Part II

Aerothermodynamics
Chapter 4
Physical Modeling

When dealing with gases at high temperatures in which heat exchanges produce a significant effect on the flow, fluid properties such as specific heat, viscosity and thermal conductivity can no longer be considered constant as in traditional aerodynamics, but they vary with temperature, pressure and chemical composition. In fact, under those conditions, phenomena like chemical dissociation, ionization or combustion can occur and an accurate description of the flowfield requires the simultaneous examination of thermal and dynamic phenomena. This leads to the expression aerothermodynamics when referring to the multi-physical science that combines the study of classical aerodynamics with thermodynamics, combustion and thermochemistry for applications that range from external flow over space vehicles to internal flow through vehicle propulsion systems such as scramjets.

The main problems that require aerothermodynamic considerations are combustion and high-speed flight or testing. Chemical reactions sustained by combustion flow systems produce high temperatures and variable gas composition. Because of oxidation (combustion) and in some cases dissociation and ionization processes, these systems are sometimes described as aero-thermochemical. In particular, during a flight at hypersonic speed, for instance while (re-)entering a planetary atmosphere, the kinetic energy used by a vehicle to overcome drag forces is converted into compression work on the surrounding gas and thereby raises the gas temperature through a strong bow shock, as schematically shown in Fig. 4.1.

At relatively high Mach numbers, the gas temperature may raise enough to cause dissociation (Mach ≥ 7) and ionization (Mach ≥ 12). Thus the gas becomes chemically active and electrically conducting. When dealing with chemically reactive gas mixtures, the Damköhler number \( Da^c = \frac{\tau_f}{\tau_c} \) is an important adimensional parameter that can be used to describe the flow regime. It is defined by the ratio between \( \tau_f \), a characteristic time for the macroscopic processes occurring in the flow, and \( \tau_c \), a characteristic time for the chemistry. Three cases can be identified:
Figure 4.1: Schematic of the flowfield surrounding a capsule during the (re-)entry phase in a planetary atmosphere (courtesy of David Hash).

1. $Da \rightarrow 0$: the flow is frozen, meaning that no chemical reactions occur;

2. $Da \rightarrow \infty$: the chemical reactions are fast enough to reach equilibrium conditions under which the mixture composition can be directly computed from the local static pressure, temperature and elemental composition;

3. $Da \approx 1$: the flow is in chemical nonequilibrium and the mixture composition can be calculated from a set of continuity equations, one for each chemical component.

Analogously, another Damköhler number $Da^v = \frac{\tau_{\text{L}}}{\tau_v}$ can be used to define conditions of thermal equilibrium, freezing or nonequilibrium. In this case, $\tau_v$ represents the relaxation time for the equilibration of energy among the different modes, in particular the roto-translational and vibrational ones. While in thermal equilibrium one temperature is enough to describe the thermodynamics, in conditions of nonequilibrium more temperatures are needed.

The present chapter will present an up-to-date set of physical models that have been integrated in COOLFluiD, allowing us to handle all the above
mentioned thermodynamic and chemical flow regimes appropriately. To this end, the author’s work has consisted in implementing the numerous systems of equations that will be described and in defining a suitable and flexible interface for getting physical quantities from dedicated libraries. In particular, all thermodynamic, transport and chemical properties used in this thesis have been implemented into the Mutation library mainly by [94] for what concerns mixtures in chemical equilibrium/nonequilibrium and by [116] with regard to thermal nonequilibrium. Mutation has been enclosed in a COOLFluiD plug-in and has been treated almost as a black-box in the present work. Many details about the algorithms and models actually used inside Mutation will therefore be left out from the following description and the proper references will be addressed whenever needed. All the sets of governing PDE’s that will be considered can be expressed in conservative and hypervectorial form as:

\[
\frac{\partial U}{\partial t} + \frac{\partial F^c}{\partial x_i} = \frac{\partial F^d}{\partial x_i} + S
\] (4.1)

where, according to the terminology adopted in COOLFluiD, \( U \) are the conservative variables, \( P \) the update variables, \( F^c \) and \( F^d \) respectively the convective and diffusive fluxes, \( S \) the source term.

A slightly different formulation can be employed for axisymmetric cases [26] in cylindrical coordinates:

\[
\frac{\partial U}{\partial t} \frac{\partial r}{\partial P} + \frac{\partial rF^c_x}{\partial x} + \frac{\partial rF^c_r}{\partial r} = \frac{\partial rF^d_x}{\partial x} + \frac{\partial rF^d_r}{\partial r} + S
\] (4.2)

where \( x \) and \( r \) are the axial and radial directions. Both the forms 4.1 and 4.2 will be used in the following description of physical models for aerothermodynamics.

### 4.1 Navier-Stokes

In the case of the unsteady Navier-Stokes equations, describing the conservation of mass, momentum and total energy (including both internal and kinetic contributions), the conservative variables in 4.1 are:

\[
U = (\rho, \rho u, \rho E)^T
\] (4.3)

where \( \rho \) is the density, \( u \) are the velocity components and
\[
\rho E = \frac{p}{\gamma - 1} + \rho \frac{|u|^2}{2}
\]  
(4.4)

is the total energy per unity volume, \(p\) is the static pressure and \(\gamma = c_p/c_v\) is the specific heat ratio. Several choices are instead possible for the update variables \(\mathbf{P}\), for instance:

- conservative variables;
- \(\mathbf{P} = (\rho, \mathbf{u}, p)^T\), primitive variables;
- \(\mathbf{P} = (p, \mathbf{u}, T)^T\), primary variables;
- \(\mathbf{P} = (\rho, \mathbf{u}, T)^T\), natural variables.

From a computational point of view, it is generally convenient to choose primary or natural variables as \(\mathbf{P}\), because they allow numerical algorithms to readily compute the useful gradients (velocity components, temperature) appearing in the diffusive fluxes.

The convective and diffusive fluxes are defined as:

\[
\mathbf{F}^c_i = \begin{pmatrix} \rho \mathbf{u} \\ \rho \mathbf{uu} + p \mathbf{I} \\ \rho \mathbf{u} H \end{pmatrix}, \quad \mathbf{F}^d_i = \begin{pmatrix} 0 \\ \tau \cdot \mathbf{u} \\ (\bar{\tau} \cdot \mathbf{u})^T - \mathbf{q} \end{pmatrix}
\]  
(4.5)

In particular, \(H = E + p/\rho\) is the total enthalpy and the pressure is given by the perfect gas law:

\[
p = \rho \frac{R}{M} T
\]  
(4.6)

where \(R = 8314.3 \ [J/kg \ \text{mole-K}]\) is the universal gas constant and \(M\) is the molecular mass of the considered gas. Moreover, \(\bar{\tau}\) stands for the tensor of viscous stresses:

\[
\tau_{ij} = \mu \left[ \left( \frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right) - \frac{2}{3} \nabla \cdot \mathbf{u} \delta_{ij} \right]
\]  
(4.7)

computed using Stokes’ hypothesis of negligible bulk viscosity effects. The dynamic viscosity \(\mu\) is given by Sutherland’s formula:

\[
\mu = C_1 \frac{T^{\frac{3}{2}}}{T + C_2}
\]  
(4.8)
with the constants $C_1$ and $C_2$ varying from one gas to another; for air, $C_1 = 1.458 \cdot 10^{-6}$ and $C_2 = 110.4$. The heat flux $q$ in Eq. 4.5 is defined as

$$q = -\lambda \nabla T$$  \hspace{1cm} (4.9)

where $\lambda = \mu c_p / Pr$ is the gas thermal conductivity and $Pr$ is the Prandtl number, taken constant and equal to 0.72.

### 4.1.1 Axisymmetric case

#### 4.1.1.1 First formulation

In an axisymmetric case, the convective and diffusive fluxes $F_c^i$ and $F_d^i$ are formally the same as in 4.5, but the diagonal entries in the viscous stresses tensor in Eq. 4.7 must be corrected as follows:

$$\tau_{ii}^{axi} = \tau_{ii} - \frac{2}{3} \mu \frac{v}{r}$$  \hspace{1cm} (4.10)

where $v$ is the radial velocity and $r$ is the radius (in our case equal to $y$). Furthermore, as demonstrated analytically in [26], the source term

$$S = (0,0, p - \tau_{\theta\theta}, 0)^T$$  \hspace{1cm} (4.11)

must be added, yielding a viscous stress component in the circumferential direction $\theta$:

$$\tau_{\theta\theta} = -\frac{2}{3} \mu \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} - \frac{2}{r} \frac{v}{r} \right)$$  \hspace{1cm} (4.12)

#### 4.1.1.2 Second formulation

An alternative axisymmetric formulation for the Navier-Stokes equations is proposed in [155]. In this case, the set of PDE’s are written in a conventional multi-dimensional way as in 4.1, but a source term for both the convective and the diffusive parts of the equations appears:

$$S = S^c + S^d = -\frac{1}{r} \left( \begin{array}{c} \rho v \\ \rho wv \\ \rho v^2 \\ \rho vH \end{array} \right) + \frac{1}{r} \left( \begin{array}{c} 0 \\ \tau_{rx} \\ \tau_{rr} - \tau_{r\theta} \\ \tau_{rx} u + \tau_{rr} v - q_r \end{array} \right)$$  \hspace{1cm} (4.13)
where $\tau_{rx} = \mu \left( \frac{\partial u}{\partial r} + \frac{\partial v}{\partial x} \right)$ as in Eq. 4.7, $\tau_{rr}$ and $\tau_{\theta\theta}$ are given respectively by 4.10 and 4.12.

While the first axisymmetric formulation in Sec. 4.1.1.1 is especially suitable for being implemented in a FV code, this alternative form can be easily integrated in both FV and RD schemes, since the only difference with a pure 2D case lies in the presence of a source term that has to be discretized.

### 4.2 Local Thermodynamic Equilibrium

When considering a chemically reacting gas mixture in thermodynamic equilibrium, the concentration of each species can be determined from a corresponding species continuity equation [145]:

$$\frac{\partial \rho y_s}{\partial t} + \nabla \cdot (\rho y_s u) = -\nabla \cdot J_s + \dot{\omega}_s, \quad s = [1, N_s] \tag{4.14}$$

where $y_s = \rho_s/\rho$ is the mass fraction of species $s$ ($\rho_s$ and $\rho$ stand for the partial and the mixture densities), $u$ the mass-averaged mixture velocity, $J_s$ the mass diffusion flux of species $s$ and $\dot{\omega}_s$ represents the mass production/destruction term of species $s$ due to chemical reactions.

It is often more convenient to describe the chemical composition through the molar fractions $x_s = n_s/n$, where $n_s$ and $n$ represent the molar densities of individual species and of the entire mixture. We indicate with $\Phi_e$ the number of atoms of pure element $e$ (e.g. O or N in an air mixture) contained in a species $s$. The following relations define the *elemental* mole and mass fractions:

$$X_e = \frac{\sum_{s \in S} \Phi_e^s x_s}{\sum_{e \in E} \sum_{s \in S} \Phi_e^s x_s}; \quad Y_e = \sum_{s \in S} \Phi_e^s y_s \frac{M_e}{M_s}; \quad M = \sum_{s \in S} x_s M_s, \tag{4.15}$$

where $M_s$ is the molar mass of species $s$ and $M$ the mixture molar mass.

If chemical reactions are sufficiently fast with respect to other macroscopic processes occurring in the flow, i.e. the Damköhler number $Da^e$ tends to $\infty$, the mixture composition, corresponding therefore to the chemical equilibrium composition, can be computed directly from statistical mechanics for given values of pressure, temperature and elemental mass fractions $Y_e$ [23] instead of by solving Eqs.(4.14):

$$y_s = y_s(p, T, Y_e). \tag{4.16}$$
Once the composition is determined by imposing the equilibrium of a minimum set of independent chemical reactions as described in [23, 94], all thermodynamic quantities (density, internal energy, etc.) and transport properties (dynamic viscosity, thermal conductivities) can be calculated as indicated in Sec. 4.4 and 4.5.

4.2.1 LTE with Fixed Elemental Fractions

An approximation commonly used in LTE simulations consists in assuming the elemental composition constant throughout the flow and equal to the free stream values. In the case of air, for example, this corresponds to fixing the elemental molar fractions of oxygen and nitrogen as follows:

\[ X^O = 0.21; \quad X^N = 0.79. \]

In this case, the convective and diffusive fluxes in Eq. 4.5 are still formally the same in the governing equations 4.1 or 4.2, but combined with the thermodynamic relations in Sec. 4.4 and with the transport properties computed by means of the algorithms described in [94].

4.2.2 LTE with Variable Elemental Fractions

As pointed out by Murphy [102] additional advection-diffusion equations, corresponding to elemental continuity equations, should be solved simultaneously to determine the locally varying elemental mass fractions:

\[
\frac{\partial \rho Y_e}{\partial t} + \nabla \cdot (\rho Y_e \mathbf{u}) = -\nabla \cdot \mathbf{J}_e, \quad e = [1, N_e].
\]  

(4.17)

Note that the source term contribution in the right hand side in the above result is zero, translating the fact that no new elements are generated in the considered chemical reactions.

The formulation given in Ref. [141] is termed explicitly "closed" because the diffusive species and enthalpy fluxes depend in an explicit manner on the solution unknowns. In particular, the elemental mass diffusion flux \( J_e \) depends on the gradients of the elemental mass fractions \( Y_e \) and of the temperature \( T \):

\[
\mathbf{J}_e = -\rho D_e^T \nabla T + \sum_{f \in \varepsilon} \rho D_{ef} \nabla Y^f (e \in \varepsilon)
\]  

(4.18)
where $D_{ef}(p, T, Y_e)$ and $D_T(p, T, Y_e)$ are respectively the elemental multi-component diffusion coefficient and the elemental thermal demixing coefficient.

Additionally, the heat flux $\mathbf{q}$ appearing in the total energy equation needs to be modified consistently in order to account for the (de)mixing phenomenon:

$$\mathbf{q} = - (\lambda + \lambda_R + \lambda_D) \nabla T - \sum_{e \in e} \lambda_{EL}^e \nabla Y_e$$  \hspace{1cm} (4.19)

in which three transport coefficients can be identified:

1. the thermal reactive conductivity $\lambda_R$ takes into account diffusive transfer of species enthalpies in the absence of elemental diffusion [27],[30];

2. the thermal demixing conductivity $\lambda_D$ adds the contribution of the additional diffusive heat transfer that occurs due to nonzero elemental diffusive fluxes when $\nabla Y_e = 0$;

3. the elemental heat transfer coefficients $\lambda_{EL}^e$ take into account heat transfer due to elemental demixing driven by gradients in elemental composition.

Precise analytical expressions for $D_{ef}(p, T, Y_e)$, $D_T(p, T, Y_e)$, $\lambda_R$, $\lambda_D$ and $\lambda_{EL}^e$ can be found in the Appendix of [141].

This model has been applied to study and characterize incompressible flows in plasma torches [140], where the assumption of thermal and chemical equilibrium can be considered reasonable. Within this thesis work, the same model has been adopted for the first time to study 2D hypersonic re-entry flows in conditions relatively close to LTE [79], showing some potential for the heat flux prediction for this kind of applications, if compared to chemical nonequilibrium calculations.

### 4.2.2.1 Conservative vectorial form of the equations

The full system of the governing PDE’s for chemically reacting gases under LTE-VEF can be cast under the the conservative form in Eq. 4.1, where the conservative and update variables are given by

$$\mathbf{U} = (\rho, \rho \mathbf{u}, \rho E, \rho Y_e)^T, \quad \mathbf{P} = (p, \mathbf{u}, T, Y_e)^T$$ \hspace{1cm} (4.20)

and the convective and diffusive fluxes are
Herein, the viscous stresses $\bar{\tau}$ and the heat flux $q$ are given by Eqs. 4.7 and 4.19.

### 4.3 Thermo-chemical nonequilibrium

We consider hereafter the equations governing the motion of a chemically reacting and possibly weakly ionized multi-component mixture of perfect gases in thermal and chemical nonequilibrium.

Under conditions of chemical nonequilibrium, the mixture composition can be determined by solving an advection-diffusion-reaction equation for each chemical species. In particular the chemical activity is driven by mass production/destruction terms that introduce considerable uncertainty in the problem, since each possible reaction rate has to be modeled and different models can yield significantly different results depending on the temperature range.

When thermal nonequilibrium is assumed, a multi-temperature model is needed in order to account for the disequilibration in the energy distribution among the different modes. Under these circumstances, a species total energy can be considered separable into different contributions:

$$ e = e_t + e_r + e_v + e_e $$

i.e. translational, rotational, vibrational and electronic energies. While all modes are present for a molecule, only the translational and electronic ones can be associated to an atom or to a free electron.

As stressed by [50], the separation in Eq. 4.22 is mathematically possible according to the Born-Oppenheimer assumption only near the conditions of ”potential well”, i.e. the region surrounding a local minimum for the interaction potential energy. In a more realistic model, it can be proven that, in general conditions far from equilibrium (e.g. during dissociation), only translational and internal energies, without further distinction, can be analytically separated. Unfortunately, this mathematically sound novel model currently lacks of physico-chemical data and it is not yet mature.
enough to be implemented into a CFD code.

State-of-the-art multi-temperature models associate distinct temperatures to different energy modes and, in a general case [51], one may think of associating one mode $e_{\delta s}$ and related temperature $T_{\delta s}$ to each mode (or even energetic level) $\delta$ admitted by each species $s$, leading to the following expression for the energy per unity of mass:

$$e = \sum_s \frac{x_s}{M} \sum_\delta e_{\delta s}(T_{\delta s})$$

(4.23)

This approach, though theoretically possible, would be extremely complex, computationally expensive and with no guarantee that this could lead to proportionally more accurate results.

Some simplifications are therefore introduced. First of all, each mode in Eq. 4.22 is considered in equilibrium with the corresponding mode in another species, except for the vibrational one. Moreover, unlike in [65, 168], it is reasonably assumed that the translational energy of the mixture is equilibrated with the rotational energy, since this equilibration process typically takes less than 10 molecular collisions [165], leading to the following full multi-temperature model:

$$e = e_t(T^{tr}) + e_e(T^e) + e_f$$ \hspace{1cm} \text{atoms} \hspace{2cm} (4.24)

$$e = e_t(T^{tr}) + e_r(T^{tr}) + e_v(T^{v,m}) + e_e(T^e)$$ \hspace{1cm} \text{molecules} \hspace{2cm} (4.25)

$$e = e_t(T^e)$$ \hspace{1cm} \text{free electrons} \hspace{2cm} (4.26)

where $e_f$ is equivalent to the formation enthalpy for atomic species. In this case, the system is represented by:

1. a roto-translational temperature $T^{tr}$ for the heavy particles;

2. a different vibrational temperature $T^{v,m}$ for each of the diatomic and polyatomic species (molecules) in the gas mixture;

3. an electron-electronic temperature $T^e$ characterizing the energy content associated to the electronic excitation of species and to the free electron translation.

A 3-temperature model is obtained by assuming that the vibration of all molecules is described by a single vibrational temperature $T^{v}$. Further simplifications can lead to Park’s 2-temperature model [118], in which the distribution of vibrational, electronic excitation and electron translation energies
is represented by a single vibro-electronic temperature \( T^e = T^e = T^{ve} \).
Finally, a 1-temperature model corresponds to thermal equilibrium conditions.

4.3.1 3-Temperature model

We present in this section the conservation equations for a reacting gas flow under conditions of thermo-chemical nonequilibrium with the 3-temperature model [54].

4.3.1.1 Species continuity equation

The species conservation reads:

\[
\frac{\partial \rho_s}{\partial t} + \nabla \cdot (\rho_s \mathbf{u}) = -\nabla \cdot (\rho_s \mathbf{u}^d) + \dot{\omega}_s \tag{4.27}
\]

Herein, \( \mathbf{u}^d \) are the species diffusion velocities, which can be calculated, as in our case, by solving the Stefan-Maxwell system, as described in [13], or by applying Fick’s law and recasting \( \rho_s \mathbf{u}^d \) into \( -\rho D_s \nabla x_s \), where \( x_s \) are the molar fractions and \( D_s \) are diffusion coefficients [54, 128]. In both cases, an approximation has been applied: only molecular diffusion due to concentration gradients is taken into account, while that due to pressure and temperature gradients is usually neglected.

The production/destruction term \( \dot{\omega}_s \) will be described in detail in Sec. 4.6. Having assumed to have a single mass averaged velocity field \( \mathbf{u} = \sum_s \frac{\rho}{\rho} \mathbf{u}_s \) in Eq. 4.27, only one vectorial conservation equation for the mixture momentum is needed, instead of one per species in a more general case.

4.3.1.2 Mixture momentum equations

If we apply the ambipolar diffusion constrain which links electron and ion diffusion in such a way that

\[
n_e = n_i = \sum_{s=\text{ions}} n_s \tag{4.28}
\]

where \( n_e \) is the electron number density and \( n_i \) is the ionic number density, the flow field in an ionized gas mixture can be considered electrically neutral, as explained in [128]. Under these circumstances, the overall mixture momentum equation retains the exact expression as in the basic Navier-Stokes
equations in Sec. 4.1 and it is therefore not recalled here.

4.3.1.3 Vibrational energy equation

The vibrational energy conservation [54] can be expressed as follows:

\[
\frac{\partial \rho e^v}{\partial t} + \nabla \cdot (\rho e^v \mathbf{u}) = -\nabla \cdot \left( \sum_s \rho_s h_s^v \mathbf{u}_s^d \right) - \nabla \cdot \mathbf{q}^v + \Omega^{vt} + \Omega^{ve} + \Omega^{CV}. \tag{4.29}
\]

The first term on the RHS represents the diffusion of vibrational energy due to concentration gradients, while the heat conduction due to the vibrational temperature gradients is given by

\[
\mathbf{q}^v = -\lambda^v \nabla T^v \tag{4.30}
\]

where \(\lambda^v\) is the vibrational thermal conductivity. \(\Omega^{vt}\), the energy exchange (relaxation) between vibrational and translational modes due to collisions, and \(\Omega^{ve}\), the relaxation between vibrational and electronic modes, can be expressed as

\[
\Omega^{vt} = \sum_m \rho_m \frac{e_m^{v,*} - e_m^v}{\tau_m}, \quad \Omega^{ve} = \sum_m \rho_m \frac{e_m^{v,**} - e_m^v}{\tau_{em}} \tag{4.31}
\]

where \(e_m^{v,*}\) and \(e_m^{v,**}\) are the equilibrium vibrational energies of molecules \(m\) evaluated at the roto-translational and electron temperature respectively. Those energy transfer terms have been modeled with the Landau-Teller formulation [11]. The latter assumes mono-quantum energy transfers: in the collision between two molecules or between a molecule and an electron, one colliding particle can gain or lose only one energetic level, while the energy of the other particle remains unchanged. The relaxation times \(\tau_m\) and \(\tau_{em}\) are given by Millikan and White [100] with Park’s correction for high temperatures [120]:

\[
\tau_m = \tau_{MW}^M (p, T) + \frac{\tau_{park}}{(\sigma_m c_m n_m)^{-1}} \tag{4.32}
\]

where \(\sigma_m\) is the effective cross section for vibrational relaxation processes, \(c_m\) is the average molecular velocity of molecule \(m\) and \(n_m\) is the number density.
Finally, $\Omega^{CV}$ stands for the vibrational energy lost or gained due to molecular dissociation or recombination:

$$\Omega^{CV} = \sum_m \omega_m \hat{D}_m \quad (4.33)$$

Several possibilities exist for the choice of $\hat{D}_m$. The simplest possibility is to impose it equal to the vibrational energy of the molecule

$$\hat{D}_m = e_m^v \quad (4.34)$$

Alternatively, if one assumes preferential dissociation and recombination of molecules in the higher vibrational states, it can be taken equal to some fraction of the molecular dissociation energy $D_m$:

$$\hat{D}_m = c_1 D_m \quad (4.35)$$

with $0 < c_1 \leq 1$. In comparison with the case in Eq. 4.34, this model tends to lower the vibrational temperature behind the shock, where dissociation occurs, and increase it inside the boundary layer, where recombination occurs.

### 4.3.1.4 Electron and electronic excitation energy equation

The equation can be written as [54]:

$$\frac{\partial \rho^e}{\partial t} + \nabla \cdot [(\rho^e + p_e) \mathbf{u}] = \mathbf{u} \nabla p_e - \nabla \cdot \left( \sum_s \rho_s h^e_s \mathbf{u}_s \right) - \nabla \cdot \mathbf{q}^e + \Omega^{et} - \Omega^I - \Omega^{ve} - Q_{\text{rad}} \quad (4.36)$$

where $p_e$ is the electron pressure and it is defined as

$$p_e = \rho_e \frac{R}{M_e} T_e \quad (4.37)$$

The term $\mathbf{u} \nabla p_e$ represents the work done by the electric field induced by the gradient of electron pressure.

As in [128], we can partially reformulate this and the convective term in Eq. 4.36 in order to keep the same formalism as in the vibrational-electronic equation for Park’s 2-temperature in Sec. 4.3.2:

$$\frac{\partial \rho^e}{\partial t} + \nabla \cdot (\rho^e \mathbf{u}) = -p_e \nabla \mathbf{u} \quad \cdots \text{ exactly as in Eq. 4.36} \quad (4.38)$$
The diffusion of species electronic enthalpies due to concentration gradients is expressed in Eq. 4.36 by $-\nabla \cdot \left( \sum_s \rho_s h_s^e \mathbf{u}_s^d \right)$, while the heat conduction due to electron temperature gradients is given by

$$q^e = -\lambda^e \nabla T^e$$

(4.39)

with $\lambda^e$ being the electronic thermal conductivity. 

$\Omega^{et}$ represents the energy exchange due to inelastic collisions between the electrons and the heavy particles:

$$\Omega^{et} = 2\rho_e \frac{3}{2} R(T - T_e) \sum_{s \neq e} \frac{\nu_{e,s}}{M_s}$$

(4.40)

where $\nu_{e,s}$ is the effective collision frequency of electron with heavy particles which is defined in [54].

$\Omega^I$ corresponds to the energy loss due to electron impact ionization and is given by

$$\Omega^I = \sum_{\text{ions}} \dot{n}_{e,s} \dot{I}_s$$

(4.41)

where $\dot{n}_{e,s}$ is the molar rate of production of species $s$ and $\dot{I}_s$ is the energy lost per unit mole by a free electron in producing species $s$ by electron impact ionization [128].

The relaxation between electronic and vibrational modes is represented by $\Omega^{ve}$, whose expression is the same as in 4.31, but it appears in Eqs. 4.29 and 4.36 with opposite signs.

The last term in Eq. 4.36 accounts for the rate of energy loss due to radiation during electronic transitions, but it is neglected here, as in [54, 128].

### 4.3.1.5 Total energy equation

The conservation of the overall energy for the gas mixture is expressed by

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho H \mathbf{u}) = -\nabla \cdot \left( \sum_s \rho_s h_s \mathbf{u}_s^d \right) - \nabla \cdot \mathbf{q} + \nabla \cdot (\vec{\tau} \cdot \mathbf{u}) - Q_{\text{rad}}$$

(4.42)

The first term on the RHS is the diffusion of species enthalpy due to concentration gradients. The global heat conduction $\mathbf{q}$ groups together all the contributions due to roto-translational, vibrational and electron temperature gradients:
\[ \mathbf{q} = - (\lambda \nabla T + \lambda^v \nabla T^v + \lambda^e \nabla T^e) \] (4.43)

The work done by the shear forces is represented by \( \nabla (\bar{\pi} \cdot \mathbf{u}) \). Finally, \( Q_{\text{rad}} \), the rate of energy loss due to radiation, is neglected, and so is the work done on the charged particles by the electric field (here not reported). The latter is considered null, because the ambipolar diffusion constraint in Eq. 4.28 is assumed.

### 4.3.1.6 Conservative vectorial form of the equations

All the equations described in Secs. 4.3.1.1-4.3.1.5 can be recast into the vectorial form of Eq. 4.1, with the conservative variables

\[ \mathbf{U} = (\rho_s, \rho u, \rho E, \rho e^v, \rho e^e)^T \] (4.44)

and the natural variables [52] as update variables

\[ \mathbf{P} = (\rho_s, u, T, T^v, T^e)^T. \] (4.45)

The convective and diffusive fluxes are given by:

\[
\begin{pmatrix}
\rho_s \mathbf{u} \\
\rho \mathbf{uu} + p \hat{I} \\
\rho \mathbf{uH} \\
\rho \mathbf{ue}^v \\
\rho \mathbf{ue}^e
\end{pmatrix},
\begin{pmatrix}
-\rho_s \mathbf{u}_s^d \\
(\bar{\pi} \cdot \mathbf{u})^T - \mathbf{q} - \sum_s \rho_s h_s \mathbf{u}_s^d \\
-\sum_s \rho_s h_s \mathbf{u}_s^d - \mathbf{q}^v \\
-\sum_s \rho_s h_s \mathbf{u}_s^d - \mathbf{q}^e
\end{pmatrix}.
\] (4.46)

and the vector of source terms

\[
\begin{pmatrix}
\dot{\omega}_s \\
0 \\
0 \\
\Omega^{et} + \Omega^{ve} + \Omega^{CV} \\
\Omega^{et} + \Omega^{I} - \Omega^{ve}
\end{pmatrix}
\] (4.47)

Even though the 3-temperature model has been implemented in COOLFluiD, it did not yield fully satisfactory results (in particular, the electron temperature tended to inexplicably increase in the free stream before the shock) and therefore we will not present them. In fact, this model is often mentioned in literature as in [54, 128], but rarely used [65]. Moreover, there are
still open issues on the physical validity of the non-conservative electron and electronic excitation energy equation and alternatives are being sought, as recently discussed in [166].

4.3.2 2-Temperature model

The 2-temperature or Park’s model [119] associates one temperature $T$ for describing the distribution of the roto-translational energy of heavy particles and a second temperature $T^{ve}$ for characterizing the vibrational, electronic and electron translational energies. According to [119], this approximation is justified by (1) the rapid energy transfer between the translational mode of free electrons and the vibrational mode of molecular nitrogen and (2) the rapid equilibration of the low-lying electronic states of heavy particles with the ground electronic state at the electronic temperature. This model is obtained from the 3-temperature one by merging Eqs. 4.29 and 4.36 into a single equation for the conservation of the vibrational-electronic energy:

$$
\frac{\partial \rho_{e^{ve}}}{\partial t} + \nabla \cdot (\rho_{e^{ve}} \mathbf{u}) = -p_e \nabla \cdot \mathbf{u} - \nabla \cdot \left( \sum_s \rho_s h_s^{ve} \mathbf{u}_s^d \right) - \nabla \cdot \mathbf{q}^{ve} + \nabla \cdot \Omega^{et} + \Omega^{et} - \Omega^I + \Omega^{CV} - Q_{rad}
$$

(4.48)

where $\mathbf{q}^{ve} = -(\lambda^v + \lambda^e) \nabla T^{ve}$ is the heat flux due to gradients of the vibrational-electronic temperature and all the other individual quantities have been defined earlier.

4.3.3 Multi-Temperature model (neutral mixtures)

In the present work, when dealing with neutral air mixture flows, we make use of a multi-temperature model which associates one temperature $T$ to the roto-translational energy modes and 2 or more temperatures to the vibrational modes of molecular species indicated with $m$ ($O_2$, $N_2$, $NO$). This model can be summarized in conservative and hypervectorial form as in Eq. 4.1, where the variables $\mathbf{U}$ and $\mathbf{P}$ are defined as:

$$
\mathbf{U} = (\rho_s, \rho u, \rho E, \rho_m e^v_m)^T, \quad \mathbf{P} = (\rho_s, u, T, T^{ve}_m)^T
$$

(4.49)

and the fluxes are given by:
\[
F^c_i = \begin{pmatrix}
\rho_s u \\
\rho u u + p \hat{I} \\
\rho u H \\
\rho_m u_c^v
\end{pmatrix}, \quad F^d_i = \begin{pmatrix}
-\rho_s u_s^d \\
\bar{\tau} (\bar{\tau} \cdot u)^T - q - \sum_s \rho_s h_s u_s^d \\
-\rho_m h_m^d u_m^d - q_m^v
\end{pmatrix}
\] (4.50)

where the heat flux corresponding to the gradient of the vibrational temperature associated to molecule \( m \) is defined as

\[
q_m^v = -y_m \lambda_m^v \nabla T_m^v
\] (4.51)

where \( y_m \) is the mass fraction of molecule \( m \).

Finally, the source term is expressed by:

\[
S = \begin{pmatrix}
\dot{\omega}_s \\
0 \\
0 \\
\Omega^{vt} + \Omega^{CV} + \Omega^{VV}
\end{pmatrix}
\] (4.52)

where \( \Omega^{VV} \), accounting for the relaxation between vibrational modes of two molecules, has been neglected in this work.

### 4.3.4 Implementation issues

All the system of equations that have been presented have been implemented in COOLFluiD by applying the the Perspective pattern presented in Sec. 2.3.1. In order to show the extensibility of such a design solution to complex physical models, we can show its application to the concrete case of the multi-species and multi-temperature model for thermo-chemical nonequilibrium described in Sec. 4.3.

#### 4.3.4.1 Example: Implementation of TCNEQ model

A class diagram corresponding to the application of the Perspective pattern to the TCNEQ model, as it is implemented in COOLFluiD, is presented in Fig. 4.2.

The class corresponding to the concrete physical model for thermo-chemical nonequilibrium NavierStokesNEQ is defined in C.L. 4.1.

In this case, ConvectionDiffusionReaction is the compositor object that binds together three different equations terms: convective, diffusive and reaction (and energy transfer) term.
Figure 4.2: Perspective pattern applied to a TCNEQ model.

1 // --- NavierStokesNEQ.hh --- //
2 class NavierStokesNEQ : public ConvectionDiffusionReaction
3     <MultiScalarTerm<EulerTerm>, NEQDiffTerm, NEQSourceTerm> {  
4     public:
5         // constructor, virtual destructor
6         virtual int getDimension() const;
7         virtual int getNbEquations() const;
8         virtual int setup() const;
9     };

Code Listing 4.1: NavierStokesNEQ class definition

The convective term is obtained by parameterizing the MultiScalarTerm, introduced in Sec. 2.3.1.3, with the EulerTerm defined in C.L. 2.17. This has the effect of attaching two subsets of physical data to EulerTerm, one corresponding to the species mass fractions \(y_s\), the other to the vibrational (and electron-electronic excitation) energies per unit mole \(e^v_m\) (and \(e_e\)) for which a conservation equation is considered.

The diffusive term NEQDiffTerm in C.L. 4.2 derives from the NSTerm class in C.L. 2.18, inherits access to the underlying library (e.g. Mutation or Pegase) for computing additional transport properties (vibrational thermal conductivities, diffusion coefficients ...) and stores them.
The class \texttt{NEQSourceTerm} gives access to an underlying library (e.g. Mutation) for computing the mass production terms and all the source terms due to energy transfer, as shown in C.L. 4.3.

\begin{verbatim}
// --- NEQSourceTerm.hh --- //
class NEQSourceTerm : public BaseTerm {
  public:
    // the values of the source terms are stored
    virtual int getDataSize() const
    { return m_sourceTerms.size(); }

    static string getName() { return "NEQSource"; }

  private:
    SafePtr<ChemLibrary> getChemLibrary() const;
    SafePtr<TFLibrary> getTransferLibrary() const;
};
\end{verbatim}

\textbf{Code Listing 4.3:} NEQSourceTerm class definition

It can be noticed that, whenever possible, full reuse of existing terms is achieved. The data set is simply extended to accommodate the new model.

\textbf{TCNEQ VarSets.} As explained in Sec. 2.3.1.2, convective VarSets are responsible for the computation of physical fluxes, jacobian (rotated) matrices, eigenvectors and eigenvalues, while keeping knowledge of the actual type of variables in function of which all those quantities are calculated. In the TCNEQ case, we normally store the solution in natural
variables $P$ and we make also use of the conservative variables $U$. This corresponds to the definition of two distinct concrete convective $\text{VarSets}$, namely $\text{EulerNEQRhoivtTv}$ (update variables) and $\text{EulerNEQCons}$ (conservative variables), as shown in Fig 4.2. They basically reimplement part of the interface for a $\text{ConvectiveVarSet}$ (i.e. the polymorphic type which is exposed to numerical algorithms) shown in C.L. 2.20, but reusing as much code as possible from $\text{EulerVarSet}$ (see C.L. 2.22). For example, C.L. 4.4 shows that $\text{EulerNEQRhoivtTv}$ derives from a $\text{MultiScalarVarSet}$ (see C.L. 2.25) parameterized with $\text{EulerVarSet}$, which takes care of computing the fluxes and eigenvalues corresponding to the basic Euler equations plus the additional entries due to the advection of partial densities and vibrational (and electron-electronic) energies.

```
1 // --- EulerNEQRhoivtTv.hh --- //

2 class EulerNEQRhoivtTv : public MultiScalarVarSet<EulerVarSet> {
3     public:
4         // constructor, virtual destructor,
5         // overridden parent virtual methods
6     }
7 ...  
```

**Code Listing 4.4:** EulerNEQRhoivtTv interface

Since the implementation of $\text{MultiScalarVarSet}$ is generic, the same procedure is reused for many other cases where additional equations are plugged to the basic Euler core (LTE-VEF, turbulence ...). All thermodynamic quantities (pressure, enthalpy, speed of sound ...) are computed by means of the acquainted thermodynamic library (Mutation, in our case) and temporarily stored in the corresponding convective term, $\text{MultiScalarTerm<EulerTerm>}$.

Similarly, $\text{NavierStokesNEQVarSet}$ inherits from $\text{NavierStokesVarSet}$, provides the implementation of the diffusive fluxes, delegates part to the implementation to variable-dependent subclasses like $\text{NavierStokesNEQRhoivtTv}$ and queries the transport library (e.g. Mutation) in $\text{NEQDiffTerm}$ for the computation of transport properties.

For sake of consistency, a source term $\text{VarSet}$ could also be defined, but, in our case, we shortcut the access to $\text{NEQSourceTerm}$ via a Strategy object ($\text{ComputeSourceTerm}$) that takes also care of discretizing the source term and that is defined in the module corresponding to the chosen numerical algorithm. This is just a simplification that shows once again the flexibility and tunability of our design solution.
4.4 Thermodynamics

4.4.0.2 Equation of state

According to Dalton’s law of partial pressures, under the assumption that each chemical component behaves as a perfect gas and therefore obeys Eq. 4.6, the equation of state for a gas mixture under conditions of thermal nonequilibrium becomes

\[ p = p_e + \sum_{s \neq e} p_s = \rho R \left( T_e \frac{y_e}{M_e} + T \sum_{s \neq e} \frac{y_s}{M_s} \right) \]  

(4.53)

where \( p_e \) represents the electron pressure and \( T^e = T \) if thermal equilibrium is assumed. In the latter case, if also chemical equilibrium holds (LTE), once that the mixture composition is known, either in terms of mass fractions \( y_s = \rho_s/\rho \) or molar fractions \( x_s = y_s M/M_s \), density is calculated from Eq. 4.53 and from

\[ \rho = \sum_s \rho_s \]  

(4.54)

If the case of nonequilibrium, Eq. 4.53 is used to calculate pressure from the local values of partial densities \( \rho_s \) and temperatures \( T \) and \( T^e \).

4.4.0.3 Mixture energy

The mixture internal energy per unit mass is given by

\[ e = \sum_s y_s e_s = \frac{1}{M} \sum_s x_s M_s e_s \]  

(4.55)

where different contributions are considered for each species energy, according to the separation scheme introduced in Eq. 4.22:

\[ e_s = e_s^0 + (e_{v,s}^t + c_{v,s}^r) \frac{e_s}{T_s^0} + e_s^v(T_s^0) + e_s^{el}(T^e) \]  

(4.56)

where each term is provided by quantum mechanics. In particular, \( e_s^0 \) can be calculated from

\[ h_s^0 = e_s^0 + \frac{R}{M_s} T_{\text{ref}} \]  

(4.57)
in which $h_0^\text{S}$ is the *heat of formation* and $T_{r,ef}$ is chosen equal to 0.

**Translational and rotational energy.** The specific heat ratio at constant volume for the translational and rotational modes are computed as:

$$c^t_{v,s} = \frac{3}{2} \frac{R}{M_s}, \quad c^r_{v,s} = \frac{d}{2} \frac{R}{M_s}$$

(4.58)

where

- $d = 0$ for monoatomic molecules;
- $d = 2$ for diatomic and inline polyatomic molecules;
- $d = 3$ for non-inline polyatomic molecules.

**Vibrational energy.** The expression of the vibrational energy $e^v_s$ is derived by assuming that the internal quantum states are populated according to a Boltzmann distribution, and that the molecule $s$ behaves as a harmonic oscillator. It reads:

$$e^v_s = \frac{R}{M_s} \frac{\theta^v_s}{e^{\theta^v_s/T^v_s} - 1}$$

(4.59)

where $T^v_s$ is the vibrational temperature of molecule $s$ and $\theta^v_s$ a characteristic temperature for vibration.

**Electronic energy.** The species electronic energy $e^\text{el}_s$ per unit mass in Eq. 4.55 is computed as in [13]:

$$e^\text{el}_s = \frac{R}{M_s} \frac{\sum_{e=0}^{\infty} g^\text{el}_{s,e} \theta^\text{el}_{s,e} \exp(-\theta^\text{el}_{s,e}/T^e)}{Q^\text{el}_s}$$

(4.60)

where $Q^\text{el}_s$ is the electronic partition function which is defined as

$$Q^\text{el}_s = \sum_{e=0}^{\infty} g^\text{el}_{s,e} \exp(-\theta^\text{el}_{s,e}/T^e)$$

(4.61)

Herein, the series in 4.60 and 4.61 are mathematically divergent and must be conveniently truncated at an energy level $e_{\text{max},s}$, different for each species. The characteristic electronic temperatures $\theta^\text{el}_{s,e}$ and the degeneracies $g^\text{el}_{s,e}$...
associated to each energy level $e$ and species $s$ can be found in [116].

The electronic energy per unit volume appearing in Eq. 4.36 includes all the electronic excitation energies of heavy particles and the free electron translational energy:

$$\rho e^e = \rho e c_{v,e} T^e + \sum_{s \neq e} \rho s e^s$$

(4.62)

### 4.4.0.4 Mixture enthalpy

As far as the mixture enthalpy is concerned, it can be expressed as:

$$h = \sum_s y_s h_s$$

(4.63)

in which the species enthalpies are given by

$$h_s = h_s^0 + (c_{p,s}^t + c_{p,s}^r) T + h_s^v(T_s^v) + h_s^e(T^e)$$

(4.64)

where the following relations hold:

$$c_{p,s}^t = c_{v,s}^t + \frac{R}{M_s}, \quad c_{p,s}^r = c_{v,s}^r$$

(4.65)

$$h_s^v = e_s^v, \quad h_s^e = \begin{cases} e_s^e & \text{for } s \neq e \\ RT^e/M_e & \text{for } s = e \end{cases}$$

(4.66)

### 4.5 Transport Properties

The dynamic viscosity $\mu$ and the translational thermal conductivity $\lambda^t$ are computed in the Mutation library by applying the efficient iterative algorithms described in [94, 95] to the rigorous expressions derived from kinetic theory [60] for single temperature gas mixtures. This approach is supposed to be more accurate than the one followed by most of the numerical solvers for chemically reacting flows [7, 54, 105, 128], all resorting to mixture rules such as Yos’ [173], Lee’s [81] or Blottner’s (1971).

The rotational, vibrational and electronic thermal conductivities are modeled by means of the Eucken approximation [56] which requires the corresponding $c_p$’s, i.e. specific heat coefficients at constant pressure.

The electron thermal conductivity and electrical conductivity are computed
with the formulas due to Devoto [151], where two non-vanishing Sonine polynomial contributions were found to yield accurate results [59]. The diffusion fluxes $\rho_s u^d_s$ have been computed solving the Stefan-Maxwell system [7, 13, 14] of equations which consist of a linear system (in the diffusion fluxes) of as many equations as the chemical species are present in the mixture. This system is supplemented by the auxiliary condition that the sum of the diffusion fluxes is zero plus the ambipolar constraint introduced in Eq. 4.28 [159].

### 4.6 Chemical kinetic model

We consider a general set of *elementary* chemical reactions involving $N_s$ species:

$$\sum_{s=1}^{N_s} \alpha_{s,r} \Psi_s = \sum_{s=1}^{N_s} \beta_{s,r} \Psi_s$$

(4.67)

where $\Psi_s$ indicate the symbols of the chemical components and $\alpha_{s,r}$ and $\beta_{s,r}$ the stoichiometric coefficients for reactants and products respectively. The time rate of production of species per unit volume, $\dot{\omega}_s$ appearing in the species continuity equations is derived following the kinetic processes occurring in the system. Many kinetic models are available and include the identification of set of reactions like the ones in Eq. 4.67 and the specification of appropriate constants for the evaluation of the corresponding reaction rates. $\dot{\omega}_s$ can be expressed as [11]:

$$\dot{\omega}_s = M_s \sum_{r=1}^{N_r} (\beta_{s,r} - \alpha_{s,r}) \Xi_r (R_{f,r} - R_{b,r})$$

(4.68)

where $N_r$ is the number of reactions belonging to the chosen chemical kinetic model, $R_{f,r}$ and $R_{b,r}$ are the forward and backward reaction rates, which are given by

$$R_{f,r} = k_{f,r} \prod_{s=1}^{N_s} \left( \frac{\rho_s}{M_s} \right)^{\alpha_{s,r}} \quad R_{b,r} = k_{b,r} \prod_{s=1}^{N_s} \left( \frac{\rho_s}{M_s} \right)^{\beta_{s,r}}$$

(4.69)

where $k_{f,r}$ and $k_{f,r}$ are the forward and backward reaction rate coefficients. Moreover, $\Xi_r = \sum_{s=1}^{N_s} \xi_{s,r} \rho_s / M_s$ represents the third body contribution to the dissociation or recombination reactions, with efficiency $\xi_{s,r}$ for reaction $r$. 
Any individual species can be excluded or included as third body in a reaction by setting the corresponding $\xi_{s,r}$ to 0 or 1 respectively.

### 4.6.1 Reaction rates

The forward reactions rates $k_{f,r}$ introduced in Eq. 4.69 are given by Arrhenius’ law:

$$k_{f,r}(T_1) = A_{f,r}T_1^{n_{f,r}}\exp(-E_{f,r}/kT_1) \quad (4.70)$$

Herein, $A_{f,r}, n_{f,r}$ and $E_{f,r}$ are determined experimentally for each reaction and are provided by the chemical kinetic models. $E_{f,r}$ is the activation energy, i.e. an energy threshold that must be crossed to activate the corresponding reaction.

The backward reactions rates $k_{b,r}$ are computed from the relation

$$k_{b,r}(T_2) = k_{f,r}(T_2)/K_{eq,r}^{c}(T_2) \quad (4.71)$$

where $K_{eq,r}^{c}$ is the *equilibrium reaction rate constant* which can be expressed as:

$$K_{eq,r}^{c} = e^{-\Delta G_r^0/RT(\Delta n)} \quad (4.72)$$

in which $\Delta G_r^0 = \sum_{\text{products}} G_s^0 - \sum_{\text{reactants}} G_s^0$ is the difference between Gibbs’ free energy between the products and reactants and $\Delta n_r$ is the difference in the number of moles of products and reactants.

In a multi-temperature context, $T_1$ and $T_2$ are the forward and backward reaction rate controlling temperatures. In Park’s model [119, 121], for instance, the rate controlling temperature is empirically defined as

$$T_i = \sqrt{T_aT_b}, \quad i = 1, 2 \quad (4.73)$$

where $a$ and $b$ can be chosen among $T$, $T^v$ or $T^e$, with different combinations according to the reaction type, as described in detail in [128].

More advanced CVDV models which couple vibration and dissociation more consistently, such as Treanor-Marrone’s [97] or Knab’s [110], require a more complex definition of the reaction rate coefficients in function of the involved temperatures.
4.6.2 Air chemistry model

Most of our simulations in TCNEQ conditions are applied to air mixture flows. At ambient temperature, air can be assumed to be made of 79% in volume of \(N_2\) and 21% of \(O_2\), even though some minor components like argon \((Ar)\), carbon dioxide \((CO_2)\) and neon \((Ne)\) are present. As temperature increases, chemical reactions occur and other species appear. Molecular oxygen is the first to dissociate in a temperature range 2000-4000 \(K\) at a pressure of 1 atm. By reaction between \(N_2\) and atomic oxygen, nitric oxide is produced above 2000 \(K\) and then the main phase of dissociation takes place in the range 3500-8000 \(K\). At around 4000 \(K\), nitric oxide ion \(NO^+\) begins to form and for temperatures higher than 6000 \(K\) \(O^+\) and \(N^+\) appear. In any case, the lower the pressure, the lower the temperature at which dissociation activates, with the result of having, for example, fully dissociated oxygen at 3000 \(K\) at 100 \(Pa\) and at 5000 \(K\) at 10000 \(Pa\). Depending on the pressure and temperature range, we can roughly distinguish between a regime in which the degree of ionization is negligible and air is well represented by a 5-species mixture \((N,O,N_2,NO,O_2)\) and another one in which air can be considered as a 11-species ionized mixture \((e^-,N,O,N_2,NO,O_2,N^+,O^+,N_2^+,NO^+,O_2^+)\).

In order to take into account all the chemical activity occurring in air in different regimes, several sets of reaction are considered, as shown in Fig. 4.3:

1. thermal dissociation of molecules by collision with all [120] or some [56] of the heavy particles;
2. bimolecular exchange reactions, which are the main responsible for the production of \(NO\) and remove \(N_2\) from the mixture more efficiently than dissociation;
3. associative ionization or dissociative recombination;
4. charge exchange reactions, which, whenever impact ionization is neglected, are the only responsible for the creation of atomic ions;
5. heavy particle impact ionization, which are present only in some models [56] and have generally a negligible effect, due to their very high activation energy;
6. electron impact ionization of atomic \(N\) and \(O\), which also have a very high activation energy, but once triggered, they cause an exponential increase of the free electrons number density.
<table>
<thead>
<tr>
<th>Thermal dissociation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2 + X \rightleftharpoons O + O + X$</td>
</tr>
<tr>
<td>$N_2 + X \rightleftharpoons N + N + X$</td>
</tr>
<tr>
<td>$NO + X \rightleftharpoons N + O + X$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bimolecular exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2 + N \rightleftharpoons NO + O$</td>
</tr>
<tr>
<td>$N_2 + O \rightleftharpoons NO + N$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Associative ionization-dissociative recombination</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N + O \rightleftharpoons NO^+ + e^-$</td>
</tr>
<tr>
<td>$N + N \rightleftharpoons N_2^+ + e^-$</td>
</tr>
<tr>
<td>$O + O \rightleftharpoons O_2^+ + e^-$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Charge exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>$NO^+ + O \rightleftharpoons N^+ + O_2$</td>
</tr>
<tr>
<td>$O_2^+ + N \rightleftharpoons N^+ + O_2$</td>
</tr>
<tr>
<td>$O^+ + NO \rightleftharpoons N^+ + O_2$</td>
</tr>
<tr>
<td>$N^+ + N_2 \rightleftharpoons N_2^+ + N$</td>
</tr>
<tr>
<td>$O_2^+ + N_2 \rightleftharpoons N_2^+ + O_2$</td>
</tr>
<tr>
<td>$O^+ + N_2 \rightleftharpoons N_2^+ + O$</td>
</tr>
<tr>
<td>$NO^+ + N \rightleftharpoons N_2^+ + O$</td>
</tr>
<tr>
<td>$O_2^+ + O \rightleftharpoons O^+ + O_2$</td>
</tr>
<tr>
<td>$NO^+ + N \rightleftharpoons O^+ + N_2$</td>
</tr>
<tr>
<td>$NO^+ + O_2 \rightleftharpoons O_2^+ + NO$</td>
</tr>
<tr>
<td>$NO^+ + O \rightleftharpoons O_2^+ + N$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Heavy particle impact ionization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_2 + N_2 \rightleftharpoons NO + NO^+ + e^-$</td>
</tr>
<tr>
<td>$NO + X \rightleftharpoons NO^+ + e^- + X$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Electron impact ionization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O + e^- \rightleftharpoons O^+ + e^- + e^-$</td>
</tr>
<tr>
<td>$N + e^- \rightleftharpoons N^+ + e^- + e^-$</td>
</tr>
</tbody>
</table>

**Figure 4.3:** Chemical reactions model for 11-species air (from [13]).
Chapter 5

Numerical methods

5.1 Implicit time discretization

The hyperbolic system in Eq. 4.1 must be discretized in time and space in order to provide a numerical solution. As far as time discretization is concerned, unconditionally stable fully implicit schemes are most suitable for the viscous and generally stiff applications which are treated in this thesis, because they allow the simulation to converge to steady state with relatively high time steps and in many less iterations, if compared to what explicit or semi-implicit schemes can yield. The good performance in terms of convergence rates of this kind of algorithms is partially counter-balanced by the large memory requirements needed to store the system matrix. However, this problem is being increasingly alleviated by the concurrent growing power and storage capabilities of the computer architectures and, mainly, by the significant performance gain offered by distributed computing. Keeping this in mind, we chose to devote our effort towards the implementation in COOLFluiD of a flexible Newton method which is able to tackle multiple weakly coupled systems of equations, as described hereafter. The algorithm represents a generalization of the one proposed in [7], but it is formulated to be independent from the choice of the actual implicit time stepping scheme, or space discretization, or variables in which the update is performed. Moreover, full profit of the available computational resources can be taken, since the overall algorithm has also been parallelized.

5.1.1 Newton method for weakly coupled systems

When integrating in space the system of PDE’s in Eq. 4.1, we obtain the following system of conservation laws:

\[
\frac{d}{dt} \int_{\Omega} U \ d\Omega + \int_{\Omega} \nabla \cdot \mathbf{F}^c \ d\Omega = \int_{\Omega} \nabla \cdot \mathbf{F}^d \ d\Omega + \int_{\Omega} \mathbf{S} \ d\Omega \quad (5.1)
\]
After having decomposed the system of equations 5.1 into a number \( N_s \geq 1 \) of weakly coupled equation subsystems, we apply the selected space discretization to it. We rewrite the system of equations and we include all the spatially discretized terms in the residuals \( \mathbf{R}_s(\mathbf{U}) \), defining the pseudo-steady residual \( \tilde{\mathbf{R}}_s(\mathbf{U}) \), as follows:

\[
\tilde{\mathbf{R}}_s(\mathbf{P}) = \frac{d}{dt} \int_\Omega \mathbf{U}_s \, d\Omega + \mathbf{R}_s(\mathbf{P}) = 0, \quad s = 1 \ldots N_s
\]  

(5.2)

where \( \mathbf{U} \) are the conservative variables and \( \mathbf{P} \) are the variables in which the solution is stored and updated, e.g. primitive, natural, conservative etc. \( \mathbf{U}_s = \mathbf{U}_s(\mathbf{P}) \) represents the subset of \( n_s \) conservative variables whose update is associated to the solution of the subsystem \( s \):

\[
\mathbf{U}_s = [\mathbf{U}_{s0}, \mathbf{U}_{s1}, \ldots, \mathbf{U}_{sn_s-1}]
\]  

(5.3)

The steady residual \( \mathbf{R}_s(\mathbf{P}) \) includes the convective fluxes \( \mathbf{F}_s^c \), the diffusive fluxes \( \mathbf{F}_s^d \), and the source terms \( \mathbf{S}_s \) for subsystem \( s \):

\[
\mathbf{R}_s(\mathbf{P}) = \int_\Sigma (\mathbf{F}_s^c - \mathbf{F}_s^d)(\mathbf{P}) \cdot \mathbf{n} \, d\Sigma - \int_\Omega \mathbf{S}_s(\mathbf{P}) \, d\Omega
\]  

(5.4)

The actual time discretized expression for the pseudo-steady residual \( \tilde{\mathbf{R}}_s(\mathbf{P}) \) depends on the choice of the scheme for the time integration. As an example, under the assumption of constant volume \( \Omega \), i.e. of non-moving mesh, we have:

\[
\tilde{\mathbf{R}}(\mathbf{P}) = \frac{\mathbf{U}(\mathbf{P}) - \mathbf{U}(\mathbf{P}^n)}{\Delta t} \Omega + \mathbf{R}(\mathbf{P}) \quad \text{Backward Euler (BE)}
\]  

(5.5)

\[
\tilde{\mathbf{R}}(\mathbf{P}) = \frac{\mathbf{U}(\mathbf{P}) - \mathbf{U}(\mathbf{P}^n)}{\Delta t} \Omega + \frac{1}{2} [\mathbf{R}(\mathbf{P}) + \mathbf{R}(\mathbf{P}^n)] \quad \text{Crank-Nicolson (CN)}
\]  

(5.6)

\[
\tilde{\mathbf{R}}(\mathbf{P}) = \frac{3\mathbf{U}(\mathbf{P}) - 4\mathbf{U}(\mathbf{P}^n) + \mathbf{U}(\mathbf{P}^{n-1})}{2\Delta t} \Omega + \mathbf{R}(\mathbf{P}) \quad \text{3-Point Backward (3B)}
\]  

(5.7)

where BE is a first order time accurate scheme, while both CN and 3B yield second order accuracy. After having performed a Taylor expansion in time around \( \tilde{\mathbf{R}}_s(\mathbf{P}^n) \) we get:
\[ \tilde{R}_s(P^{n+1}) = 0 \Rightarrow \tilde{R}_s(P^{n+1}) = \tilde{R}_s(P^n) + \frac{\partial \tilde{R}_s}{\partial P_s}(P^n) \Delta P_s^n = 0 \quad (5.8) \]

where we have neglected the influence of the cross-derivatives between residuals and update variables corresponding to different subsystems:

\[ \frac{\partial \tilde{R}_s}{\partial P_{w \neq s}} = 0 \Rightarrow \sum_{w \neq s} \frac{\partial \tilde{R}_s}{\partial P_w}(P^n) \Delta P_w^n = 0 \quad (5.9) \]

Following a standard Newton procedure, the following \( N_s \) linear systems, one per each equation subsystem, must be solved, starting with \( P^0 = P^n \):

\[
\begin{align*}
\left[ \frac{\partial \tilde{R}_s}{\partial P_s} (P^k) \right] \Delta P_s^k &= -\tilde{R}_s(P^k) \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad P_s^{k+1} &= P_s^k + \Delta P_s^k
\end{align*}
\quad (5.10)
\]

In a steady case, where time accuracy is not required, the solution at the new time step \( P_s^{n+1} \) can be directly set equal to \( P_s^{k+1} \) after one Newton sub-iteration. In an unsteady case, we must keep on iterating till \( \| \Delta P_s^k \| < \varepsilon \) and then set

\[ P_s^{n+1} = P_s^{k_{\text{last}}+1}. \quad (5.11) \]

before advancing to the next time level.

Herein, two iterative strategies have been implemented:

1. all linear systems \((s = [1, N_s])\) are first assembled and then solved within the same Newton loop; at the end of the whole iterative procedure, a single solution update is applied to all the variables in \( P \);

2. one linear system at a time is assembled and solved, so that \( N_s \) distinct Newton loops (each one with potentially different number of Newton iterations) and corresponding solution updates are performed.

In our applications involving weak coupling, such as turbulent and Inductively Coupled Plasma (ICP) simulations in Chapter 7, only the first scheme has been applied.

Even though the present thesis from now on will focus solely on steady applications, for which the linear Backward Euler scheme (with one Newton step) in Eq. 5.5 has been employed, the algorithm here described has also
been successfully applied to unsteady problems in [172] and especially in [169], in combination with the second order time accurate 3B scheme in Eq. 5.7. Moreover, the scheme here presented offers flexibility in the choice of the update variables while retaining a conservative form. This is extremely useful in case of flows characterized by complex thermodynamic relations, like the ones presented in this thesis, where one can conveniently choose $\mathbf{P}$ such that the functions $\mathbf{U}_s(\mathbf{P})$ are explicit.

### 5.1.2 Jacobian computation

A crucial ingredient for the overall performance of an implicit solver is the calculation of the flux jacobian terms to be included in the matrix of the linear system to solve:

$$
\left[ \frac{\partial \overline{R}_s}{\partial \mathbf{P}_s}(\mathbf{P}^k) \right] = J_t + J_R
$$

(5.12)

Two contributions to the jacobian matrix can be identified, namely $J_t$ corresponding to the time-dependent part of the equation and $J_R$ related to the discretization of the spatial part (convective and diffusive fluxes, source term). The expression of $J_t$ and $J_R$ depend on the choice of the time stepping method:

$$
J_t^{BE} = \Omega \frac{\partial \mathbf{U}_s}{\partial \mathbf{P}_s}(\mathbf{P}^k), \quad J_t^{CN} = \frac{\Omega}{\Delta t} \frac{\partial \mathbf{U}_s}{\partial \mathbf{P}_s}(\mathbf{P}^k), \quad J_t^{3B} = \frac{3 \Omega}{2 \Delta t} \frac{\partial \mathbf{U}_s}{\partial \mathbf{P}_s}(\mathbf{P}^k),
$$

$$
J_R^{BE} = \frac{\partial \overline{R}_s}{\partial \mathbf{P}_s}(\mathbf{P}^k), \quad J_R^{CN} = \frac{1}{2} \frac{\partial \overline{R}_s}{\partial \mathbf{P}_s}(\mathbf{P}^k), \quad J_R^{3B} = \frac{\partial \overline{R}_s}{\partial \mathbf{P}_s}(\mathbf{P}^k)
$$

(5.13), (5.14), (5.15)

The jacobian of the space terms $J_R$ typically incorporates different contributions, depending on the chosen space discretization method, as explained more in detail in Sec. 5.2.6.1 and Sec. 5.3.5.1.

### 5.1.3 Linear system solver

Each linear system arising from a Newton linearization in Eq. 5.10 is solved in COOLFluID by a dedicated third party library such as PETSc [73], Trilinos [72] or SAMG [4] which have been encapsulated behind a common
interface, as shown in Sec.2.4.1.2 and are treated as black-boxes by the CFD solver.
A non negligible effort was needed in order to make these libraries work in parallel inside COOLFluiD in a transparent way for the numerical solvers. This work mainly consisted in defining some extra LSS Command objects that encapsulate a few calls to the MPI based routines of the original libraries. Moreover, the storage of some extra connectivity/indexing information was needed in order to correctly insert the jacobian contributions to the system matrices (one per equation subsystem in weakly coupled cases) only coming from the local updatable States in each processor, without duplicating the entries inside the overlap region. In the case of different weakly coupled systems, our flexible implementation allows the user to associate a different linear system solver package to each one of the systems, according to his/her own needs.
Even though other options were available, all the results presented in this thesis have made use of the Generalized Minimal RESidual (GMRES) method [143] in combination with the restricted additive Schwarz preconditioner provided by PETSc. GMRES is an iterative algorithm that approximates the solution of the given linear system by the vector in a Krylov subspace with minimal residual.
5.2 Cell-Centered Finite Volume

During the last three decades, the Finite Volume method [16, 58, 82, 86, 152] has consolidated itself as the de facto standard technique for simulating a variety of flows in regimes ranging from incompressible to hypersonic. The success of the method is mainly due to the capability to adapt to every kind of meshes (structured, unstructured, polyhedral, cartesian, non-conformal) and to the good shock capturing properties, which make it suitable for handling compressible flows exhibiting complex shock interactions and discontinuities in general. Moreover, the algorithm is relatively easy to implement and efficiently parallelizable.

The basic idea of the technique consists in subdividing the computational domain in finite cells or volumes and, in a first order approximation, in assuming the solution to be constant inside each cell and stored in the corresponding centroids, as indicated in Fig. 5.1.

![Figure 5.1: Example of a computational domain on which a cell centered Finite Volumes discretization is applied.](image)

The algorithm discretizes the system of PDE’s in Eq. 4.1 written in integral form as in Eq. 5.1:

\[
\frac{d}{dt} \int_{\Omega} U \, d\Omega + \int_{\Sigma} F^c \cdot n \, d\Sigma = \int_{\Sigma} F^d \cdot n \, d\Sigma + \int_{\Omega} S \, d\Omega \quad (5.16)
\]
where $\Omega$ is the volume of each single cell in the computational domain and the Gauss’ theorem has been applied to convert the volume integrals of the divergence of the fluxes into contour integrals of the fluxes on the boundary of each volume. At this point, every single term in the equation can be discretize separately, typically leading to an upwind or central treatment of the convective fluxes, a central discretization of the diffusive fluxes, a cell-centered treatment of the source term in combination with an explicit (forward Euler, Runge Kutta) or implicit scheme (like the one presented in Sec. 5.1) for the (pseudo-)time stepping. Moreover, second order accuracy in space is obtained by evaluating the fluxes with a linear polynomial reconstruction of the solution, while high order in time can be easily achieved by selecting a high order scheme for the time integration like a n-th order Runge Kutta order (explicit) or multi-point backward formulas (implicit).

### 5.2.1 Discretization of Convective Fluxes

If we discretize the convective term, we obtain:

$$\int_{\Sigma} F^c \cdot n \, d\Sigma = \sum_{f=1}^{N_f} F_f \Sigma_f$$

(5.17)

where $F_f = F^c_f \cdot n_f$ is the numerical convective flux projected onto the normal $n_f$ to the interface $f$ with area $\Sigma_f$. In a first order cell centered Finite Volume method, the numerical flux $F$ depends on the state vectors corresponding to the left and right neighboring cell centers of the considered face:

$$F = F(U^L, U^R)$$

(5.18)

This is due to the fact that a piecewise constant representation of the solution holds within each control volume in which the domain is subdivided, meaning that neighboring states are in general not equal. This leads to the definition of a nonlinear Riemann problem on the interfaces, as pointed out by Godunov [55], which can be solved with the methods described in [58, 152]. Suitable numerical schemes must be able to resolve monotonically the discontinuities (shocks, contact surfaces) appearing in compressible flows, without yielding overshoots or undershoots in the solution. Moreover, schemes should not violate the entropy condition, i.e. the numerical counterpart of the second principle of thermodynamics, which states that no expansion shocks are admissible as weak solutions of the conservation law.
in Eq. 5.16. The effort towards the construction of satisfactory schemes is still ongoing, but many different successful formulations already exist. In particular, three main classes of schemes can be identified: Flux Difference Splitting (FDS), Flux Vector Splitting (FVS) and hybrid schemes. FDS [111, 114, 142] are known to be the most accurate especially for resolving viscous flows, because of their ability to correctly resolve contact discontinuities, while this corresponds to the major deficiency of FVS [83, 148], the advantage of the latter lying in their superior robustness in capturing strong shocks. Hybrid schemes such as HUS [13, 31] or AUSM [87–90] try to combine the best properties of FDS and FVS, while providing, especially in the case of AUSM, a superior computational efficiency and robustness.

In this section, we describe only a few of the schemes we implemented in order to deal with high-enthalpy and chemically reactive flows and for which some results will be presented in Chapter 6.

### 5.2.1.1 Roe

Roe’s flux splitting scheme [142] is based on an exact solution of the linearized Riemann problem across an interface and provides the most accurate results among the finite volume based discretization techniques for convective fluxes. In particular it yields a precise resolution for contact discontinuities and shear layers, which makes it highly suitable for dealing with viscous flows at high Reynolds number, since boundary layers are recognized as contact discontinuities by the Riemann solver. In its general formulation it reads:

$$F^{Roe} = \frac{1}{2} [F_n(P^L) + F_n(P^R)] - \frac{1}{2} |A(\bar{Z})| [U(P^R) - U(P^L)]$$  \hspace{1cm} (5.19)

with

$$A = \frac{\partial F_n}{\partial U}, \quad |A| = R |A| R^{-1}, \quad |A|_{ij} = |\lambda| \delta_{ij}$$  \hspace{1cm} (5.20)

where $R$ is the matrix of right eigenvectors, $A$ the diagonal matrix of eigenvalues, $U$ are the conservative variables and $P$ are the variables in which the solution is updated (primitive or conservative in our case). A key ingredient of the method is the linearization of the flux jacobian matrix $|A|$, projected onto the direction of the face normal $n$. An exact linearization is possible for relatively simple cases (e.g. Euler equations for perfect non-reactive gas) through the definition of a unique averaged parameter vector
Z. However, for the system of equations describing a mixture of perfect gases in thermo-chemical non equilibrium, which is what our work focuses on, such a linearization is not directly available. One of the properties that the matrix $\mathbf{A}$ has to satisfy is:

$$\Delta(\mathbf{F}_n) = \mathbf{A}\Delta(\mathbf{U}) \quad (5.21)$$

where $\Delta(.) = (.)_R - (.)_L$. Eq.5.21 is rigorously valid for all the flux components which are homogeneous quadratic functions in the components of $Z$, except for the pressure flux. In particular, in the case of a neutral gas mixture with $N_s$ chemical species, the momentum components of Eq. 5.21 are satisfied if the following condition holds:

$$\Delta(p) = \sum_{s=1}^{N_s} \bar{\alpha}_s \Delta(\rho_s) + \bar{\beta} \Delta(\rho_{\text{tr}}) \quad (5.22)$$

where

$$\beta = \frac{\partial p}{\partial \rho E}, \quad \alpha_s = RT/M_s - \beta e_{\text{tr},s}, \quad e_{\text{tr},s} = (f_s RT + \Delta h_{f,s})/M_s \quad (5.23)$$

with $f_s = 2.5$ for molecules and $f_s = 1.5$ for atoms. Different methods have been proposed in order to satisfy Eq.5.22, like the approximate and relatively complex solution in [91], the recipe based on computational experience in [53], the choice of a simple average of left and right states in conservative variables as in [144].

For our implementation, we have chosen the simple and accurate approach presented in [127], which requires that $\bar{\alpha}_s$ and $\bar{\beta}$ form a consistent set of thermodynamic variables together with the other Roe-average variables. If we define $a = \sqrt{\rho_L}/(\sqrt{\rho_L} + \sqrt{\rho_R})$ and $b = (1 - a)$, the following relations hold:

$$\bar{y}_s = a(y_s)_L + b(y_R)_R, \quad s = 1, ..., N_s \quad (5.24)$$

$$\bar{u} = au_L + bu_R \quad (5.25)$$

$$\bar{H} = aH_L + bH_R \quad (5.26)$$

$$\bar{e}_V = a(e_V)_L + b(e_V)_R \quad (5.27)$$

Moreover, if we choose the following definitions for $\bar{\beta}, \bar{\alpha}_s, \bar{e}_{\text{tr},s}$ and $a^2$:
\[ \bar{\beta} = \left[ \sum_{s=1}^{N_s} \bar{y}_s/M_s \right] / \left[ \sum_{s=1}^{N_s} f_s \bar{y}_s/M_s \right] \]  

(5.28)

\[ \bar{\alpha}_s = R \bar{T}/M_s - \beta \bar{e}_{tr,s} \]  

(5.29)

\[ \bar{e}_{tr,s} = (f_s RT + \Delta h_{f,s})/M_s \]  

(5.30)

\[ \bar{a}^2 = \sum_{s=1}^{N_s} \bar{\alpha}_s \bar{y}_s + \bar{\beta}(\bar{H} - \bar{q} - \bar{e}_V) \]  

(5.31)

[127] shows that we get some remarkably simple results like

\[ \bar{T} = aT_L + bT_R \]  

(5.32)

\[ \bar{e}_{tr} = a(e_{tr})_L + b(e_{tr})_R \]  

(5.33)

\[ \bar{p}/\bar{\rho} = a(p/\rho)_L + b(p/\rho)_R \]  

(5.34)

and this allow us to perform a consistent linearization for the rotated jacobian matrix \( |A| \).

### 5.2.1.2 Carbuncle fix

All Riemann solvers, including Roe’s, are affected by the so-called carbuncle phenomenon. The latter consists in a numerical instability which develops from the stagnation stream line in flows characterized by strong bow shocks (typically at Mach higher that 6) and moves outward into the flow field, causing a severe deterioration of the solution. Several fixes have been developed so far to cure the carbuncle specifically for the Roe scheme. Quirk [133] proposes a hybrid scheme, where a pressure based shock detector allows to identify regions in vicinity of shocks, where Roe’s scheme is replaced by a more dissipative one. Sanders et al. [144] identify the multidimensional nature of the phenomenon and present a parameter-free upwind dissipation modification, best known as \( H \)-correction, whose stencil is shown in Fig. 5.2. They define the maximum variation of the eigenvalues across a cell interface \( f \) with \( n_f \) as unit normal:

\[ \eta_f = \frac{1}{2} \max_{l \in \mathcal{L}} \left( |\lambda_l(U^R, n_f) - \lambda_l(U^L, n_f)| \right) \]  

(5.35)
Figure 5.2: Cell interfaces defining the stencil for the H-correction for (a) structured uniform mesh and (b) unstructured triangular mesh (from [123]).

and then they determine a multidimensional correction by selecting the maximum \( \eta_f \) within a H-shaped stencil, centered on the current face:

\[
\eta^{SA} = \max(\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)
\]  

Finally, \( \eta^{SA} \) is injected into one of the following entropy corrections in order to calculate \( |\tilde{\lambda}_t| \):

\[
|\tilde{\lambda}_t| = |\lambda_t(\bar{Z})| + \eta
\]  

\[
|\tilde{\lambda}_t| = \max(|\lambda_t(\bar{Z})|, \eta)
\]  

\[
|\tilde{\lambda}_t| = \begin{cases} |\lambda_t(\bar{Z})| & \text{if } |\lambda_t(\bar{Z})| \geq 2\eta \\ |\lambda_t(\bar{Z})|^2 / 4\eta + \eta & \text{otherwise} \end{cases}
\]

A variant of this method is proposed by Pandolfi et al. in [115], where it is proposed to define

\[
\eta^{PA} = \max(\eta_2, \eta_3, \eta_4, \eta_5)
\]  

and to apply the fix only to the entropy and shear waves, in order to avoid to introduce unnecessary artificial viscosity. Another fix is designed in [123] especially tailored for truly unstructured meshes. It consists in applying the modified multidimensional dissipation by Pandolfi to the entropy and shear waves (es) and the Van Leer entropy correction [84] to the acoustic ones (ac):

\[
|\tilde{\lambda}_t| = \begin{cases} |\lambda_{es}(\bar{Z})| & \text{if } |\lambda_{es}(\bar{Z})| \geq 2\eta^{VL} \\ |\lambda_{es}(\bar{Z})|^2 / 4\eta^{VL} + \eta^{VL} & \text{if } |\lambda_{es}(\bar{Z})| < 2\eta^{VL} \\ \max(|\lambda_{es}(\bar{Z})|, \eta^{PA}) & \text{otherwise} \end{cases}
\]
where $\eta^{VL} = \max(\lambda_R - \lambda_L, 0)$.

### 5.2.1.3 Analytical Flux Jacobian

The Roe scheme is computationally more expensive than most of the other flux schemes, especially because of its matrix dissipation term. However, when used in combination with implicit time stepping, its flux jacobian can be easily computed analytically and this results in a quite efficient algorithm. If we consider frozen the matrix $A$ during the derivation as in [42], the Jacobian terms with respect to the left and right state are respectively:

\[
\frac{\partial F_{Roe}}{\partial P_L} = \frac{1}{2} (A(P_L) + |A(\tilde{Z})|) \frac{\partial U}{\partial P}(P_L) \quad (5.42)
\]

\[
\frac{\partial F_{Roe}}{\partial P_R} = \frac{1}{2} (A(P_R) - |A(\tilde{Z})|) \frac{\partial U}{\partial P}(P_R) \quad (5.43)
\]

where the linearized rotated matrix dissipation is kept constant during the derivation.

### 5.2.1.4 Modified Steger-Warming

The original Steger-Warming scheme [148] belongs to the flux vector splitting family and relies on the fact that the inviscid flux vector $F$ is homogeneous in the vector of conservative variables $U$. This implies that

\[ F = \frac{\partial F}{\partial U} U = AU \quad (5.44) \]

If we diagonalize the Jacobian $A$ such that

\[ A = RAR^{-1} \quad (5.45) \]

where $R$ is the matrix of right eigenvectors, the fluxes can be split into forward and backward moving components, according to the sign of the eigenvalues:

\[ F = F_+ + F_- \quad (5.46) \]

with

\[ F_\pm = (RA_\pm R^{-1}) U = A_\pm U \quad (5.47) \]
\[ \Lambda_+ = \frac{\Lambda_+ + |\Lambda|}{2}, \quad \Lambda_- = \frac{\Lambda_- + |\Lambda|}{2}, \]  
\hspace{1cm} (5.48) 

where \( \Lambda_{\pm} \) are the diagonal matrices of the positive and negative eigenvalues. A straightforward discretization of the convective fluxes across a finite volume face with normal \( \mathbf{n} \) leads to

\[ \mathbf{F}^{SW}_{+,L} \cdot \mathbf{n} = \mathbf{A}_+(\mathbf{U}_L, \mathbf{n})\mathbf{U}_L, \quad \mathbf{F}^{SW}_{+,R} \cdot \mathbf{n} = \mathbf{A}_-(\mathbf{U}_R, \mathbf{n})\mathbf{U}_R \]  
\hspace{1cm} (5.49) 

It is well known that this formulation offers a good robustness in capturing strong shocks and rarefaction waves but it lacks accuracy in resolving boundary and shear layers. The performance of the scheme can be considerably enhanced \[42, 105\] by evaluating the split Jacobian matrices in states \( \hat{\mathbf{P}}_{\pm} \), corresponding to the following weighted averages of the primitive variables \( \mathbf{P} \):

\[ \hat{\mathbf{P}}_+ = (1 - \omega)\mathbf{P}_L + \omega \mathbf{P}_R, \quad \hat{\mathbf{P}}_- = \omega \mathbf{P}_L + (1 - \omega)\mathbf{P}_R \]  
\hspace{1cm} (5.50) 

with pressure-related weight \( \omega \) defined as

\[ \omega = \frac{1}{2} [1/(|\nabla \bar{p}|^2 + 1)], \quad \nabla \bar{p} = \sigma_2(p_R - p_L)/\min(p_R, p_L) \]  
\hspace{1cm} (5.51) 

As suggested in \[42\], we set \( \sigma_2 = 0.5 \). The resulting blended scheme reverts to the original Steger-Warming formulation in the vicinity of strong shocks, while it smoothly switches to a more accurate form, having \( \mathbf{A}_+ \) and \( \mathbf{A}_- \) both evaluated in the same averaged data, in regions of small pressure gradients, as in the case of contact discontinuities, shear layers and boundary layers.

### 5.2.1.5 AUSM Family

The schemes proposed in \[88–90\] combine the simplicity and efficiency of the flux vector splitting method with the high level of accuracy typical of Godunov type schemes. They rely on the splitting of the flux \( \mathbf{F} \) into a convective \( \mathbf{F}^{(c)} \) and a pressure component \( \mathbf{F}^{(p)} \). Therefore, at a continuum level we have:

\[ \mathbf{F}(\mathbf{U}) = \mathbf{F}^{(c)} + \mathbf{F}^{(p)} = \dot{m}\mathbf{\Psi} + \mathbf{F}^{(p)} \]  
\hspace{1cm} (5.52) 

The numerical discretization of the component of flux \( \mathbf{F} \) normal to a given interface between left (\( L \)) and right (\( R \)) states can consequently be defined as:
\[
\mathbf{F}_{1/2}(\mathbf{U}_L, \mathbf{U}_R, \mathbf{n}) = \dot{m}_{1/2} \Psi_{L/R} + \mathbf{p}_{1/2}
\]  
(5.53)

where different choices are possible for the quantities \( \dot{m}_{1/2} = \dot{m}_{1/2}(\mathbf{U}_L, \mathbf{U}_R, \mathbf{n}) \) and \( \mathbf{p}_{1/2} = \mathbf{p}_{1/2}(\mathbf{U}_L, \mathbf{U}_R, \mathbf{n}) \) which determine the splitting of convective and pressure components. Herein, in the case of a gas mixture in thermal and chemical nonequilibrium with \( N_s \) chemical components and \( N_m \) molecules, i.e. one of the most comprehensive cases presented in this thesis, the scalar mass flux \( \dot{m} \), the vector quantity \( \Psi \) and the pressure flux \( \mathbf{F}^{(p)} \) can be expressed as:

\[
\dot{m} = \rho q_n, \quad \Psi = \begin{pmatrix} y_s \\ \mathbf{u} \\ H \\ y_m E_m \end{pmatrix}, \quad \mathbf{F}^{(p)} = \begin{pmatrix} 0 \\ \rho n \\ 0 \\ 0 \end{pmatrix}
\]  
(5.54)

where \( q_n = \mathbf{u} \cdot \mathbf{n} \) is the velocity projected onto the normal \( \mathbf{n} \) to the considered cell interface.

**Liou-Steffen’s AUSM.** In the original AUSM formulation in [90], the following definitions apply:

\[
\dot{m}_{1/2} = M_{1/2} \begin{cases} a_L \rho_L & \text{if } M_{1/2} > 1, \\ a_R \rho_R & \text{otherwise} \end{cases}
\]  
(5.55)

\[
M_{1/2} = \mathcal{M}^+(M_L) + \mathcal{M}^-(M_R)
\]  
(5.56)

\[
\mathbf{p}_{1/2} = \mathcal{P}^+(M_L)\mathbf{n} + \mathcal{P}^-(M_R)\mathbf{n}
\]  
(5.57)

where the split Mach number polynomials \( \mathcal{M}^\pm \) read

\[
\mathcal{M}^\pm(M) = \begin{cases} \mathcal{M}^\pm_{(1)}(M), & \text{if } |M| > 1, \\ \mathcal{M}^\pm_{(2)}(M), & \text{otherwise} \end{cases}
\]  
(5.58)

with

\[
\mathcal{M}^\pm_{(1)}(M) = \frac{1}{2}(|M| + M)
\]  
(5.59)

\[
\mathcal{M}^\pm_{(2)}(M) = \pm \frac{1}{4}(|M| + 1)^2
\]  
(5.60)

and the split pressure polynomials \( \mathcal{P}^\pm \) are given by
\( \mathcal{P}^\pm (M) = \begin{cases} \frac{1}{M} \mathcal{M}^\pm_1, & \text{if } |M| > 1, \\ \pm \mathcal{M}^\pm_2(2 \mp M), & \text{otherwise} \end{cases} \) \hspace{1cm} (5.61)

with the normal Mach number \( M_i = \frac{q_{ni}}{a_c} \) and the normal speed \( q_{ni} = \mathbf{u}_i \cdot \mathbf{n} \).

Then simple upwinding is applied to define \( \Psi_{1/2} : \)

\[ \Psi_{1/2} = \begin{cases} \Psi_L & \text{if } M_{1/2} \geq 0, \\ \Psi_R & \text{otherwise} \end{cases} \] \hspace{1cm} (5.62)

**AUSM**. An improved version of the previous scheme is presented in [89].

The split Mach numbers are redefined as follows:

\[ \mathcal{M}^\pm_4(M) = \begin{cases} \mathcal{M}^\pm_1, & \text{if } |M| \geq 1, \\ \mathcal{M}^\pm_2(1 \mp 16 \beta M_\pm) & \text{otherwise} \end{cases} \] \hspace{1cm} (5.63)

where \(- \frac{1}{16} \leq \beta \leq \frac{1}{2}\).

The split pressure polynomials \( \mathcal{P}^\pm \) are also modified:

\[ \mathcal{P}^\pm_5(M) = \begin{cases} \frac{1}{M} \mathcal{M}^\pm_1, & \text{if } |M| \geq 1, \\ \mathcal{M}^\pm_2[(\pm 2 - M) \mp 16 \alpha M] & \text{otherwise} \end{cases} \] \hspace{1cm} (5.64)

with \( \frac{3}{16} \leq \alpha \leq \frac{1}{8} \). The mass flux at the interface \( \dot{m}_{1/2} \) is based on an average sound speed \( a_{1/2} \) between the left and right states:

\[ \dot{m}_{1/2} = M_{1/2} a_{1/2} \begin{cases} \rho_L & \text{if } M_{1/2} > 1, \\ \rho_R & \text{otherwise} \end{cases} \] \hspace{1cm} (5.65)

Several choices are possible for expressing \( a_{1/2} = a(\mathbf{U}_L, \mathbf{U}_R) : \)

\[ a_{1/2} = \sqrt{a_L a_R} \] \hspace{1cm} (5.66)

\[ a_{1/2} = \frac{a_L + a_R}{2} \] \hspace{1cm} (5.67)

\[ a_{1/2} = \min(\tilde{a}_L, \tilde{a}_R), \quad \tilde{a} = \frac{a^*}{\max(a^*, |q_{ni}|)} \] \hspace{1cm} (5.68)

[88, 89] recommend the last expression, based on the critical sound speed

\[ a^* = \sqrt{\frac{2(\gamma - 1)}{\gamma + 1} H} \] \hspace{1cm} (5.69)
which yields the exact solution for a single stationary shock discontinuity. In our computations dealing with flows in thermo-chemical non-equilibrium, we adopt a more suitable definition for the critical sound speed, as inspired by [146]:

\[ a^* = \sqrt{\frac{2\tilde{\gamma}(\tilde{\gamma} - 1)}{2\tilde{\gamma} + \tilde{\gamma}(\tilde{\gamma} - 1)}} H \]  

where the frozen specific heat ratio \( \tilde{\gamma} = \frac{\sum y_i \partial h_s / \partial T}{\sum y_i \partial e_s / \partial T} \) and the equivalent specific heat ratio \( \bar{\gamma} = 1 + p/\rho e \) are considered.

**AUSM\(^+-\)up.** A further extension of the AUSM\(^+\) flux to accurately simulate flows from low speed to hypersonic regime is discussed in [87, 88]. According to the new formulation, a pressure diffusion term is integrated in Eq. 5.56:

\[ M_{1/2} = M_{1/2}^{AUSM^+} + K_p \max(1 - \sigma \bar{M}^2, 0) \frac{p_R - p_L}{\rho_{1/2}^2 a_{1/2}^2} \]  

where \( 0 \leq K_p \leq 1 \), \( \sigma \leq 1 \) and the averaged Mach number and interface density are given by

\[ \rho_{1/2} = \frac{\rho_L + \rho_R}{2}, \quad \bar{M} = \frac{q_{nL}^2 + q_{nR}^2}{2a_{1/2}^2}. \]  

The pressure flux 5.57 is also modified as follows:

\[ p_{1/2} = p_{1/2}^{AUSM^+} - K_u P_{(5)}^+ P_{(5)}^- (\rho_L + \rho_R)(f_a a_{1/2}^2)(q_{nR} - q_{nL}) \]  

with \( 0 \leq K_u \leq 1 \) and the scaling factor \( f_a \) given by

\[ f_a(M_0) = M_0(2 - M_0), \quad M_0^2 = \min(1, \max(\bar{M}^2, M_{co})) \]

where the cut-off Mach number \( M_{co} \) is user defined and \( O(M_\infty) \). Moreover, the parameters \( \alpha \) and \( \beta \) appearing in the split Mach and pressure functions are set to

\[ \alpha = \frac{3}{16} (-4 + 5 f_a^2) \in \left[ -\frac{3}{4}, \frac{3}{16} \right], \quad \beta = \frac{1}{8} \]

As suggested in [87, 88], we set as default values \( K_p = 0.25, K_u = 0.75 \) and \( \sigma = 1.0 \).
5.2.2 High Order Reconstruction

The basic finite volume method assumes a constant average solution vector on each cell and this leads to a first order accurate discretization in space. As observed by Van Leer [82], higher accuracy can be reached by replacing the piecewise constant left and right states with a piecewise polynomial representation. In particular, in order to get second order accuracy, each one of the cell centered state variables \( u_i \) must be linearly extrapolated to the face quadrature points \( q \) as follows:

\[
\tilde{u}(x_q) = u_i + \nabla u_i \cdot (x_q - x_i) \quad (5.76)
\]

where \( x_i \) denotes the centroid position of the control volume \( \Omega_i \). The linearly reconstructed state variables \( \tilde{u} \) can be calculated with different methods, some of which are described hereafter. Moreover, when dealing with the simulation of compressible flows, a flux limiter must be employed on the reconstructed states, in order to prevent the appearance of oscillations near discontinuities.

Finally, the convective fluxes presented in Sec. 5.2.1 must be evaluated in function of the reconstructed (and limited, if needed) states \( \tilde{u} \).

5.2.2.1 Weighted MUSCL

The procedure of extrapolating the cell centered variables in order to generate high order upwind schemes is often indicated as MUSCL (Monotone Upstream-centered Schemes for Conservation Laws). The original method [82] is based on a one-dimensional reconstruction on both side of an interface, with a stencil involving a number of aligned cell centers and it is applicable only to regular structured meshes, where the required stencil can be easily built.

In this work, we have employed this reconstruction on structured meshes (with only quadrilaterals in 2D or only hexahedra in 3D) but introducing some weights in order to account for the irregularity of the mesh, due to the different size of neighboring cells. When applying a linear extrapolation of the variables \( \mathbf{U} \) on the left and on the right of an interface, the resulting formulas are:

\[
\tilde{U}_L = U_L + \omega_L(U_L - U_{LL}) \quad (5.77)
\]

\[
\tilde{U}_R = U_R - \omega_R(U_{RR} - U_R) \quad (5.78)
\]

where the weights \( \omega \) given by:
where \( \bar{x} \) is the position of the interface midpoint and \( x_i \) are the cell centroid positions. Moreover, on regular meshes we have \( \omega_i = \frac{1}{2} \), as in the original MUSCL extrapolation.

A special treatment is applied on the boundary faces. In traditional cell centered structured finite volume codes, two layers of ghost cells are employed. In our case, we only adopt one layer, which means that no \( \mathbf{U}_{RR} \) is available for the boundaries. In that case, assuming that no discontinuity lies on the boundary itself, we apply a simple unlimited weighted average variable extrapolation:

\[
\tilde{\mathbf{U}}_L = \tilde{\mathbf{U}}_R = \frac{\omega_L \mathbf{U}_L + \omega_R \mathbf{U}_R}{\omega_L + \omega_R} \tag{5.81}
\]

with \( \omega_i = \frac{1}{|x-x_i|} \). If the ghost state is located in a symmetric position with respect to the inner cell center and the boundary, those weights are equal to \( \frac{1}{2} \). However, in some cases (e.g. isothermal wall condition) we allow the ghost node to move and get closer to the boundary to keep reasonable values for some extrapolated variables (e.g. temperature > 0) and this gives different values for the weights.

### 5.2.2.2 Least Squares Technique

On a general polyhedral unstructured mesh, the cellwise gradient \( \nabla u \) can be computed with a least square (LS) approach as the result of the following linear system [15, 16]:

\[
\begin{bmatrix}
\mathbf{L}_x & \mathbf{L}_y & \mathbf{L}_z
\end{bmatrix}
\n\begin{bmatrix}
\nabla u_i
\end{bmatrix}
= \mathbf{f}_u \tag{5.82}
\]

The matrix on the LHS is generally non-square and its column vectors \( \mathbf{L}_d \) are defined as:

\[
\mathbf{L}_d = [w_1(\Delta x_d)_1, \ldots, w_{N_i}(\Delta x_d)_{N_i}]^T \tag{5.83}
\]

where the weights \( w_k \) multiply the distances between the centroid of the current cell and the centroids of its \( N_i \) neighbor cells, belonging to the chosen computational stencil. Linear weights can be based on the inverse of
distances and computed as \( w_j = 1/\| \Delta x_j \| \).

The non linear RHS vector \( f_u \) reads:

\[
 f_u = \left[ w_1 \Delta (u_1 - u_i), \ldots, w_N (u_{N_i} - u_i) \right]^T \quad (5.84)
\]

The system in Eq. 5.82 can be solved in a least squares sense with an orthogonalization technique, leading to

\[
 [L_x \ L_y \ L_z]^T \cdot [L_x \ L_y \ L_z] \cdot (\nabla u) \Omega_i = [L_x \ L_y \ L_z]^T \cdot f_u \quad (5.85)
\]

After having defined the dot products \( l_{jk} = L_j \cdot L_k \) and \( f_l = L_j \cdot f_u \), Eq. 5.85 simplifies to

\[
 \nabla u_i = \{ l_{jk} \}^{-1} f_l \quad (5.86)
\]

where

\[
 \{ l_{jk} \} = [L_x \ L_y \ L_z]^T \cdot [L_x \ L_y \ L_z], \quad f_l = [f_x, f_y, f_z]^T \quad (5.87)
\]

If we take into account the definition of the inverse for a 3x3 matrix, Eq. 5.86 can be develop further and gives

\[
 \nabla u_i = \frac{1}{\det(\{ l_{jk} \})} \begin{pmatrix} \det(M_{xx}^L) f_x + \det(M_{xy}^L) f_y + \det(M_{xz}^L) f_z \\ \det(M_{yx}^L) f_x + \det(M_{yy}^L) f_y + \det(M_{yz}^L) f_z \\ \det(M_{zx}^L) f_x + \det(M_{zy}^L) f_y + \det(M_{zz}^L) f_z \end{pmatrix} \quad (5.88)
\]

where \( M_{jk}^L \) is a minor of the matrix \( \{ l_{jk} \} \). The system 5.85 is not necessarily well posed and a sufficiently large stencil is needed to prevents singularities \( (\det(\{ l_{jk} \}) \approx 0) \).

**Reconstruction Stencil.** [37] and [85] show and analyze the importance that the chosen computational stencil for the least square extrapolation has on the accuracy and robustness of the solution. That’s the reason why, in our implementation, we offer great flexibility for the choice of the stencil, which is user-defined and assembled in a pre-processing step, independently from the core reconstruction algorithm.

The following possibilities are available:

1. **Face:** includes only the face neighbors for each cell;

2. **Face-Vertex:** includes all the distant-1 cell neighbors, i.e. all cells sharing at least one vertex with the current cell (ghost cells can be included or not), as shown in Fig. 5.3a;
3. **Face-BVertex**: includes only the face neighbors for internal faces and the face-vertex neighbors for the boundary faces, as illustrated in Fig. 5.3b;

4. **Face-Edge**: includes all cell neighbors sharing at least one edge with the current cell.

Options (1) and (3), the cheapest ones from a computational point of view, do not provide enough robustness and accuracy, especially in 2D. Options (2) and (4) yield the best results, with the latter to be preferred in 3D computations, because of the prohibitive large stencil associated to the first one, especially for meshes with tetrahedra.

### 5.2.3 Flux Limiters

In order to prevent the appearance of spurious oscillations near shock waves or contact surfaces, i.e. the so-called Gibbs phenomenon, when using a high order solution reconstruction, a flux limiter, hereafter indicated with $\Phi$, is typically applied. Therefore, when dealing with compressible flows, the linear reconstruction for a generic variable $u(x)$ in Eq. 5.76 must be modified as follows:

$$\tilde{u}(x_q) = u_i + \Phi_i \nabla u_i \cdot (x_q - x_i)$$  \hspace{1cm} (5.89)
Chapter 5. Numerical methods

with \( \Phi = [0, 1] \). This comes at a cost of an unavoidable accuracy deterioration in proximity of flow discontinuities and of a hampering of convergence, which can both vary greatly depending on the choice of the limiter function itself.

### 5.2.3.1 Venkatakrishnan’s Limiter

Venkatakrishnan [163] developed his multidimensional limiter in order to overcome some deficiencies of Barth-Jespersen’s limiter [17], namely the degradation of accuracy in nearly smooth flow regions and its poor convergence properties. The \( \min(r, y) \) function used in [17] is replaced by a differentiable function:

\[
\Phi(r) = \frac{r^2 + 2r}{r^2 + r + 2}
\]

and the limiter is given by

\[
\Phi_{i, n_q} = \begin{cases} 
\Phi \left( \frac{\Delta^+}{\Delta^-} \right) & \text{for } \Delta^- > 0 \\
\Phi \left( \frac{\Delta^+}{\Delta^-} \right) & \text{for } \Delta^- < 0 \\
1 & \text{for } \Delta^- = 0
\end{cases}
\]

where

\[
\Delta^- = \Delta T_{n_q} \nabla u_i, \quad \Delta^+_{\max} = \max_i - u_i, \quad \Delta^+_{\min} = \min_i - u_i
\]

in which

\[
\max_i = \max_{j=0, \ldots, N_i} (u_i, u_j), \quad \min_i = \min_{j=0, \ldots, N_i} (u_i, u_j)
\]

In order to maintain the accuracy in almost uniform regions where \( \Delta^+ \approx \Delta^- \approx 0 \), the flux function 5.90 is reformulated as follows:

\[
\Phi \left( \frac{\Delta^+}{\Delta^-} \right) = \frac{\Delta^+^2 + 2\Delta^+\Delta^- + \epsilon}{\Delta^+^2 + \Delta^+\Delta^- + 2\Delta^-^2 + \epsilon}
\]

where the \( \epsilon \) term has to be appropriately scaled to be negligible in non smooth flow regions and predominant where the flow is nearly uniform. To this end, \( \epsilon \) can be estimated as

\[
\epsilon = K U_0^2 \left( \frac{h}{L} \right)^3
\]
where \( U_0 \approx O(\| u \|) \) in the whole domain, \( h \) is a local characteristic length (e.g. the average distance between the centroid of the current cell and the neighbors ones), \( L \) is a local characteristic solution length in the smooth flow regions and \( K \approx O(1) \) is a user-defined constant.

### 5.2.3.2 MUSCL Limiters

Within the frame of the MUSCL reconstruction, several limiters are available in literature, e.g. [1, 152, 167]. For sake of completeness, we report hereafter the ones that have been implemented in COOLFluiD:

\[
\Phi(r) = \max(0, \min(1, r)) \quad \text{(MINMOD) (5.95)}
\]
\[
\Phi(r) = \frac{3}{2} \max(0, \min(\min(2r, (1 + 3r)/4), 4)) \quad \text{(SMART) (5.96)}
\]
\[
\Phi(r) = \max(0, \max(\min(2r, 1), \min(r, 2))) \quad \text{(SUPERBEE) (5.97)}
\]
\[
\Phi(r) = \max(0, \min(\min(2r, (1 + 2r)/3), 2)) \quad \text{(KOREN) (5.98)}
\]
\[
\Phi(r) = \max(0, \min(\min(2r, (1 + r)/2), 2)) \quad \text{(MC) (5.99)}
\]

\[
\Phi(r) = \frac{r(3r + 1)}{(r + 1)^2} \quad \text{(CHARM) (5.100)}
\]
\[
\Phi(r) = \frac{3}{2} \frac{r + |r|}{r + 2} \quad \text{(HCUS) (5.101)}
\]
\[
\Phi(r) = \frac{2(r + |r|)}{r + 3} \quad \text{(HQUICK) (5.102)}
\]
\[
\Phi(r) = \frac{r^2 + r}{r^2 + 1} \quad \text{(Van Albada 1) (5.103)}
\]
\[
\Phi(r) = \frac{2r}{r^2 + 1} \quad \text{(Van Albada 2) (5.104)}
\]
\[
\Phi(r) = \frac{r + |r|}{1 + r} \quad \text{(Van Leer) (5.105)}
\]
\[
\Phi(r) = \frac{3}{2} \frac{r^2 + r}{r^2 + r + 1} \quad \text{(OSPRE) (5.106)}
\]

Herein, \( r \) represents the ratio of successive gradients on the solution mesh. For example:

\[
ri = \frac{ui - u_{i-1}}{ui+1 - ui} \quad (5.107)
\]
New arbitrarily complex limiter functions can be easily provided as user-defined functions directly in the COOLFluiD input file, without having to actually implement them in the code.

5.2.3.3 Historical Modification

The choice of one limiter function or another can improve the convergence, but after a drop of a few orders (2-3) of magnitude the residual tends always to oscillate. In order to cure this behaviour, we apply the treatment introduced in [37], namely the so-called historical modification. It consists in choosing

\[ \Phi^n_i = \min(\Phi^{n-1}_i, \Phi^n_i) \]  

but only after a starting period sufficiently long to obtain a rough convergence of the solution, during which no special treatment is applied. In our experience, this technique works well for many situations, even in presence of very strong discontinuities, but not in cases where complex shock/boundary layer interactions occur, e.g. in hypersonic flows on double cones. In the latter case, the historical modification might help to converge, but unfortunately to a wrong solution. Another disadvantage of this technique is that it requires the non-negligible storage in memory of the limiter values \( \Phi^{n-1}_i \) computed during the previous iteration for each variable in each computational cell (in combination with the LS reconstruction) or face (if standard MUSCL is used): this cost turns out to be prohibitive for 3D multi-species reactive flows.

5.2.4 Discretization of Diffusive Fluxes

The discretization of the diffusive term leads to:

\[ \int_{\Sigma} \mathbf{F}^d \cdot \mathbf{n} d\Sigma = \sum_{f=1}^{N_f} \mathbf{G}_f \Sigma_f \]  

where \( \mathbf{G}_f = \mathbf{F}^d_f \cdot \mathbf{n}_f \) and the diffusive fluxes \( \mathbf{F}^d \) typically depend on the primitive variables \( \mathbf{P} \) and their gradients:

\[ \mathbf{F}^d = \mathbf{F}^d(\mathbf{P}, \nabla \mathbf{P}) \]
5.2.4.1 Gradient Calculation

The application of Green-Gauss’ theorem within a chosen control volume \( \Omega^v \) can be used to determine the above mentioned gradients:

\[
\nabla P = \frac{1}{\Omega^v} \int_{\Omega^v} \nabla P \, d\Omega^v = \frac{1}{\Omega^v} \oint_{\Sigma^v} P \, n \, d\Sigma^v
\]

(5.111)

A popular choice for \( \Omega^v \) on unstructured meshes is a diamond-shaped volume [38, 122] like the one in Fig. 5.4 which is built around the considered face and which includes all the face nodes, the left and right cell centers as vertices.

![Diamond control volume](image)

**Figure 5.4**: Diamond control volume for the calculation of the gradients for the diffusive fluxes.

The discretized version of Eq. 5.111 becomes:

\[
\nabla P = \frac{1}{\Omega^v} \sum_{f=1}^{N_f} \bar{P}_f n_f \Sigma^v, \quad \text{with} \quad \bar{P}_f = \frac{1}{N_{n_f}} \sum_{j=1}^{N_{n_f}} P^j_f
\]

(5.112)

where \( \bar{P}_f \) is a face-averaged value of \( P \) calculated from the values \( P^j_f \) in the vertices of the diamond volume, whose number of faces is \( N_f \). Moreover, \( N_{n_f} \) is the number of vertices in each face \( f \) of the control volume, i.e. \( N_{n_f} \) is equal to the space dimension of the problem (2 or 3).

5.2.4.2 Nodal Extrapolation

The values \( P^j_f \) in Eq. 5.112 include both cell centered values (i.e. the "true" degrees of freedom) and mesh vertex values. The latter, here called \( P_v \), are calculated starting from the first ones, \( P_c \), using a weighted averaged extrapolation:
\[ P_v = \frac{\sum_{c \in \zeta_v} P_c \omega_c}{\sum_c \omega_c} \]  
(5.113)

where \( \zeta_v \) is the set of cell centers surrounding the considered vertex and possible definitions for the weights \( \omega_c \) are

\[ \omega_c = \frac{1}{\| \Delta x_c \|} \]  
(distance based)  
(5.114)

\[ \omega_c = \frac{1}{\Omega_c} \]  
(volume based)  
(5.115)

\[ \omega_c = 1 + \lambda \cdot \Delta x_c \]  
(Holmes-Connell)  
(5.116)

with \( \Delta x_c = x_c - x_v \). In particular, in the last approach [38, 61, 122], the weighting coefficients \( \lambda = (\lambda_x, \lambda_y, \lambda_z) \) are calculated by assuming that expression 5.113 is exact for a linear evolution of \( P \). We can therefore write:

\[ P_c = P_v + \nabla P_v \cdot \Delta x_v \]  
(5.117)

If we rewrite Eq. 5.113 as

\[ \sum_{c \in \zeta_v} \omega_c (P_v - P_c) = 0 \]  
(5.118)

and, because of Eq. 5.117, we obtain

\[ \sum_{c \in \zeta_v} \omega_c (\nabla P_v \cdot \Delta x_v) \]  
(5.119)

The latter relation should remain valid for any arbitrary value of the gradients \( \nabla P_v \) if \( P_v \) is linear in \((x, y, z)\). By conveniently choosing three possible values \((1, 0, 0), (0, 1, 0), (0, 0, 1)\) for the gradients, we are left with an algebraic system of equations in the unknowns \((\lambda_x, \lambda_y, \lambda_z)\):

\[
\begin{align*}
\sum_c \omega_c \Delta x_c &= \sum_c \Delta x_c + \lambda_x \sum_c \Delta x_c^2 + \lambda_y \sum_c \Delta x_c \Delta y_c + \lambda_z \sum_c \Delta x_c \Delta z_c = 0 \\
\sum_c \omega_c \Delta y_c &= \sum_c \Delta y_c + \lambda_x \sum_c \Delta x_c \Delta y_c + \lambda_y \sum_c \Delta y_c^2 + \lambda_z \sum_c \Delta y_c \Delta z_c = 0 \\
\sum_c \omega_c \Delta z_c &= \sum_c \Delta z_c + \lambda_x \sum_c \Delta x_c \Delta z_c + \lambda_y \sum_c \Delta y_c \Delta z_c + \lambda_z \sum_c \Delta z_c^2 = 0
\end{align*}
\]

The system 5.2.4.2 simply requires the inversion of a 3x3 matrix for each mesh vertex. Since the weights \( \omega_c \) do not depend on the flow solution, the system can be solved only once in a pre-processing step, if the nodal
coordinates don’t change along the simulation (i.e. no mesh movement or adaptation is performed) as in our case. As far as memory requirements are concerned, the storage of three (two in 2D) $\lambda_i$ coefficients per mesh vertex is needed to make this reconstruction procedure computationally efficient. Even though all the strategies for computing the weighting coefficients described so far have been implemented, the inverse distance-based approach has been our preferred choice for our calculations, because of its superior robustness.

**Treatment of Corner Vertices.** The nodal extrapolation may become tricky for boundary vertices. In the case of corner vertices shared by different topological surfaces, for instance, the nodal extrapolation must impose only one boundary condition. In our case, since the application of boundary conditions is based on ghost states, symmetrically built with respect to the internal cell centers and the boundary surface, only contributions from ghost states corresponding to one chosen topological region must be included for the computation of the corner values $P_v$.

### 5.2.5 Discretization of Source Terms

The discretization of the source term appearing on the LHS of 5.16 is based on the cell centered value in a given cell $i$:

$$\int_{\Omega} S(P) \, d\Omega \approx S(P_i)\Omega_i = S_i\Omega_i \tag{5.120}$$

When the source term includes derivatives of some dependent variable $p$ (e.g. the stress term $\tau_{\theta\theta}$ appearing in the case of axisymmetric Navier-Stokes), these are calculated by applying the Green-Gauss theorem, similarly to what explained in Sec. 5.2.4.1:

$$\nabla p = \frac{1}{\Omega_i} \int_{\Omega_i} \nabla p \, d\Omega_i \equiv \frac{1}{\Omega_i} \oint_{\Sigma_i} p \, n \, d\Sigma_i \tag{5.121}$$

In this case, however, the chosen control volume coincides with the volume of the current cell, $\Omega_i$, while a diamond-shaped one was used for the computation of the diffusive fluxes. The discretized version of Eq. 5.121 is identical to Eq. 5.112.
5.2.6 Implicit scheme

5.2.6.1 Numerical Jacobian

The analytical jacobian of the space terms $J_R$ in Eq. 5.12 incorporates two distinct contributions: one during a loop over all faces (boundary and internal ones) in order to assemble the contribution from the convective and diffusive fluxes, another one during a global loop over cells to calculate also the source term contributions.

The jacobian term can be computed numerically, by approximating each partial derivative by means of a forward finite difference formula. In our case, the jacobian of the residual $\mathbf{R}$ with respect to the $k$ component of the update variables $\mathbf{P}$ can be expressed as follows [64]:

$$
\left( \frac{\partial \mathbf{R}}{\partial \mathbf{P}_m} \right)_k \cdot \mathbf{1}_k \approx \frac{\mathbf{R} \left( \mathbf{P}_m + \varepsilon_k \mathbf{1}_k \right) - \mathbf{R} \left( \mathbf{P}_m \right)}{\varepsilon_k}.
$$

(5.122)

where $k = 1 \ldots N_{eq}$ and the index $m$ depends on the degree of freedom in the computational stencil whose corresponding $\mathbf{P}$ variables are perturbed.

When looping over the faces to assemble the fluxes, the following choice for $m$ holds:

$$
m = \{ \text{left, right} \quad \text{on internal faces} \\
\text{left} \quad \text{on boundary faces} \}
$$

(5.123)

This means that on the boundary face, only the (left) internal state is perturbed, while the (right) ghost state is recomputed to satisfy the numerical boundary condition for the given perturbed internal state, allowing us not to include entries for the ghost states in the jacobian matrix. The same technique has been used in [40]. In the case of the source term jacobian, however, since the discretization is based only on the cell centered state, only that one is perturbed and the contribution from the neighbor states is completely neglected.

The perturbation $\varepsilon_k$ in 5.122 is defined as [13, 64]:

$$
\varepsilon_k = \delta \ \text{sign} \left( P_{m,k} \right) \max \left( \left| P_{m,k} \right|, u_{k}^{user} \right),
$$

(5.124)

where the sign function is given by

$$
\text{sign} \left( x \right) = \left\{ \begin{array}{ll}
-1 & \text{if } x < 0 \\
1 & \text{if } x \geq 0
\end{array} \right.
$$

(5.125)
and where $\delta \leq 10^{-5}$ and $u_k^{\text{user}}$ is a user defined value whose magnitude should be of the order of the corresponding update variable $P_k$.

5.2.6.2 Treatment of Source Terms

The choice of the given cell as control volume leads to an efficient implicit discretization of the source term itself, which does not depend on neighbor states. If we apply a Taylor expansion in time to the source term, we get

$$S_i (P_i^{n+1}) = S_i (P_i^n) + \frac{\partial S_i}{\partial P_i} (P_i^n) \Delta P_i^n$$

(5.126)

where the cross derivatives $\frac{\partial S_i}{\partial P_j}$ are identically equal to zero, meaning that the source term Jacobian only contributes to the block diagonal of the global matrix and can be easily calculated on a separate loop over cells. If derivatives are present in the source terms, we keep frozen the extrapolated state values at the cell vertices during the Jacobian calculation, so that Eq. 5.126 still holds.

5.2.6.3 Performance Considerations

In order to improve the performance of the solver, the influence of the left-/right state perturbations on the nodal extrapolation (Sec. 5.2.4.2) is neglected. In other words, the extrapolation from cell centers to vertices is performed only once at the beginning of each time step and kept frozen during the calculation of the numerical Jacobian. In fact, a certain influence may be expected on the face-based gradient computation for the diffusive fluxes with the diamond volume approach described in Sec. 5.2.4.1, which relies on the face-vertex states and the left and right cell centers. However, numerical experiments didn’t fully support this conjecture, since in most cases the convergence histories didn’t seem to be affected at all by the inclusion of the perturbation effect on the nodal extrapolation. If we analyze the impact on the performance, the inclusion of this effect increases the global computational time by 20% in 2D and 50% in 3D, as detected while profiling the code.

Another simplification is made by assuming that the perturbation of the left-/right states of a face don’t affect other states involved in the stencil for the high order reconstruction, the same approach adopted in [13]. This is theoretically justified by the relative small influence that each state has on the gradient computation in the least squares reconstruction (Sec. 5.2.2.2) and
in the multidimensional limiting procedure (Sec. 5.2.3) based on a full stencil, i.e. involving all the vertex neighbors of the cell, especially in 3D. This assumption is very important because it allows to keep also in second-order the sparsity pattern corresponding to the first-order scheme and reduces the amount of computation within the overlap region in a parallel calculation. Moreover, in a parallel simulation, only contributions to the rows of the global matrix corresponding to locally updatable states are considered, while contributions to rows corresponding to states belonging to the overlap region but locally not updatable are not computed.

5.2.7 Boundary conditions

All boundary conditions are imposed weakly by means of the well known ghost cell (state in our case) approach [58, 86]. This allows the numerical algorithm to treat uniformly both boundary and internal faces, the only minor difference being in the computation of the residual jacobian term as just described in Sec. 5.2.6.1. While a conventional implementation of inlets, outlets and mirror conditions is provided, an innovative algorithm has been developed for handling no-slip walls and is therefore described here after.

5.2.7.1 No-slip wall

This boundary condition [117] is implemented following a ghost node approach, where an ad-hoc treatment is employed in order to keep the temperatures always positive during the transient at the beginning of the computation, while rigorously preserving a consistent gradient calculation. To this end, whenever one (or more) of the temperatures (roto-translational, vibrational) calculated in the ghost node as \( T_g^j = 2 \cdot T_w - T_{in}^j \) becomes smaller than a user-specified positive value \( T_u \), the ghost node, originally located in a position \( \mathbf{X}_{g0} \), symmetric to the internal cell center \( \mathbf{X}_{in} \) with respect to the wall boundary, is repositioned closer to the wall with a recursive dichotomycal algorithm until all the temperatures are at the same time bigger than \( T_u \).

In pseudo C++ code this leads to the algorithm in C.L. 5.1, where \( T_w \) is the wall temperature, and \( T_j \), the array of temperatures in the ghost node, is computed from a unidimensional linear extrapolation using the same slope \( \frac{\Delta T}{\Delta x} \) between the internal (cell-center) and the wall values. The constant \( K \) must be set bigger than 1.0, and it is usually set to 2.0, meaning that, during each repositioning, the ghost node is moved half way between the old position and the wall. As a consequence of the repositioning, velocity
components must be linearly interpolated in a similar way to vanish at the wall, while extrapolated components like pressure remain unchanged in the ghost node, even if it is relocated.

```
1  Xg = Xg_0;
2  f = 1;
3  while (!(T_j > T_u)) {
4      f *= K;
5      Xg_1 = ((f - 1.)*X_w + Xg)/f;  // new ghost node position
6          T_j = T_in - (T_in - T_w)/distance(X_in, X_w)*
7              distance(X_in, Xg_1);
8  }
9  Xg = Xg_1;  // update ghost position
```

**Code Listing 5.1:** Repositioning of the ghost node in no-slip wall condition

The simple but original (to the author’s knowledge), boundary treatment here proposed is fully consistent with the cell-centered space discretization and, in our experience, contributes to enhance the robustness of the code during the initial transient phase of the simulation after having started from an initial uniform flow field. To this end, care must be taken so that the solution extrapolation from the cell centers to the cell vertices, which is needed to compute the gradients of primitive variables for the diffusive fluxes, consistently accounts for the possibility of the ghost node movement. In the case of fixed temperature or velocity components, this can be achieved by strongly imposing the desired values on the wall vertices.

**Radiative Equilibrium Wall** The radiative equilibrium wall condition [13, 128] consists in imposing that the heat released from the gas into the wall by conduction ($q_g^{\text{cond}}$) and convection ($q_g^{\text{conv}}$) is exactly balanced by the heat lost by radiation ($q_r$) from the wall itself. This translates into the non linear equation:

$$ Q(T_w) = q_g^{\text{cond}} + q_g^{\text{conv}} + q_r = 0 $$

(5.127)

to be solved iteratively at the wall. After having substituted the actual expressions for all terms, this expression becomes:
\[ Q(T_w) = -\left( \lambda \frac{\partial T}{\partial n} + \sum_{m} \lambda_m \frac{\partial T_w}{\partial n} \right) + \sum_{s} h_s J_s \cdot n_w - \sigma (\epsilon^e_w T_w^4 - \epsilon^a_w T_t^4) = 0 \]

(5.128)

where \( \sigma = 5.67 \cdot 10^{-8} \text{ [W/m}^2/\text{K}^4] \) is the Stefan-Boltzmann constant, \( \epsilon^e_w \) and \( \epsilon^a_w \) are respectively the wall emissivity and absorptivity and \( T_t^4 \) is the distant body temperature. In Eq. 5.128 the contribution \( \lambda \frac{\partial T_w}{\partial n} \) has been discarded, because if the electron temperature is separated, a condition of adiabatic wall is generally applied to it [118, 128]. If we discretize the normal gradients of temperature in Eq. 5.128 and we plug-in the repositioning algorithm presented above we get the functional:

\[ Q(T_w) = -\sum_{j} \lambda_j \frac{T_{in} - T_j(T_w)}{\| \mathbf{X}_{in} - \mathbf{X}_g(T_w) \|} + \sum_{s} h_s (T_w) J_{s,n}(T_w) - \sigma (\epsilon^e_w T_w^4 - \epsilon^a_w T_t^4) \]

(5.129)

where the normal species diffusion fluxes \( J_{s,n} \) are computed by solving the Stefan-Maxwell equations [13] with the gradients of normal molar fractions as driving forces.

The solution \( T_w \) of the non linear equation \( Q(T_w) = 0 \) is then obtained by applying a Newton procedure, where the possible repositioning of the ghost node is also taken into account at each Newton step. In order to enhance numerical robustness, as recommended in [13], the maximum variation of wall temperature for each boundary face between two time steps is limited to 100 or 200 K.

**Local Equilibrium Wall** This condition imposes the chemical composition of the gas mixture at the wall to be equal to the equilibrium values. According to what described in Sec. 4.2.1, under the local thermodynamic and chemical equilibrium assumption, the species molar composition at the wall \( \mathbf{X}_w \) can be computed from the local temperature \( T_w \), pressure \( p_w \) and elemental molar composition \( \mathbf{X}_w^e \) by solving iteratively an algebraic system of equations [24, 94, 140]. In our implementation, \( p_w \) is extrapolated from the interior cell, whilst the wall temperature is either fixed or calculated from the radiative equilibrium condition, as explained in the previous section. The species mass fractions are then linearly interpolated between the inner and the possibly repositioned ghost node.

While in a full non-catalytic case, the heating due to gas convection is null,
i.e. \( q_{\text{conv}}^{s} = \sum_{s} h_{s} J_{s} \cdot n_{w} = 0 \), if we assume local equilibrium at the wall, this term plays a role, since the gradient of species fractions across the wall is not null and contributes to increase the heat flux.

### 5.2.8 Implementation issues

The Finite Volume discretization that has just been presented is tackled in COOLFluiD by means of the MCS pattern introduced in Sec. 2.4.1, as shown in the OMT diagram in Fig. 5.5.

![OMT diagram for the MCS pattern applied to the Finite Volume method.](image)

Figure 5.5: OMT diagram for the MCS pattern applied to the Finite Volume method.

To this end, the interface for a base `SpaceMethod` defined in C.L. 2.27 is implemented by a concrete `CellCenterFVM` class, whose definition is presented in C.L. 5.2. Each specific action such as initialization, setup, unsetup, residual and system jacobian computation, application of boundary conditions etc. is handled by a dedicated Command. Each available Command is given a name by the corresponding `Provider` object (see Sec. 2.5). As explained in Sec. 2.4.1.1, this gives to the end-user full control on the choice of the Commands to be used in the simulation for performing a certain task: he/she can easily select alternative Com-
mands to the default ones just by specifying their names in the input file. For instance, a user can select setup and unsetup Commands specific for a second order scheme based on least square reconstruction, which requires the allocation of many more data (cell limiters, cell gradients, weights etc.) than a first order one, in this way:

\[
\text{SpaceMethod} = \text{CellCenterFV} \\
\text{CellCenterFV.SetupCom} = \text{LeastSquareP1Setup} \\
\text{CellCenterFV.UnSetupCom} = \text{LeastSquareP1UnSetup}
\]

Analogously, the end-user can specify by name the boundary conditions, their parameters and the TRSs on which they are applied. For example:

```
# list of the BC Commands and corresponding aliases
CellCenterFV.BcComds = SuperInletFVCC MirrorFVCC
CellCenterFV.BcNames = SInlet Mirror

# TRS to which MirrorFVCC will be applied
```

**Code Listing 5.2:** CellCenterFV class definition

```cpp
// *** CellCenterFV.hh *** /
class CellCenterFV : public SpaceMethod {
public:
  typedef SelfRegistPtr<Command<CellCenterFVData>> FVCom; // constructor, destructor, overridden virtual functions

private:
  // data to share between FVCom commands
  std::auto_ptr<CellCenterFVData> m_data;

  std::pair<FVCom,string> m_setup; // setup Command
  std::pair<FVCom,string> m_unsetup; // unsetup Command

  // Commands computing the residual/jacobian
  std::pair<FVCom,string> m_computerSpaceRHS; // space part
  std::pair<FVCom,string> m_computerTimeRHS; // time part

  // Commands that initialize the solution in the domain
  std::vector<FVCom> m_inits;
  std::vector<string> m_initsStr; // init Commands names

  // Commands that computes the boundary conditions (bc)
  std::vector<FVCom> m_bcs;
  std::vector<string> m_bcsStr; // bc Commands names
};
```
CellCenterFV.Mirror.applyTRS = Symmetry

# input inlet profiles (function of x,y) for each variable
CellCenterFV.SInlet.Vars = x y
CellCenterFV.SInlet.Def = if(y<0.1,3,x^2) 50 0 2*sin(3*x)
CellCenterFV.SInlet.applyTRS = Inlet

```cpp
// --- CellCenterFVData.hh --- //
class CellCenterFVData : public SpaceMethodData {
public:
    //constructor, destructor, configuration functions
    // polynomial reconstructors
    SafePtr<PolyReconstructor> getPolyReconstructor() const;

    // convective, diffusive flux, source term computers
    SafePtr<FluxSplitter> getFluxSplitter() const;
    SafePtr<ComputeDiffusiveFlux> getDiffFluxComputer() const;
    SafePtr<ComputeSourceTerm> getSourceTermComputer() const;
    SafePtr<NodalExtrapolator> getNodalExtrapolator() const;

    // linearization variable set
    SafePtr<ConvectiveVarSet> getLinearizationVar() const;

    // vectorial transformer from update to solution variables
    SafePtr<VarSetTransformer> getUpdateToSolutionVecTrans() const;

    // other accessors/mutators ...
private:
    SelfRegistPtr<FluxSplitter> m_fluSplitter;
    string m_fluSplitterStr;

    SelfRegistPtr<PolyReconstructor> m_polyRec;
    string m_polyRecStr;

    // ... the same for all the other objects
};
```

**Code Listing 5.3**: CellCenterFVData class definition

All FVComs share access to a policy class [10], `CellCenterFVData`, which groups together all the Strategy objects needed by the Commands to fulfill their job: for sake of simplicity only `PolyReconstructor` (the polynomial reconstructor) and `FluxSplitter` (the flux splitting scheme) are shown in Fig. 5.5, but, in reality, other Strategies are present, such as
ComputeDiffusiveFlux (the diffusive flux discretizer), ComputeSourceTerm (the source term computer) and the NodalExtrapolator.
Moreover, CellCenterFVData also aggregates a number of Perspectives like VariableSets and VariableTransformers (see Sec. 2.3.1), that provide a dynamic binding of the numerical algorithm to the physics. As shown in C.L. 5.3, CellCenterFVData inherits the interface of a parent SpaceMethodData, which provides access to parameters, Strategies and Perspectives (e.g. a ConvectiveVarSet for the update $P$ and solution (or conservative) $U$ variables, a DiffusiveVarSet, etc.) which are common to possibly all the concrete SpaceMethods. Likewise all MethodData, CellCenterFVData is also a self-configurable object, and this implies that the user can select the concrete Strategies by name at run-time: e.g. Roe, AUSM, StegerWarming, ... as FluxSplitter, Constant or LeastSquare for PolyRecostructor, etc., while the developer can implement and register new ones without needing to modify the client code, as long as the core virtual interfaces defined in the kernel or in the module itself are respected.

5.2.8.1 Decoupling between numerics and physics

In order to show the power of an object-oriented approach in decoupling the numerical algorithm from the physical description of the problem, we consider again, as an example, the case of the Roe scheme, presented in Sec. 5.2.1.1:

$$\mathbf{F}^{Roe} = \frac{1}{2} \left[ \mathbf{F}_n(\mathbf{P}^L) + \mathbf{F}_n(\mathbf{P}^R) \right] - \frac{1}{2} \left| \mathbf{A}(\bar{Z}) \right| \left[ \mathbf{U}(\mathbf{P}^R) - \mathbf{U}(\mathbf{P}^L) \right] \quad (5.130)$$

In our case, the single terms in Eq. 5.130 are tackled as follows:

- the physical rotated fluxes $\mathbf{F}_n$ and the matrix $|\mathbf{A}|$ are given by the ConvectiveVarSet corresponding to the conservative variables $\mathbf{U}$;

- the average linearized state $\bar{Z}$ in which the jacobian $|\mathbf{A}|$ is evaluated is provided by a JacobianLinearizer object;

- the analytical vectorial transformation of the type $\mathbf{U}(\mathbf{P})$, i.e. from update variables $\mathbf{P}$ to conservative variables $\mathbf{U}$, is implemented by a VariableTransformer;

- the global physics-independent expression of the flux is assembled in RoeFlux, a concrete class deriving from FluxSplitter.
This design allows developers to focus solely on the implementation of purely physics-dependent terms, spread into a few Perspective objects, in order to extend the applicability of the basic Roe scheme. In this way, Euler, Magneto Hydrodynamics, Thermo-chemical nonequilibrium models etc. have been easily handled without touching the core algorithm. Moreover, additional flexibility if offered on the choice of update $P$ and linearization $Z$ variables.

In case a different formulation or a more efficient hard-coded version of the scheme is preferred, one can always provide an alternative implementation (a derived class from $\text{FluxSplitter}$ or $\text{RoeFlux}$ itself) possibly more strictly bound to the physics, at the cost of a potential loss in flexibility and reusability.