[j+1]  += 1.0 / gdata.dt * DdxDsdot * epsilon / 8.0 * (gas[j].rho * gas[j].h +
    3.0 * gas[j+1].rho * gas[j+1].h)
        #e[j]    += 1.0 / gdata.dt * DdxDsdot * epsilon / 8.0 * (3.0 * (gas[j].rho * gas[
    j].V**2 / 2.0) + (gas[j+1].rho * gas[j+1].V**2 / 2.0))
        #e[j+1]  += 1.0 / gdata.dt * DdxDsdot * epsilon / 8.0 * ((gas[j].rho * gas[j].V
    **2 / 2.0) + 3.0 * (gas[j+1].rho * gas[j+1].V**2 / 2.0))
    dEdT += diag(a,-1) + diag(c,1) + diag(b)
    return dEdT

## Jacobian system to be resolved, J*deltaT = d

def build_RHS(gdata, grid, mix): ## d
    d = zeros(gdata.npoints)
    E = calculate_mix_E(gdata, grid, mix)
    F = calculate_gas_F(gdata, grid, mix)
    G = calculate_mix_G(gdata, grid, mix)
    Q = calculate_mix_Q(gdata, grid, mix)
    d = -1/gdata.dt*E - Q + G - F
    return d

def build_J(gdata, grid, mix): ## J
```

```
    J = zeros((gdata.npoints,gdata.npoints))
    dEdT = calculate_mix_E_jacobian(gdata, grid, mix)
    dFdT = calculate_gas_F_jacobian(gdata, grid, mix)
    dGdT = calculate_mix_G_jacobian(gdata, grid, mix)
    dQdT = calculate_mix_Q_jacobian(gdata, grid, mix)
    J = dEdT + dQdT - dGdT + dFdT
    return J
```

B.3 sacram.py

```
#!/usr/bin/python

from numpy import *
from scipy.integrate import odeint
from math import exp
import numpy as nmp
import sys
import scipy
from gas import *
from mixture_energy import *

######################CLASS DEFINITION ####################

##--------------------INPUT DATA---------------------------

## node: a node has an x-dimension, x in % of total length, and x-velocity

class node(object):
    def __init__(self, x=None,eta=None,Ux=None):
        self.x = x
        self.eta = eta
        self.Ux = Ux # normal velocity of node during recession (Amar, eq 5)



## BC: boundary condition has a name, temperature, q_wall (initialised at 0 because it is
     overwritten later),
## char and virgin emissivities and boltzmann constant sigma

class BC(object):
    def __init__(self, Tw=850, qw=0, eps_solid_virgin=0.8, eps_solid_char=0.9, sigma
    =5.670373e-8):
        self.Tw = Tw
        self.qw = qw
        self.eps_solid_char = eps_solid_char
        self.eps_solid_virgin = eps_solid_virgin
        self.sigma = sigma

## gdata: includes original domain length L0, number of points, dt, t_final,the boundary
    condition

class gdata(object):

    L0 = 0.05
    npoints = 51
    dt = 1.0
    t_final = 50
    BC = BC()

## grid_data:  Stores above data for each node

class grid_data(object):
```

```
    def __init__(self,sdot=0.0, s=0.0):
        self.sdot = sdot
        self.s = s
        self.deta = 1.0/(gdata.npoints-1);
        self.dx = (gdata.L0-self.s)/(gdata.npoints-1);
        self.nodes = []
        for i in range(gdata.npoints-1):
            x = self.s + i*self.dx
            eta = (gdata.L0-x)/(gdata.L0-self.s)
            Ux = sdot * eta
            self.nodes.append(node(x, eta, Ux)) # input information for each node in
    tables
        self.nodes.append(node(gdata.L0, 0.0, 0.0)) # ended for loop. Add dimensions for
    last data point

########################## FUNCTIONS ##########################

##----------------- MATERIAL INPUT DATA ----------------------

## mass_loss: gives drho for each component of the material
#  Two for resin (A,B), one for binder (C).
#  rho_solid= resin_vol_frac*(rhoA+rhoB) + (1-resin_vol_frac)*rhoC

def mass_loss(rho0, time, T):
    k = [12000, 4.480E+09, 0]; # Thermal conductivity in Arrhenius relationship (TACOT_1
    .5.xls, pyrolysis model)
    rho_V = [300, 900, 1600]; # define virgin density
    rho_C = [0,600,1600]; # define char density
    psi = [3,3,0];
    E = [8.556E+03, 2.044E+04, 0.0]; # Activation energy
    drho = [0,0,0]; # Initialise drho at 0 0 0 - describes decomposition of each
    component

    #Arrhenius relationship - Amar (eq.13)
    drho[0] = -k[0]*rho_V[0]*((rho0[0]-rho_C[0])/rho_V[0])**psi[0]*exp(-E[0]/T);
    drho[1] = -k[1]*rho_V[1]*((rho0[1]-rho_C[1])/rho_V[1])**psi[1]*exp(-E[1]/T);
    drho[2] = -k[2]*rho_V[2]*((rho0[2]-rho_C[2])/rho_V[2])**psi[2]*exp(-E[2]/T);

    return drho

##----------------- BOUNDARY CONDITIONS -------------------------
## apply_BC: in BC, values get added to the first row of d and the first cell of J to
    account for radiation heat flux
#  Using nmp.loadtxt created lookup table for Hayabusa heat flux data

def apply_BC(gdata, grid, mix, J, d, t):
    hf = nmp.loadtxt('haya_data.txt') # lookup table for heat flux data, taken from
    Hayabusa xls 16kg
    if gdata.dt >= 0.1: # if dt >= 0.1, takes values from table (dt=0.1 in haya_data.txt)
       gdata.BC.qw = 1.0e4*hf[10*t,3] # 1.0e4 to convert from W/cm^2 to W/m^2, 10t
    because line_number=10*t
    else :
       f1 = interp1d([int(10*t),int(10*t)+1], [hf[int(10*t),3], hf[int(10*t)+1,3]]) # if
    dt<=0.1, interpolates between values from table
```

```
       gdata.BC.qw = 1.0e4*f1(10*t)
    qrad = (mix[0].solid.yC*gdata.BC.eps_solid_char+mix[0].solid.yV*gdata.BC.
    eps_solid_virgin)*gdata.BC.sigma*(mix[0].T**4-gdata.BC.Tw**4)
    d[0] += gdata.BC.qw - qrad
    J[0,0] += 4*(mix[0].solid.yC*gdata.BC.eps_solid_char+mix[0].solid.yV*gdata.BC.
    eps_solid_virgin)*gdata.BC.sigma*mix[0].T**3  # J=-d(d)/dT (Amar thesis, 5.55)

    return J, d

##------------------------- SOLVE JACOBIAN ---------------------

## integrate_in_time: Solves J.dT=d and gives mdot_g

def integrate_in_time(gdata, grid, mix, mix_old,t):

    E_n = calculate_mix_E(gdata, grid, mix_old)
    nu = 0
    max_iter = 1.0e4
    while nu < max_iter+1:
        d = build_RHS(gdata, grid, mix)
        J = build_J(gdata, grid, mix)
        d += E_n/gdata.dt
        J, d = apply_BC(gdata, grid, mix_old, J, d, t) # send the old d to the apply_BC
    function, then you update it and output new d
        Tx = nmp.linalg.solve(J,d) # solution of jacobian system, temperature increment
    calculated and added onto previous T
        #print "Tx[0] = ", Tx[0]
        for j in range(gdata.npoints):
            #if mix[j].T+Tx[j] >= 3300 :  # 3300K = limit of solid_props.data -> function
     can be blocked to prevent solver from exiting
            #   Tx[j]= 0.0
            mix[j]=mix_data(mix[j].T+Tx[j],mix[j].solid.rhoABC, mix[j].gas.V, mix[j].
    solid.rhos, mix[j].solid.rhov, mix[j].solid.rhoc, mix[j].solid.gamma, mix[j].solid.
    phi)
        if nmp.linalg.norm(Tx) < 1e-4: # precision of Tx
            print "Non-linear solver exiting at nu= ", nu
            break
        if nu == max_iter:
            print "Non-linear solver not converged!"
            print "Bailing out!"
            sys.exit()
        nu += 1
        if nu%100 == 0:
            print "nu = ", nu, "norm_Tx=", nmp.linalg.norm(Tx)

##-------------- CALCULATE MASS FLUX OF PYROLYSIS GASES ----------

    # Computes mdot_g'''

    time = linspace(t-gdata.dt,t,20); #integrates from t-dt to t over 20 points; changes
    every dt
    rhos_i = mix[0].solid.rhos # initial rhos at wall (mix(0))
    for j in range(gdata.npoints):
```

```
        res1 = odeint(mass_loss, mix[j].solid.rhoABC, time, args=(mix[j].T,))  #
    Resolution of equation (Amar thesis 6.1)--> find rhoi and thus rhoABC
        rhoABC = res1[-1]
        mix[j] = mix_data(mix[j].T,rhoABC, mix[j].gas.V, mix[j].solid.rhos, mix[j].solid.
    rhov, mix[j].solid.rhoc, mix[j].solid.gamma, mix[j].solid.phi)
    rhos_f = mix[0].solid.rhos # final rhos at wall
    print "rhos_i =", rhos_i, "---> rhos_f =", rhos_f
    mdot_g = (rhos_i-rhos_f)/gdata.dt  # Amar thesis 6.1 : solid phase continuity
    equation
    # approximate that the entirity of solid lost is transformed into gas (mdot_g = -
    mdot_s)
    print "mdot_g =", mdot_g,"kg/m^2/s"
    for j in range(gdata.npoints-1):
        mix[j].gas.V = (mix[j].solid.kappa + mix[j+1].solid.kappa) / (mix[j].gas.mu + mix
    [j+1].gas.mu) * (mix[j].gas.p - mix[j+1].gas.p)/grid.dx
    #  Here gas.V is the superficial gas velocity calculated with discrete form of the
    Dacry's law,
    #  equation 5.19 in Amar thesis with kappa=permeability and mu=viscosity
    return mdot_g

########################## Main function ##########################

def main():

    grid=grid_data()

    read_gas = nmp.loadtxt("gas_props.data")
    read_solid = nmp.loadtxt("solid_props.data")
    gas_prop_tab.T = read_gas[:,0]                        # read the contents of the
    gas_props and solid_props tables
    gas_prop_tab.rho = read_gas[:,1]            # and store each column of data to their
    respective variables.
    gas_prop_tab.M = read_gas[:,2]
    gas_prop_tab.Cp = read_gas[:,3]
    gas_prop_tab.Cv = read_gas[:,4]
    gas_prop_tab.h = read_gas[:,5]
    gas_prop_tab.mu = read_gas[:,6]
    solid_prop_tab.T = read_solid[:,0]
    solid_prop_tab.Vir_Cp = read_solid[:,1]
    solid_prop_tab.Vir_k = read_solid[:,2]
    solid_prop_tab.Vir_h = read_solid[:,3]
    solid_prop_tab.Char_Cp = read_solid[:,4]
    solid_prop_tab.Char_k = read_solid[:,5]
    solid_prop_tab.Char_h = read_solid[:,6]

##---------------------------MIX INITIAL VALUES------------------
    mix = []            # set up empty array for mix variable
    rhoABC=array([300, 900, 1600]);             # set values for rhoABC - densities of
    solid composite components A, B, C. eq11 Amar dev paper
    V = 0.0
    x_V_98 = 0.0                # initial values for virgin and char layer distance, and
    mass flow rate of gas
    x_C_02 = 0.0                # these are being reset at zero every time the loop
    executes... because they are the initial values
```

```
    t = 0
    rhos=280
    rhov = 280
    rhoc = 220
    gamma = 0.5
    phi = 0.8

##------------------------INITIALISATION ----------------------------

    for i in range(gdata.npoints):              # this will iterate through npoints
        mix.append(mix_data(gdata.BC.Tw, rhoABC, V, rhos, rhov, rhoc, gamma, phi)) #
    mix_data [gas.py]

    build_RHS(gdata, grid, mix)
    build_J(gdata, grid, mix)

    mix_old = mix

    Tfile = open('./Temp.txt', 'w')             # open output text files to record
    results
    Tfile.write("time (s)\t Tw (K)\t\t T2 (K)\t\t T3 (K)\t\t T4 (K)\t\t T5 (K)\t\t T6 (K)
    \t\t T7 (K)\t\t T8 (K)\t\t T9 (K)\n")
    Mfile = open('./Mass_loss.txt', 'w')
    Mfile.write("time (s)\t m_dot_g (kg/m2/s)\t\t rho_solid (kg/m3)\t\t rhoA_V (kg/m3)\t\
    t rhoB_V (kg/m3)\t\t rhoC_V (kg/m3)\n")

##------------------------- MAIN LOOP -------------------------------

    while t <= gdata.t_final:
        grid=grid_data(grid.sdot,grid.sdot*t)
        mdot_g = integrate_in_time(gdata, grid, mix, mix_old,t)
        print
        print "time=",t,"s", " and  q_total=", gdata.BC.qw,"W/m^2"
        print "Twall=", mix[0].solid.T,"K"
        outputT = array([t, mix[0].solid.T, mix[int(1.0e-3/(gdata.L0/(gdata.npoints-1)))
    ].solid.T, mix[int(2.0e-3/(gdata.L0/(gdata.npoints-1)))].solid.T,
                        mix[int(4.0e-3/(gdata.L0/(gdata.npoints-1)))].solid.T, mix[int
    (8.0e-3/(gdata.L0/(gdata.npoints-1)))].solid.T,
                        mix[int(12.0e-3/(gdata.L0/(gdata.npoints-1)))].solid.T, mix[int
    (16.0e-3/(gdata.L0/(gdata.npoints-1)))].solid.T,
                        mix[int(24.0e-3/(gdata.L0/(gdata.npoints-1)))].solid.T, mix[int
    (50.0e-3/(gdata.L0/(gdata.npoints-1)))].solid.T] )
        outputT.tofile(Tfile, sep='\t', format='%4.3e')
        Tfile.write('\n')


       ''' for j in range(gdata.npoints):
            if (mix[j].solid.rhos <= (mix[j].solid.rhoc + 0.98*(mix[j].solid.rhov - mix[j
    ].solid.rhoc))):
                x_V_98 = grid.nodes[j].x
            if (mix[j].solid.rhos <= (mix[j].solid.rhoc + 0.02*(mix[j].solid.rhov - mix[j
    ].solid.rhoc))):
                x_C_02 = grid.nodes[j].x
```

```
        print "x 2% char =", x_C_02
        print "x 98% virgin =", x_V_98'''

        outputM = array([t, mdot_g, mix[0].solid.rhos, mix[0].solid.rhoABC[0], mix[0].
    solid.rhoABC[1], mix[0].solid.rhoABC[2]])
        outputM.tofile(Mfile, sep='\t\t', format='%4.3e')
        Mfile.write('\n')
        mix_old = mix
        t += gdata.dt

## end of main iteration loop ##

if __name__ == '__main__':
    main()
```

Bibliography

[1] Daniel Potter. Modelling of radiating shock layers for atmospheric entry at earth and mars. 2011.

[2] Alan M Cassell, Michael W Winter, Gary A Allen, Jay H Grinstead, ME Anitmisiaris, James Albers, and Peter Jenniskens. Hayabusa reentry: Trajectory analysis and observation mission design. *AIAA Paper*, 2011.

[3] Adam Joseph Amar. Modeling of one-dimensional ablation with porous flow using finite control volume procedure. 2006.

[4] Chul Park, Richard L. Jaffe, and Harry Partridge. Chemical-Kinetic Parameters of Hyperbolic Earth Entry. *Journal of Thermophysics and Heat Transfer*, 15:76–90, 2001. doi: 10.2514/2.6582.

[5] Elaine S Oran and Jay P Boris. Numerical simulation of reactive flow. second edition. *Measurement Science and Technology*, 12(11):2021, 2001. URL http://stacks.iop.org/0957-0233/12/i=11/a=707.

[6] Dabir S. Viswanath et al. *"Viscosity of liquids"*. Springer, 2007.

[7] Antonio Viviani, Giuseppe Pezzella, and Salvatore Borrelli. Effect of finite rate chemical models on the aerothermodynamics of reentry capsules. In *Proceedings of the 15th AIAA Space Planes and Hypersonic Systems and Technologies Conferences*, volume 28, 2008.

[8] *"Eilmers Theory Book: Basic for Gas Dynamics and Thermochemistry"*.

[9] John David Anderson. *Hypersonic and high temperature gas dynamics*. Aiaa, 2000.

[10] Bianca Rose Capra. Aerothermodynamic simulation of subscale models of the fire ii and titan explorer vehicles in expansion tubes. 2007.

[11] James A Fay. Theory of stagnation point heat transfer in dissociated air. *Journal of the Aerospace Sciences*, 25(2):73–85, 1958.

[12] JN Moss EV Zoby and K Sutton. Approximate convective-heating equations for hypersonic flows. *Journal of Spacecraft and Rockets*, 18(1):64–70, 1981.

[13] Michael E Tauber and K Sutton. Stagnation-point radiative heating relations for earth and mars entries. *Journal of Spacecraft and Rockets*, 28(1):40–42, 1991.

[14] Peter A Gnoffo. Planetary-entry gas dynamics 1. *Annual Review of Fluid Mechanics*, 31 (1):459–494, 1999.

[15] Lester Lees. Laminar heat transfer over blunt-nosed bodies at hypersonic flight speeds. *Journal of Jet Propulsion*, 26(4):259–269, 1956.

[16] Henry Darcy. *Les fontaines publiques de la ville de Dijon*. V. Dalmont, 1856.

[17] Adrian E Scheidegger. The physics of flow through porous media. *Soil Science*, 86(6):355, 1958.

[18] Richard A. Thompson and Peter A. Gnoffo. "Implementation of a Blowing Boundary Condition in the LAURA Code". In *46th AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[19] Martin and Iain D. Boyd. "Strongly coupled computation of material response and nonequilibrium flow for hypersonic ablation". In *41st AIAA Thermophysics Conference*, 2009.

[20] *"The Eilmer3 Code: User Guide and Example Book"*.

[21] Yih-Kanq Chen and Frank S Milos. Finite-rate ablation boundary conditions for a carbon-phenolic heat-shield. *AIAA Paper*, 2270, 2004.