The LisbOn KInetics Boltzmann solver
Reference manual

Current version: `LoKI-B_25.10` and `LoKI-B++_25.10`

A Tejero-del-Caz[1], D Boer[2] and L L Alves[3]

[1]Departamento de Física, Facultad de Ciencias, Universidad de Córdoba, Campus de Rabanales, Spain
[2]Department of Applied Physics and Science Education, Eindhoven University of Technology, The Netherlands
[3]Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa, Portugal

email: loki@tecnico.ulisboa.pt

September 29, 2025

The following is the reference manual of the LisbOn KInetics Boltzmann (LoKI-B) simulation tool [1, 2]. LoKI-B solves the space independent form of the two-term electron Boltzmann equation (EBE) to calculate the isotropic and the anisotropic parts of the electron distribution function (the former usually termed the *electron energy distribution function*, EEDF). LoKI-B applies to non-magnetised non-equilibrium low-temperature plasmas (LTPs), excited by DC/HF electric fields or time-dependent (non-oscillatory) electric fields (a new feature introduced in version 2.0.0) [2] from different gases or gas mixtures. The tool uses a stationary description for DC fields, a Fourier time-expansion description for HF fields, and a time-dependent description for time-varying fields.

LoKI-B was developed as a response to the need of having an electron Boltzmann solver easily addressing the simulation of the electron kinetics in any complex gas mixture (of atomic / molecular species), describing first and second-kind electron collisions with any target state (electronic, vibrational and rotational), characterized by any user-prescribed population. LoKI-B includes electron-electron collisions, it handles rotational collisions adopting either a discrete formulation or a more convenient continuous approximation, and it accounts for variations in the number of electrons due to non-conservative events (ionisation and attachment) by assuming either a space-homogeneous exponential temporal growth or a time-constant exponential spatial growth of the electron density.

LoKI-B leverages on the scientific heritage in the field of non-equilibrium plasma kinetics of the Portuguese group N-Plasmas Reactive: Modeling and Engineering, N-PRiME [3] of *Instituto de Plasmas e Fusão Nuclear* with *Instituto Superior Técnico*, and it was developed resorting to well-grounded scientific foundations established years ago [4,5]. Several numerical improvements and updates have recently been implemented in LoKI-B, and further developments are ongoing, as part of a fruitful collaboration with the *Departamento de Física* of *Facultad de Ciencias* with *Universidad de Córdoba* and the group Elementary Processes in Gas Discharges, EPG [6] of the Eindhoven University of Technology.

The LisbOn KInetics Boltzmann solver is an open-source tool, distributed under the *GNU general public license*, and developed with an ontology that prioritizes the clear separation between tool and data. Its implementation follows a flexible and extensible object-oriented design, using the following programming languages:

- MATLAB® [7] for LoKI-B, leveraging its matrix-based architecture;
- C++ [8] for LoKI-B++, taking advantage of the higher performance of a compiled language, and linked with Eigen [9], a C++ template library for linear algebra.

LoKI-B++ is a C++ counterpart of MATLAB's LoKI-B, replicating essentially all of its original features. Unless otherwise specified, the acronym LoKI-B in this manual refers to either version of the code.

Both LoKI-B and LoKI-B++ are part of the LoKI tool suite and are freely available at `https://github.com/LoKI-suite`, for users to perform electron kinetics calculations, and for modelers / developers who are invited to continue testing the tool and/or to contribute for its development and improvement.

You are much welcome to report any problem or bug you may find using the code, or to send any feedback with suggestions for further developments.

Acknowledgments

This work was partially funded by Portuguese FCT - Fundação para a Ciência e a Tecnologia, initially under project PTDC/FISPLA/1243/2014 (KIT-PLASMEBA), and currently under projects 2022.04128.PTDC, UIDB/50010/2020, UIDP/50010/2020 and LA/P/0061/2020.

Part of this work is supported by the Dutch Science Foundation NWO, and by ASML.

Contents

1 Introduction

The simulation tool LoKI-B solves the space independent form of the two-term EBE, calculating the EEDF and the corresponding electron macroscopic parameters [1]. LoKI-B can also adopt a prescribed EEDF, such as a generalized Maxwellian. The current version of the tool is for non-magnetised LTPs. The initial capabilities of LoKI-B have been extended to the solution of the EBE for electrons under the action of a time-dependent (non-oscillatory) electric field [2], as described in section 1.3.

The scientific foundations of LoKI-B were established years ago [4, 5], and are based on the early works of Lorentz, Holstein, Allis and Delcroix [10–13]. The topical review [14], integrated in the collection *Foundations of low-temperature plasmas and their applications* published by *Plasma Sources Science and Technology*, gives a summary of the essentials of the EBE, when written for an electron distribution function expanded in Legendre polynomials $P_l(\cos(\theta))$ around the angle θ, defining the spatial orientation of the velocity vector with respect to the polar direction of the total anisotropy (produced by electric fields and density gradients). The users are advised to carefully read the extensive literature on the two-term EBE [14], for obtaining details about its solution.

Also, the topical review [1] presents the basic features of LoKI-B (also included in this user manual), along with examples of results obtained for different model and real gases, verifying the tool against analytical results, benchmarking it against numerical calculations, and validating the output by comparison with available measurements of swarm parameters.

1.1 General description of the problem

LoKI-B describes the electron kinetics of a plasma with electron density n_e, excited from a general mixture of atomic / molecular gases, by applying an electric field along the z-axis with the form

$$\vec{E}(t) = -E_p \cos(\omega t)\, \vec{e}_z \quad , \tag{1}$$

where the oscillation frequency ω is either

- $\omega = 0$, in the DC case;
- $\omega \gg \nu_c$ (ν_c is the typical collision frequency), in the HF case.

LoKI-B can also describe plasmas excited by time-dependent (non-oscillatory, $\omega = 0$) electric fields along the z-axis (see section 1.3).

LoKI-B handles simulations in a gas mixture, with user-defined distributions for the internal degrees of freedom (electronic, vibrational and rotational) of each gas, organized as a system of "Russian puppets". Table 1 exemplifies the system of states for a N_2-N-He gas mixture. Here,

- **System:** is the ensemble of all the gases present in the simulation ($N_2 \cup N \cup He \cup \cdots$).
- **Gas:** refers to each type of atom or molecule present in the simulation (N_2, N, He, ...). Each gas k has mass M_k and individual density N_k, being composed by several electronic / vibrational / rotational levels.
- **State:** specific configurations of electronic (i), vibrational (v) and/or rotational (J) levels belonging to a given gas (e.g, N_2(X), N_2(X,v=0), N_2(X,v=0,J=2), N(^{4}S), He(2^3S), ...). Each gas k is composed by several electronic states k_i with absolute density N_{k_i}; each k_i-state can include several vibrational states k_{i_v} with absolute density $N_{k_{i_v}}$, each one composed by several rotational states $k_{i_{v_J}}$ with absolute density $N_{k_{i_{v_J}}}$.
- **Parent (state):** state of a certain gas comprising a set of child states. Parent states can be electronic or vibrational, but not rotational (*e.g.*, N_2(X) is the parent of N_2(X,v=0,1,2,...)). Each parent level comprises all of its children.
- **Child (states):** ensemble of states that share the same parent. Child states can be vibrational or rotational, but not electronic (*e.g.*, N_2(X,v=0,J=0,1,2,...) are the children of N_2(X,v=0)).
- **Sibling (states):** relation between the states that share the same parent (*e.g.*, N_2(X,v=0,1,2,...) are siblings).

Table 1: Example of a gas mixture (childless states are displayed in red).

	Gases	States		
System	**N_2**	**N_2(X)** Parent of → ← Childs of	**N_2(X,v=0)** Parent of → ← Childs of	**N_2(X,v=0,J=0)**
				N_2(X,v=0,J=1)
				N_2(X,v=0,J=2)
				⋮
			N_2(X,v=1)	
			N_2(X,v=2)	
			⋮	
		N_2(A)		
		N_2(B)		
		⋮		
	N	**N(^{4}S)**		
		N(^{2}D)		
		N(^{2}P)		
		⋮		
	He	**He(1^1S)**		
		He(2^3S)		
		He(2^1S)		
		⋮		
	⋮			

- **Childless (states):** ensemble of the innermost states (without children) (*e.g.*, N_2(X,v=0,J=0) is childless).

The previous terminology, allows introducing several definitions and relations for the particle densities with the different species (gases / states) in the system.

- **Total gas density:** total particle density of the heavy species in the system, evaluated as:

$$N = \frac{p}{k_B T_g} \quad , \tag{2}$$

  where p is the total pressure, k_B is the Boltzmann constant and T_g is the gas temperature (assumed equal for all k-gases in the system).
  In the code, this quantity is stored in variables labeled with `totalGasDensity`.

- **Individual gas density:** total particle density of a gas in the system. For gas k this is given by N_k, with $\sum_k N_k = N$.
  In the code, this quantity is stored in variables labeled with `individualGasDensity`.

- **Gas fraction:** fraction of a gas in the system. For gas k this is given by $\chi_k \equiv N_k/N$.

- **Absolute density:** particle (absolute) density of a state. For a generic state i of gas k this is represented by N_{k_i}.
  In the code, this quantity is stored in variables labeled with `abs`.

- **Population:** density of a state relative to its siblings. For a generic state i of gas k this is represented by $\xi_{k_i} \equiv N_{k_i}/N_{\text{parent of } k_i} = N_{k_i}/(\sum_{i\in\text{siblings}} N_{k_i})$.
- **Density:** particle density of a state relative to the total gas density. For a generic state i of gas k this is represented by $\delta_{k_i} \equiv N_{k_i}/N$.
  In LoKI-B, this quantity is stored in variable `density`.

From the previous definitions, it is straightforward to obtain the following normalisation conditions

$$\sum_{k\in\text{system}} \chi_k = 1 \tag{3a}$$

$$\sum_{i\in\text{siblings}} \xi_i = 1 \tag{3b}$$

$$\sum_{i\in\text{childless}} \delta_i = 1 \quad , \tag{3c}$$

concluding also that the following relationships hold

$$\delta_i = \xi_i \delta_{i\text{'s parent}} \tag{4a}$$

$$\delta_{k_i} = \xi_{k_i}\chi_k \quad . \tag{4b}$$

LoKI-B uses SI units for all physical quantities, except the energies that are expressed in eV (electron-volt) and the reduced fields that are expressed in Td (Townsend; 1 Td = 10^{-21} V m^2).

1.2 The electron Boltzmann equation

In the classical two-term approximation, the electron distribution function can be written assuming a separation between the Legendre expansion of the energy distribution and the space-time profile of the electron density $n_e(z,t)$ as follows

$$\begin{aligned} F(z,u,t) &\simeq \left\{ f^0(u) + \left[\vec{f}^1(u)\cdot\frac{\vec{v}}{v}\right] \cos(\omega t)\right\} n_e(z,t) \\ &= \left[f^0(u) + f^1(u)\cos(\theta)\ \cos(\omega t)\right] n_e(z,t) \quad , \end{aligned} \tag{5}$$

where $u = m_e v^2/(2e)$ is the electron kinetic energy in eV (with m_e and e the electron mass and charge, respectively); $f^0(u) \equiv f(u)$ is the isotropic part of the electron distribution function, identified with the EEDF in the framework of the two-term approximation, and satisfying the normalisation condition $\int_0^\infty f(u)\sqrt{u}du = 1$; and $f^1(u)$ is the (real part of the) first-anisotropy of the electron distribution function.

Under the previous conditions, the EBE including a non-conservative collisional operator can be written as (with $\gamma \equiv \sqrt{2e/m_e}$) [1]

$$\frac{1}{\gamma\sqrt{u}}\frac{\langle\nu_{\text{eff}}\rangle}{N} u f(u) + \frac{1}{N}\frac{1}{\gamma}\frac{dG(u)}{du} = S(u) \tag{7a}$$

$$f^1(u) = -\frac{\zeta(E/N)}{\Omega_{\text{PT}}(u)}\frac{df(u)}{du} \quad , \tag{7b}$$

or

$$\frac{1}{3}\frac{\alpha_{\text{eff}}}{N} u f^1(u) + \frac{1}{N}\frac{1}{\gamma}\frac{dG(u)}{du} = S(u) \tag{8a}$$

$$f^1(u) = -\frac{(\alpha_{\text{eff}}/N)}{\Omega_{\text{SST}}(u)} f(u) - \frac{(E/N)}{\Omega_{\text{SST}}(u)}\frac{df(u)}{du} \quad , \tag{8b}$$

where (7a) and (8a) or (7b) and (8b) correspond to the isotropic and anisotropic components of the EBE, respectively, in the following approximations:

[1] As mentioned, in a HF case the quantity $f^1(u)$ in (7b) represents the *real part of* the first-anisotropy of the electron distribution function. Indeed, in this case $f^1(u,t) = \tilde{f}^1(u)e^{j\omega t}$ with

$$\tilde{f}^1(u) = -\gamma\sqrt{u}\frac{E_p}{N\gamma\sqrt{u}\,\Omega_c(u) + j\omega}\frac{df(u)}{du} = -\frac{\zeta(E/N)}{\Omega_{\text{PT}}(u)}\frac{df(u)}{du} + j\frac{\zeta(E/N)\,\frac{\omega/N}{\Omega_c\gamma\sqrt{u}}}{\Omega_{\text{PT}}(u)}\frac{df(u)}{du} \quad . \tag{6}$$

- **temporal growth model:** the space-homogeneous equations (7a)-(7b), which can be used in either the DC or the HF cases, assume an exponential temporal growth of the electron density with a growth constant $\langle\nu_{\rm eff}\rangle \equiv \langle\nu_{\rm ion}\rangle - \langle\nu_{\rm att}\rangle \equiv \sum_{k,i}\delta_{k_i}\left[\langle\nu_{k_i,{\rm ion}}\rangle - \langle\nu_{k_i,{\rm att}}\rangle\right]$, corresponding to the electron net creation frequency for the system under study

$$\frac{\partial n_e(t)}{\partial t} = \langle\nu_{\rm eff}\rangle n_e(t) \quad , \tag{9}$$

with $\langle\nu_{\rm ion}\rangle$ and $\langle\nu_{\rm att}\rangle$ the corresponding electron mean frequencies for ionisation and attachment, respectively, and where the sums extend over all levels of all gases in the mixture (see section 2.1). The temporal growth model was used to simulate Pulsed Townsend (PT) discharges by Tagashira *et al* [15], using the formulation of Thomas [16] for the one dimensional continuity equation of electrons under the action of an uniform electric field;

- **spatial growth model:** the time-constant equations (8a)-(8b), which can only be used in the DC case, assume an exponential spatial growth of the electron density in the direction opposite to that of the applied electric field, with a growth constant $\alpha_{\rm eff} \equiv \alpha - \eta \equiv \sum_{k,i}\delta_{k_i}\left(\alpha_{k_i} - \eta_{k_i}\right)$, corresponding to the effective electron Townsend coefficient

$$\frac{\partial n_e(z)}{\partial z} = \alpha_{\rm eff}\, n_e(z) \quad , \tag{10}$$

with α and η the corresponding first and second Townsend coefficients for ionisation and attachment, respectively) (see section 2.1). The spatial growth model is usually adopted when simulating Steady State Townsend (SST) discharges maintained with a DC electric field [15].

Note that the inclusion of growth models for the electron density is necessary if one intends a space and time independent description, yet including non-conservative collisional mechanisms such as electron ionisation and attachment. Note further that the current version of LoKI-B does not include electron recombination mechanisms.

In these equations:

- E/N is the *reduced electric field*, where $E \equiv E_p/\zeta$ (with $\zeta = 1\,{\rm or}\,\sqrt{2}$ for the DC or the HF cases, respectively);

- the quantities $\Omega_{\rm PT}$ and $\Omega_{\rm SST}$ have the dimensions of a cross section, being defined as

$$\Omega_{\rm PT}(u) \equiv \Omega_c(u) + \frac{\left[1/(\gamma^2 u)\right](\omega/N)^2}{\Omega_c(u)} \tag{11a}$$

$$\Omega_{\rm SST}(u) \equiv \sigma_c(u) \quad , \tag{11b}$$

where $\Omega_c(u) \equiv \sigma_c(u) + \left[1/(\gamma\sqrt{u})\right]\left(\langle\nu_{\rm eff}\rangle/N\right)$, with $\sigma_c(u) \equiv \sum_k \chi_k \sigma_{k,c}$ the electron-neutral total momentum-transfer cross section, for the system under study, and $\sigma_{k,c}$ the corresponding cross section for gas k;

- $G \equiv G_E + G_{\rm el} + G_{\rm CAR} + G_{ee}$ is the *upflux* function that contains power gain/loss contributions due to, in order, the applied electric-field, the elastic collisions, the rotational collisions (in the continuous approximation for rotations, CAR, with a Chapman-Cowling correction due to the gas temperature [17–20], written here for the case of homonuclear diatomic molecules), and the electron-electron collisions [19], respectively

$$\begin{aligned} G_E(u) &= N\gamma\frac{u}{3}\frac{1}{\zeta}\frac{E}{N}f^1(u) \\ &= -N\gamma\begin{cases} g_E^{\rm PT}(u)\left(\dfrac{E}{N}\right)^2\dfrac{df(u)}{du} & , \text{ for a PT situation} \\ g_E^{\rm SST}(u)\left[\dfrac{E}{N}\dfrac{\alpha_{\rm eff}}{N}f(u) + \left(\dfrac{E}{N}\right)^2\dfrac{df(u)}{du}\right] & , \text{ for a SST situation} \end{cases} \end{aligned} \tag{12a}$$

$$\begin{aligned} G_{\rm el}(u) &= -\sum_k 2\frac{m_e}{M_k}\nu_{k,c}^{\rm el}(u)u^{3/2}\left[f(u) + \frac{k_B T_g}{e}\frac{df(u)}{du}\right] \\ &= -N\gamma\, g_{\rm el}(u)\left[f(u) + \frac{k_B T_g}{e}\frac{df(u)}{du}\right] \end{aligned} \tag{12b}$$

$$
\begin{aligned}
G_{\mathrm{CAR}}(u) &= -\sum_k 4B_k\nu_{0,k}u^{1/2}\left[f(u)+\frac{k_BT_g}{e}\frac{df(u)}{du}\right] \\
&= -N\gamma\, g_{\mathrm{CAR}}(u)\left[f(u)+\frac{k_BT_g}{e}\frac{df(u)}{du}\right] \qquad (12c)\\
G_{ee}(u) &= -2\nu_{ee}(u)u^{3/2}\left[I(u)f(u)+J(u)\frac{df(u)}{du}\right] \\
&= -N\gamma\, g_{ee}(u)\left[I(u)f(u)+J(u)\frac{df(u)}{du}\right] \quad , \qquad (12d)
\end{aligned}
$$

with

$$
\begin{aligned}
g_E^{\mathrm{PT}}(u) &= \frac{u}{3}\frac{1}{\Omega_{\mathrm{PT}}(u)} \qquad (13a)\\
g_E^{\mathrm{SST}}(u) &= \frac{u}{3}\frac{1}{\Omega_{\mathrm{SST}}(u)} \equiv D(u) \qquad (13b)\\
g_{\mathrm{el}}(u) &= 2u^2\sum_k \frac{m_e}{M_k}\chi_k\sigma_{k,c}^{\mathrm{el}}(u) \qquad (13c)\\
g_{\mathrm{CAR}}(u) &= 4u\sum_k B_k\chi_k\sigma_{0,k}(u) \qquad (13d)\\
g_{ee}(u) &= \frac{2}{\gamma}\frac{\nu_{ee}(u)}{N}u^{3/2} \quad , \qquad (13e)
\end{aligned}
$$

and where $\nu_{k,c}^{\mathrm{el}} \equiv N_k\gamma\sqrt{u}\,\sigma_{k,c}^{\mathrm{el}}$ is the electron-neutral elastic momentum-transfer collision frequency for the gas k ($\sigma_{k,c}^{\mathrm{el}}$ being the corresponding cross section), $\nu_{0,k} \equiv N_k\gamma\sqrt{u}\,\sigma_{0,k}$ is the electron-neutral collision frequency for the rotational excitation of gas k ($\sigma_{0,k} \equiv 8\pi Q_k^2 a_0^2/15$ being the corresponding cross section, with a_0 the Bohr radius), B_k and Q_k are the rotational constant and the quadrupole-moment constant (in units of ea_0^2) of the gas k, respectively [2], $I(u)$ and $J(u)$ are the relevant Spitzer integrals [12, 21], and $\nu_{ee} \equiv 4\pi\left[e^2/(4\pi\varepsilon_0 m_e)\right]^2 \ln\Lambda_c/v^3 n_e$ is the electron-electron collision frequency (with ε_0 the vacuum permittivity, $\ln\Lambda_c$ the Coulomb logarithm ($\Lambda_c \equiv 12\pi n_e\lambda_D^3$) and λ_D the Debye length);

- $S \equiv \sum_{i,j>i} S_{i,j} + \sum_i S_{i,\mathrm{ion}} + \sum_i S_{i,\mathrm{att}}$ is the discrete collision operator that contains power loss/gain contributions due to inelastic/superelastic mechanisms, ionisation and attachment (as mentioned before, recombination is not yet considered), respectively

$$
\begin{aligned}
S_{i,j}(u) &= \delta_i\left[(u+V_{i,j})\sigma_{i,j}(u+V_{i,j})f(u+V_{i,j}) - u\sigma_{i,j}(u)f(u)\right] \\
&+ \delta_j\frac{g_i}{g_j}\left[u\sigma_{i,j}(u)f(u-V_{i,j}) - (u+V_{i,j})\sigma_{i,j}(u+V_{i,j})f(u)\right] \qquad (14a)\\
S_{i,\mathrm{ion}}(u) &= \delta_i\left\{\int_{u+V_{i,\mathrm{ion}}}^{2u+V_{i,\mathrm{ion}}} u'\sigma_{i,\mathrm{ion}}^{\mathrm{sec}}(u',u'-V_{i,\mathrm{ion}}-u)f(u')du'\right. \\
&+ \left.\int_{2u+V_{i,\mathrm{ion}}}^{\infty} u'\sigma_{i,\mathrm{ion}}^{\mathrm{sec}}(u',u)f(u')du' - u\sigma_{i,\mathrm{ion}}(u)f(u)\right\} \qquad (14b)\\
S_{i,\mathrm{att}}(u) &= -\delta_i u\sigma_{i,\mathrm{att}}(u)f(u) \quad . \qquad (14c)
\end{aligned}
$$

In (14a)-(14c), the subscripts i, j and the densities δ_i, δ_j now refer to any state of any gas in the system; $\sigma_{i,j}$ is the electron-collision cross section for the $i \to j$ excitation with energy threshold $V_{i,j}$ and statistical weights g_i and g_j, respectively (note that the third and fourth superelastic terms in (14a) were written resorting to the Klein-Rosseland relation [22]); $\sigma_{i,\mathrm{att}}(u)$ is the electron-collision cross section for the attachment of state i; $\sigma_{i,\mathrm{ion}}^{\mathrm{sec}}(u',u)$ is the single differential cross section (SDCS) for the ionisation of state i with energy threshold $V_{i,\mathrm{ion}}$; and

$$
\sigma_{i,\mathrm{ion}}(u) = \int_0^{(u-V_{i,\mathrm{ion}})/2} \sigma_{i,\mathrm{ion}}^{\mathrm{sec}}(u,u')du' \qquad (15)
$$

[2] In `LoKI-B_v1.0.0`, the rotational cross section $\sigma_{0,k}$ was wrongly calculated with a linear (not quadratic) dependence on the quadrupole-moment constant Q_k. The bug is now corrected. We acknowledge Jan van Dijk and Daan Boer (TU/e, Eindhoven, The Netherlands) for pointing this out.

is the integral cross section for the ionisation of state i. The SDCS takes into account the distribution of the energy available after an ionisation event, caused by a primary electron with energy u', yielding a secondary electron born in the event with energy u and a scattered electron with energy $u' - V_{i,\mathrm{ion}} - u$.

Note that the effect of rotational inelastic/superelastic mechanisms, in the electron kinetics of homonuclear diatomic molecules, can be considered by adopting either the CAR approach of (12c) or the discrete approach of the collision operator (14a), yielding similar results provided that the rotational constants Q_k and B_k are coherently matched to the rotational cross section set $\sigma_{i,j}$ [17].

LoKI-B allows the user to choose between different descriptions of the ionisation mechanism, namely:

- **conservative:** taking ionisation as conservative a mechanism, described by the inelastic part of the collisional operator (14a), in which case the first terms in the left-hand side of (7a) and (8a) should be set to zero;
- **usingSDCS:** taking ionisation as a non-conservative mechanism, described by the collisional operator (14b) with a SDCS, and using the EBE given by (7a)-(8b);
- **oneTakesAll:** taking ionisation as a non-conservative mechanism, and prescribing no-sharing of energy between the secondary electron, introduced at $u = 0$, and the scattered electron, introduced at energy $u' - V_{i,\mathrm{ion}}$. In this case the SDCS for the secondary electrons writes $\sigma^{\mathrm{sec}}_{i,\mathrm{ion}}(u', u) = \sigma_{i,\mathrm{ion}}(u')\delta(u)$ and the ionisation operator becomes

$$S_{i,\mathrm{ion}}(u) = \delta_i \left[(u + V_{i,\mathrm{ion}})\sigma_{i,\mathrm{ion}}(u + V_{i,\mathrm{ion}})f(u + V_{i,\mathrm{ion}}) - u\sigma_{i,\mathrm{ion}}(u)f(u) + \frac{1}{\gamma}\frac{\langle \nu_{i,\mathrm{ion}} \rangle}{N}\delta(u) \right] , \tag{16}$$

  corresponding to the inelastic part of (14a) plus an extra term that describes the introduction of new electrons at zero-energy;
- **equalSharing:** taking ionisation as a non-conservative mechanism, and prescribing equal-energy-sharing between the secondary and the scattered electrons, introduced at energy $(u' - V_{i,\mathrm{ion}})/2$. In this case the SDCS for the secondary electrons writes $\sigma^{\mathrm{sec}}_{i,\mathrm{ion}}(u', u) = \sigma_{i,\mathrm{ion}}(u')\delta[u - (u' - V_{i,\mathrm{ion}})/2]$ and the ionisation operator becomes

$$S_{i,\mathrm{ion}}(u) = \delta_i \left[4(2u + V_{i,\mathrm{ion}})\sigma_{i,\mathrm{ion}}(2u + V_{i,\mathrm{ion}})f(2u + V_{i,\mathrm{ion}}) - u\sigma_{i,\mathrm{ion}}(u)f(u) \right] , \tag{17}$$

  where the first term of $S_{i,\mathrm{ion}}(u)du$ describes the entrance of both the secondary and the scattered electrons at a total energy between $2u$ and $2u + 2du$, produced by a primary electron with energy $u' = 2u + V_{i,\mathrm{ion}}$, whereas the second term describes the exit of primary electrons with energy u after ionizing collisions.

Naturally, the attachment mechanism is taken **always** as non-conservative, meaning that, in the case of electronegative gases, the code adopts the electron-density growth-model (**temporal** or **spatial**) specified by the user.

Note that the current version of LoKI-B does not include any external sources of electrons, such as electron beam injection or photoionization. However, the model adopted in the code can be easily upgraded as to consider these effects: (i) on the right-hand side of the EBE, with the addition, to the source function S, of an energy-resolved gain-term; (ii) on the left-hand side of the EBE, with the addition of the corresponding energy-integral term, describing the growth of the electron density in a stationary situation. The inclusion of these external sources of electrons, as well as the handling of the non-conservative electron-collision mechanisms (particularly the attachment), must be carefully monitored in all cases, since they can be responsible for the development of significant anisotropies, in which case the two-term expansion of the EBE adopted here is no longer valid.

1.3 The time-dependent electron Boltzmann equation

The original capabilities of LoKI-B have been extended in order to obtain the solution of the EBE for plasma electrons under the action of a time-dependent (non-oscillatory, $\omega = 0$) electric-field along the z-axis [2]. In this case, assuming the classical two-term expansion and a space-independent exponential temporal growth of the

electron density

$$\begin{aligned} F(u,t) &\simeq \left[f^0(u,t) + f^1(u,t)\,\cos(\theta)\right] n_e(t) \\ \frac{dn_e(t)}{dt} &= \langle \nu_{\rm eff}\rangle(t) n_e(t) \quad , \end{aligned} \tag{18}$$

the EBE is updated as follows (see equations (7a)-(7b)):

$$\frac{1}{N}\frac{1}{\gamma}\sqrt{u}\frac{\partial f(u,t)}{\partial t} + \frac{1}{\gamma\sqrt{u}}\frac{\langle \nu_{\rm eff}\rangle(t)}{N} u f(u,t) + \frac{1}{N}\frac{1}{\gamma}\frac{\partial G(u,t)}{\partial u} = S(u,t) \tag{19a}$$

$$f^1(u,t) = -\frac{(E(t)/N)}{\Omega_c(u,t)}\frac{\partial f(u,t)}{\partial u} \quad . \tag{19b}$$

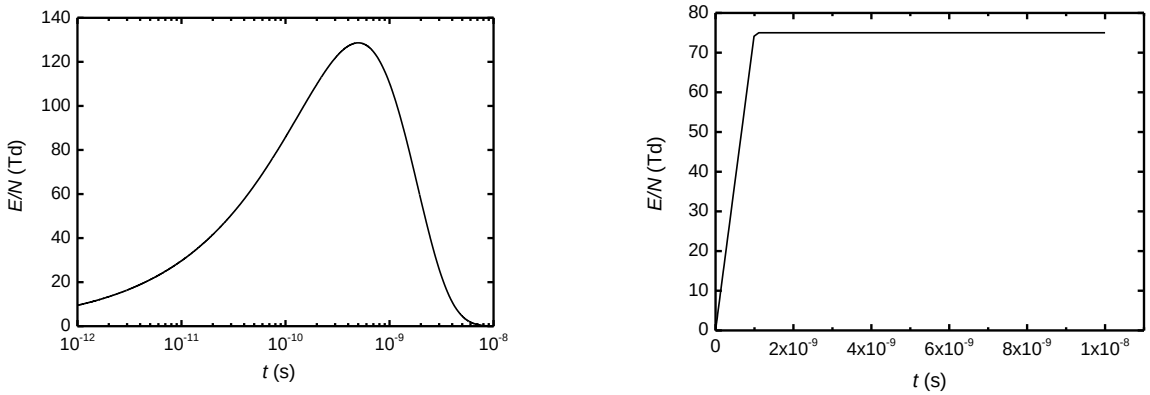


Figure 1: Example of time-dependent (non-oscillatory) reduced electric-fields that can be used in LoKI-B simulations: electric-field pulse (left) and electric-field step (right), calculated using (22a) and (22b), respectively.

In these equations, the first term on the left-hand side of (19a) accounts for the intrinsic time evolution of the isotropic component of the electron distribution function; the second term on the left-hand side of (19a) and the time-dependent collision frequency $\Omega_c(u,t)$ in (19b) account for the intrinsic time evolution of the electron density. In this case, the collision frequency is given by

$$\Omega_c(u,t) \equiv \sigma_c(u) + \frac{1}{\gamma\sqrt{u}}\frac{1}{N n_e(t)}\frac{dn_e(t)}{dt} = \sigma_c(u) + \frac{1}{\gamma\sqrt{u}}\frac{\langle \nu_{\rm eff}\rangle(t)}{N} \quad . \tag{20}$$

It is assumed a steady-state form for equation (19b), describing the first anisotropy, because the characteristic evolution time of the EEDF τ is much larger than the characteristic evolution time of the anisotropic component, *i.e.*

$$\tau \simeq \frac{1}{\sum_k (m_e/M_k)\,\nu_{k,c}^{\rm el}} \gg \frac{1}{\nu_{k,c}^{\rm el}} \quad . \tag{21}$$

Equations (19a)-(19b) are used to obtain the time-dependent solution of the EBE for user-prescribed electric-field forms, initial and final times, and sampling values of the field. As an example, figure 1 shows two forms of the reduced electric-field that can be used in the time-dependent solution of LoKI-B, corresponding to an electric field pulse given by

$$\frac{E(t)}{N}(\text{Td}) = \frac{E_0}{N}\sqrt{\frac{t(\text{s})}{\tau_{\rm rise}}}\,\exp\left(-\frac{t(\text{s})}{\tau_{\rm rise}}\right) \tag{22a}$$

with $E_0/N = 300$ Td and $\tau_{\rm rise} = 10^{-9}$ s; and an electric-field step given by

$$\frac{E(t)}{N}\,(\text{Td}) = \begin{cases} \frac{E_{\max}}{N}\frac{t(\text{s})}{\tau_{\rm rise}}\,, & 0 \le t(\text{s}) \le \tau_{\rm rise} \\ \frac{E_{\max}}{N}\,, & t(\text{s}) > \tau_{\rm rise} \end{cases} \quad , \tag{22b}$$

with $E_{\max}/N = 75$ Td and $\tau_{\rm rise} = 10^{-9}$ s.

1.4 Prescribed EEDFs

LoKI-B calculations can also adopt a prescribed EEDF, corresponding to a generalized Maxwellian given by the expression [23–25]

$$f(u) = s\left(\frac{2}{3T_e}\right)^{3/2} \frac{\Gamma(\frac{5}{2s})^{3/2}}{\Gamma(\frac{3}{2s})^{5/2}} \exp\left(-\left[\frac{\Gamma(\frac{5}{2s})}{\Gamma(\frac{3}{2s})}\frac{2u}{3T_e}\right]^s\right) \quad , \tag{23}$$

where Γ represents the gamma-function, T_e is in eV and s is a positive real number that controls the "shape" of the EEDF. In particular, (23) yields a Maxwellian EEDF and a Druyvesteyn EEDF for $s = 1$ and $s = 2$, respectively.

This setting corresponds to a local thermodynamic equilibrium framework, where the electron energy (defined by the electron temperature, given as input parameter) is dissipated exclusively in electron-neutral *conservative* collisions. Thus, in this case LoKI-B considers neither an electron-density growth model nor time-dependent calculations. Indeed, when adopting a prescribed EEDF, LoKI-B calculations are limited to evaluating (i) the EEDF for given T_e, according to (23), (ii) the electron rate coefficients for the set of cross sections defined as input data (see section 2.1), (iii) the most relevant electron transport parameters (see section 2.1), (iv) the power transferred in electron-neutral elastic/inelastic/superelastic collisions (see section 2.2), and (v) the *equivalent* reduced electric-field that would ensure a local equilibrium between the power gained from the field and lost in collisions (see section 2.2).

2 Simulation input/output

The general input of LoKI-B defines the operating work conditions (*e.g.* the applied reduced electric field and frequency, the gas pressure and temperature, the electron density, and the fractions of the different gases in the case of mixtures), the distribution of populations for the electronic, vibrational and rotational states of the atomic / molecular gases considered, and the relevant sets of electron-scattering cross sections obtained from the open-access website LXCat [26].

On output, LoKI-B calculates the isotropic and the anisotropic parts of the electron distribution function, in addition to various electron macroscopic parameters obtained by integration over the EEDF (obtained from the solution to the EBE or adopting some other form prescribed by the user, *e.g.* a generalized Maxwellian EEDF), namely:

- swarm parameters (transport parameters and rate coefficients), which can be used to adjust complete sets of electron-scattering cross sections [27, 28] or as input parameters in macroscopic (fluid / global) plasma models [14, 29, 30];
- power-transfer terms [14, 21], providing the distribution of power among the different collisional channels and controlling the calculation errors (see section 7.3.1).

2.1 Swarm parameters

The electron mean energy (in eV) is defined as

$$\varepsilon \equiv \int_0^\infty u^{3/2} f(u) du \quad , \tag{24}$$

and is used to introduce the kinetic temperature (in eV)

$$T_e \equiv \frac{2}{3}\varepsilon \quad . \tag{25}$$

The DC reduced transverse free-diffusion coefficient (in $\mathrm{m^{-1}\ s^{-1}}$) and the DC reduced mobility (in $\mathrm{V^{-1}\ m^{-1}\ s^{-1}}$) are calculated from [5, 14, 21]

$$D_e N = \frac{\gamma}{3} \int_0^\infty \frac{u}{\Omega_x(u)} f(u) du \tag{26a}$$

$$\mu_e N = -\frac{\gamma}{3} \int_0^\infty \frac{u}{\Omega_x(u)} \frac{df(u)}{du} du \quad , \tag{26b}$$

with $x = c, \mathrm{SST}$ for temporal or spatial growth of the electron density, respectively (see (11a)-(11b)), and these quantities allow defining the electron characteristic energy (in eV)

$$u_k \equiv \frac{D_e N}{\mu_e N} \quad . \tag{27}$$

Similarly to (26a)-(26b), the DC reduced transverse diffusion coefficient and the DC reduced mobility for energy transport (in eV $\mathrm{m^{-1}\ s^{-1}}$ and $\mathrm{m^{-1}\ s^{-1}}$, respectively) are calculated from [5, 31] [3]

$$D_\varepsilon N = \frac{\gamma}{3} \int_0^\infty \frac{u^2}{\Omega_x(u)} f(u) du \tag{28a}$$

$$\mu_\varepsilon N = -\frac{\gamma}{3} \int_0^\infty \frac{u^2}{\Omega_x(u)} \frac{df(u)}{du} du \quad , \tag{28b}$$

with $x = c, \mathrm{SST}$ for temporal or spatial growth of the electron density, respectively (see (11a)-(11b)).

[3] These transport coefficients correspond to the thermoelectricity β and the heat diffusion G defined in [12]. Note that here, and contrary to [5, 12, 31], D_ε and μ_ε are not normalized to the electron mean energy ε.

In the case of HF simulations at $\omega \gg \nu_c$, in addition to the DC parameters (26a)-(28b) that might be of interest for fluid/global models, the code evaluates the *complex* HF mobility, with the real [4] and the imaginary parts given by [21]

$$\Re[\mu_{\rm HF} N] = -\frac{\gamma}{3}\int_0^\infty \frac{u}{\Omega_{\rm PT}(u)}\frac{df(u)}{du}du \tag{29a}$$

$$\Im[\mu_{\rm HF} N] = \frac{\gamma}{3}\int_0^\infty \frac{u}{\Omega_{\rm PT}(u)}\frac{\omega/N}{\Omega_c\gamma\sqrt{u}}\frac{df(u)}{du}du \quad . \tag{29b}$$

The rate coefficients (in $\mathrm{m^3\ s^{-1}}$), for excitation / de-excitation mechanisms between states i and $j > i$, are given by [14]

$$C_{i,j} = \gamma\int_0^\infty u\sigma_{i,j}(u)f(u)du \tag{30a}$$

$$C_{j,i} = \gamma\int_0^\infty \frac{g_i}{g_j}(u+V_{i,j})\sigma_{i,j}(u+V_{i,j})f(u)du \quad . \tag{30b}$$

Note that (30a)-(30b) can be applied not only to the cross sections adopted for solving the EBE, but also to the *extra* set of cross sections defined by the user as additional input data of LoKI-B (see section 4).

The ionisation and the attachment rate coefficients are defined similarly to (30a) as

$$C_{i,\rm ion} \equiv \frac{\langle\nu_{i,\rm ion}\rangle}{N} = \gamma\int_0^\infty u\sigma_{i,\rm ion}(u)f(u)du \tag{31a}$$

$$C_{i,\rm att} \equiv \frac{\langle\nu_{i,\rm att}\rangle}{N} = \gamma\int_0^\infty u\sigma_{i,\rm att}(u)f(u)du \quad . \tag{31b}$$

In a DC case, the **anisotropic equations** (7b) and (8b) can be integrated after multiplication by $\gamma u/3$, to define the electron mean drift velocity (in $\mathrm{m\ s^{-1}}$)

$$v_d \equiv \frac{\gamma}{3}\int_0^\infty u f^1(u)du \quad ,$$

yielding, respectively,

$$v_d = \mu_e E \ , \ \textbf{for a PT situation} \tag{32a}$$

$$v_d = \mu_e E - D_e\frac{1}{n_e}\frac{dn_e}{dz} = \mu_e E - D_e\alpha_{\rm eff} \ , \ \textbf{for a SST situation} \quad . \tag{32b}$$

In an HF case, simulations can only be performed for a PT situation. In this case, the **anisotropic equation** (6) can be integrated after multiplication by $\gamma u/3$, to define the *time-dependent* electron drift velocity

$$v_d(t) \equiv \frac{\gamma}{3}\int_0^\infty u f^1(u,t)du = \Re\left[\frac{\gamma}{3}\int_0^\infty u\tilde{f}^1(u)e^{j\omega t}du\right] \equiv \Re[\mu_{\rm HF}E_p e^{j\omega t}] \quad .$$

Consistently, the parameters (32a)-(32b) are not calculated, and the user can obtain $v_d(t)$ from the HF mobility (29a)-(29b)

$$v_d(t) = \Re[\mu_{\rm HF}E_p e^{j\omega t}] = \frac{E_p}{2}\Re[\mu_{\rm HF}e^{j\omega t} + \mu^\star_{\rm HF}e^{-j\omega t}] = \{\Re[\mu_{\rm HF}N]\cos(\omega t) - \Im[\mu_{\rm HF}N]\ \sin(\omega t)\}\frac{\zeta E}{N} \quad .$$

For DC/HF simulations involving the **time-growth of the electron density**, the **isotropic equation** (7a) can be integrated after multiplication by γ, to define the reduced-effective frequency (in $\mathrm{m^3\ s^{-1}}$)

$$\frac{\langle\nu_{\rm eff}\rangle}{N} = \gamma\sum_{k,i}\delta_{k_i}\int_0^\infty \left[S_{k_i,\rm ion}(u) + S_{k_i,\rm att}(u)\right]du \quad , \tag{33}$$

[4] In `LoKI-B_v1.0.0`, the `swarmParameters` datafile in the output of HF simulations was writing the DC mobility, calculated from (26b) at $\omega = 0$, instead of the real part of the HF mobility. We acknowledge Ilija Stefanovic (Ruhr Universität Bochum, Germany) for pointing this out.

where

$$\nu_{\text{eff}} = \langle \nu_{\text{ion}} \rangle - \langle \nu_{\text{att}} \rangle \quad , \tag{34}$$

with

$$\begin{aligned} \langle \nu_{\text{ion}} \rangle &\equiv \sum_{k,i} \delta_{k_i} \langle \nu_{k_i,\text{ion}} \rangle \\ \langle \nu_{\text{att}} \rangle &\equiv \sum_{k,i} \delta_{k_i} \langle \nu_{k_i,\text{att}} \rangle \quad . \end{aligned}$$

In this case, equation (33) is solved iteratively until convergence of the parameter $\langle \nu_{\text{eff}} \rangle / N$ is achieved (see section 7.5).

For DC simulations involving the **spatial-growth of the electron density**, the **isotropic equation** (8a) can be integrated after multiplication by γ, to define the effective Townsend coefficient

$$\alpha_{\text{eff}} = \frac{\langle \nu_{\text{ion}} \rangle - \langle \nu_{\text{att}} \rangle}{v_d} = \frac{\langle \nu_{\text{eff}} \rangle}{v_d} \quad , \tag{35}$$

with

$$\frac{\alpha}{N} \equiv \frac{\langle \nu_{\text{ion}} \rangle / N}{v_d} \tag{36a}$$

$$\frac{\eta}{N} \equiv \frac{\langle \nu_{\text{att}} \rangle / N}{v_d} \tag{36b}$$

the first and the second reduced Townsend coefficients (in m^2) for ionisation and attachment, respectively. In this case, equations (32b) and (35) can be combined to yield [32]

$$D_e \alpha_{\text{eff}}^2 - \mu_e E \alpha_{\text{eff}} + \langle \nu_{\text{eff}} \rangle = 0 \quad , \tag{37}$$

which is solved iteratively until convergence of the parameter α_{eff} is achieved (see section 7.5).
Note that the initial EEDF guess may lead to a negative discriminant $\mu_e^2 E^2 - 4 D_e \langle \nu_{\text{eff}} \rangle$ of the quadratic equation (37), in which case, it can be calculated

- either assuming that there is no electron density gradient (see (32b)): $\alpha_{\text{eff}} = \langle \nu_{\text{eff}} \rangle / v_d = \langle \nu_{\text{eff}} \rangle / (\mu_e E)$;
- or assuming that the discriminant is zero (and using the expression for the root of the quadratic equation): $\alpha_{\text{eff}} = \mu_e E / (2 D_e) = 4 D_e \langle \nu_{\text{eff}} \rangle / (2 D_e \mu_e E) = 2 \langle \nu_{\text{eff}} \rangle / (\mu_e E)$.

Note further that the parameters (35)-(36b) are not calculated in the case of HF simulations.

2.2 Power-transfer terms

The balance equation for the electron power-density, per electron at unit gas density, is obtained as follows:

i) by writing the **isotropic component** of the EBE as a gain/loss equation, *i.e.*

- for the **steady-state EBE**:
  - when adopting a **temporal-growth model** (see equation (7a))

$$0 = -\frac{1}{\gamma \sqrt{u}} \frac{\langle \nu_{\text{eff}} \rangle}{N} u f(u) \Big|_n - \frac{1}{N} \frac{1}{\gamma} \frac{dG(u)}{du} \Big|_n + S_n \quad ; \tag{38a}$$

  - when adopting a **spatial-growth model** (see equation (8a))

$$0 = -\frac{1}{3} \frac{\alpha_{\text{eff}}}{N} u f^1(u) \Big|_n - \frac{1}{N} \frac{1}{\gamma} \frac{dG(u)}{du} \Big|_n + S_n \quad ; \tag{38b}$$

- for the **time-dependent EBE**:

– when adopting a **temporal-growth model** (see equation (19a))

$$\left.\frac{\partial f(u,t)}{\partial t}\right|_n = -\left.N\frac{\langle\nu_{\rm eff}\rangle}{N}f(u)\right|_n - \left.\frac{1}{\sqrt{u}}\frac{dG(u)}{du}\right|_n + N\gamma\sqrt{u}\frac{1}{u}S_n \quad ; \tag{38c}$$

– when adopting a **spatial-growth model**

$$\left.\frac{\partial f(u,t)}{\partial t}\right|_n = -\left.\frac{1}{3}\gamma\sqrt{u}N\frac{\alpha_{\rm eff}}{N}f^1(u)\right|_n - \left.\frac{1}{\sqrt{u}}\frac{dG(u)}{du}\right|_n + N\gamma\sqrt{u}\frac{1}{u}S_n \quad . \tag{38d}$$

ii) by multiplying either (38a) or (38b) by γu and integrating it over all energies; or by multiplying either (38c) or (38d) by $u^{3/2}/N$ and integrating it over all energies.

The general result writes (in eV s^{-1} m^3)

$$\frac{\Theta(t)}{N} = \frac{\Theta_{\rm growth}}{N} + \frac{\Theta_E}{N} + \frac{\Theta_{\rm coll}}{N} \quad , \tag{39a}$$

where the different terms on the right-hand side represent, in order, the variation of the power density due to non-conservative mechanisms inducing a time/space net growth of the electron density, the power density gained from the applied electric field [5], and the power density gained/lost in collisional ("coll") events, which can be separately written as (note that all "gain" terms are positively defined and all "loss" terms are negatively defined).

$$\frac{\Theta_{\rm coll}}{N} \equiv \frac{\Theta_{\rm coll}^{\rm gain}}{N} + \frac{\Theta_{\rm coll}^{\rm loss}}{N} \quad . \tag{39b}$$

The expressions for the various power-transfer terms are given below (see equations (12a)-(12d); see also section 2.1)

$$\frac{\Theta(t)}{N} = \begin{cases} 0 \ , \text{ for stationary and steady-state simulations} \\ \dfrac{1}{N}\dfrac{\partial\varepsilon(t)}{\partial t} \ , \text{ for time-dependent simulations} \end{cases} \tag{40a}$$

$$\frac{\Theta_{\rm growth}}{N} = \begin{cases} -\dfrac{\langle\nu_{\rm eff}\rangle}{N}\varepsilon \ , \text{ for a PT situation} \\ -\dfrac{\alpha_{\rm eff}}{N}\left(\mu_\varepsilon E - D_\varepsilon\alpha_{\rm eff}\right) \ , \text{ for a SST situation} \end{cases} \tag{40b}$$

$$\begin{aligned} \frac{\Theta_E}{N} &= -\frac{1}{N}\int_0^\infty u\frac{dG_E(u)}{du}du \\ &= \gamma\begin{cases} \displaystyle\int_0^\infty u\frac{d}{du}\left[g_E^{\rm PT}(u)\frac{df(u)}{du}\right]du\left(\frac{E}{N}\right)^2 \ , \text{ for a PT situation} \\ \displaystyle\int_0^\infty u\frac{d}{du}\left\{g_E^{\rm SST}(u)\left[\frac{E}{N}\frac{\alpha_{\rm eff}}{N}f(u) + \left(\frac{E}{N}\right)^2\frac{df(u)}{du}\right]\right\}du \ , \text{ for a SST situation} \end{cases} \end{aligned} \tag{40c}$$

$$\begin{aligned} \frac{\Theta_{x={\rm el,CAR}}}{N} &= -\frac{1}{N}\int_0^\infty u\frac{dG_x(u)}{du}du \\ &= \gamma\int_0^\infty u\frac{d}{du}\left\{g_x(u)\left[f(u) + \frac{k_BT_g}{e}\frac{df(u)}{du}\right]\right\}du \end{aligned} \tag{40d}$$

$$\frac{\Theta_{x={\rm el,CAR}}^{\rm gain}}{N} = \gamma\int_0^\infty u\frac{d}{du}\left[g_x(u)\frac{k_BT_g}{e}\frac{df(u)}{du}\right]du \tag{40e}$$

[5] Note that the electron power-density gained from the applied electric field corresponds to

$$\frac{\Theta_E}{N} = \begin{cases} \mu_e N\left(\frac{E}{N}\right)^2 \ , \text{ for DC simulations} \\ \Re[\mu_{\rm HF}N]\left(\frac{E}{N}\right)^2 \ , \text{ for HF simulations} \end{cases} .$$

$$\frac{\Theta^{\text{loss}}_{x=\text{el,CAR}}}{N} = \frac{\Theta_{x=\text{el,CAR}}}{N} - \frac{\Theta^{\text{gain}}_{x=\text{el,CAR}}}{N} \tag{40f}$$

$$\begin{aligned}\frac{\Theta_{ee}}{N} &= -\frac{1}{N}\int_0^\infty u\frac{dG_{ee}(u)}{du}du \\ &= \gamma\int_0^\infty u\frac{d}{du}\left\{g_{ee}(u)\left[I(u)f(u)+J(u)\frac{df(u)}{du}\right]\right\}du \end{aligned}\tag{40g}$$

$$\frac{\Theta^{\text{gain}}_{ee}}{N} = \gamma\int_0^\infty u\frac{d}{du}\left[g_{ee}(u)J(u)\frac{df(u)}{du}\right]du \tag{40h}$$

$$\frac{\Theta^{\text{loss}}_{ee}}{N} = \frac{\Theta_{ee}}{N} - \frac{\Theta^{\text{gain}}_{ee}}{N} \tag{40i}$$

$$\frac{\Theta^{\text{gain}}_{\text{sup}}}{N} = \gamma\sum_{i,j>i}\int_0^\infty S_{i,j}|_{\text{sup}}(u)udu = \sum_{i,j>i}\delta_j C_{j,i}V_{i,j} \tag{40j}$$

$$\frac{\Theta^{\text{loss}}_{\text{inel}}}{N} = \gamma\sum_{i,j>i}\int_0^\infty S_{i,j}|_{\text{inel}}(u)udu = -\sum_{i,j>i}\delta_i C_{i,j}V_{i,j} \tag{40k}$$

$$\frac{\Theta_{\text{ion}}}{N} = \gamma\sum_{i}\int_0^\infty S_{i,\text{ion}}(u)udu = -\sum_{i}\delta_i C_{i,\text{ion}}V_{i,\text{ion}} \tag{40l}$$

$$\frac{\Theta_{\text{att}}}{N} = \gamma\sum_{i}\int_0^\infty S_{i,\text{att}}(u)udu \tag{40m}$$

In these equations, the labels "inel" and "sup" refer to inelastic and superelastic collisions, respectively, associated with electronic, vibrational and rotational excitation / de-excitation mechanisms.

In time-dependent simulations the quantity $\Theta(t)/N$ is calculated by summing all the power terms on the right-hand side of (39a), and on output the value of this quantity is tagged as *Power balance* (see section 4.4).

In stationary and steady-state simulations, the code evaluates the relative power balance equation, dividing (39a) by the *reference power-density* (per electron at unit gas density) Θ_{ref}/N, with the latter defined as the sum of all power-gain terms.

As mentioned, LoKI-B can also adopt a generalized Maxwellian EEDF prescribed by the user (see section 1.4). In terms of output, the expressions in section 2.1 hold for all swarm parameters, but the power-transfer terms are taken in a slightly different way.

For the latter, the following applies

- the electron power density gained from the applied electric field is deduced from the (stationary) power balance equation, obtained from (39a)-(39b) at $\Theta(t)/N = \Theta_{\text{growth}}/N = 0$

$$\frac{\Theta_E}{N} = -\frac{\Theta_{\text{coll}}}{N} \equiv -\frac{\Theta^{\text{gain}}_{\text{coll}}}{N} - \frac{\Theta^{\text{loss}}_{\text{coll}}}{N} \quad , \tag{41a}$$

  with the $\Theta^{\text{gain}}_{\text{coll}}/N$ and $\Theta^{\text{loss}}_{\text{coll}}/N$ obtained from the different terms (40d)-(40l), calculated for the prescribed EEDF (23);

- the *equivalent* applied reduced electric-field is deduced from

$$\frac{E}{N} = \sqrt{\frac{-\Theta_{\text{coll}}/N}{(\Theta_E/N)/(E/N)^2}} \quad , \tag{41b}$$

  where the expression in the denominator corresponds to (40c) (for a PT situation) divided by the square

of the reduced electric-field [6];

- the electron power density transferred in collisional events is calculated assuming conservative mechanisms only, *i.e.* treating ionisation as an excitation mechanism and setting to zero the power-transfer due to attachment.
  For this reason, **selecting a prescribed EEDF for studying electronegative gases is not recommended.**

[6] In practice, the *equivalent* applied reduced electric-field is deduced from

$$\frac{E}{N} = \begin{cases} \sqrt{\frac{-\Theta_{\mathrm{coll}}/N}{\mu_e N}} & \text{, for DC simulations} \\ \sqrt{\frac{-\Theta_{\mathrm{coll}}/N}{\Re[\mu_{\mathrm{HF}} N]}} & \text{, for HF simulations} \end{cases} .$$

3 The "How to..." basics

This section provides basic information enabling a user to quickly obtain and run LoKI-B and LoKI-B++ for the first time.

3.1 Information about LoKI-B

3.1.1 How to obtain the code

You can obtain the latest version of the code from the github URL
https://github.com/LoKI-Suite/LoKI-B
by downloading the `LoKI-B_25.10` zip-file and extracting it to your local repository.

3.1.2 How to find my way in the code

The `LoKI-B_25.10` folder contains

1. A subfolder `Documentation` containing the user manual and relevant papers.
2. A subfolder `Code` containing
   (a) Several `'*.m'` files corresponding to the MATLAB® code, of which `'lokibcl.m'` is the main file.
   (b) A subfolder `Input`, containing the input files required for the simulations, organised as follows
      i. Default configuration files in text format `'default_lokib_setup.in'` and `'default_lokib_pulse_setup.in'` (see sections 3.1.3 and 4).
      ii. Default configuration files in JSON format `'default_lokib_setup.json'` and `'default_lokib_pulse_setup.json'` (see sections 3.1.3 and 4).
      iii. A subfolder `Databases` with `'*.txt'` files, containing different properties (masses, energies of states, atomic/molecular constants, ...) for the gases used in the simulations.
      iv. Several subfolders `Argon`, `CO`, ... `Nitrogen`, `Oxygen` with `'*.txt'` files containing
         - electron-scattering cross sections used in the simulations, usually obtained from the open-access website LXCat [26]; see section 5);
         - setup input files to run swarm simulations.

      Subfolders `Argon`, `Nitrogen`, `Oxygen` contain also `'*.json'` files with electron-scattering cross sections (obtained from the *demo* version of LXCat 3.0 [33]) and input setups for swarm simulations.
   (c) A subfolder `PropertyFunctions`, with several `'*.m'` auxiliary functions for calculating some predefined distribution of states (Boltzmann, Treanor, ...), the statistical weights of states due to their degeneracy, the energy of states according to some models, etc. (see section 6.1).
   (d) A subfolder `OtherAuxFunctions`, with several `'*.m'` auxiliary functions that calculate some working conditions, help to process or parse data, etc. (see section 6.2).
   (e) A subfolder `Output` (eventually), where LoKI-B will write the output files resulting from the simulations (see section 4.4).

3.1.3 How to run the code

LoKI-B is developed under MATLAB®, to benefit from its matrix-based architecture. The minimum requirements to run the code is a computer with an installation of MATLAB® (oldest recommended version R2017b; we cannot ensure that all the features of LoKI-B will work properly under different versions).

LoKI-B runs upon calling the MATLAB® function `lokibcl(setupFile)`. The end user interacts with the code by specifying a particular "setup" for the simulation. This setup is sent to the `lokibcl()` function through the *required* input argument `setupFile`. As mentioned in section 3.1.2, the setup files should be located in `[repository folder]/LoKI-B_25.10/Code/Input/` with

- `.in` extension (this is just a recommendation in order to keep the input folder organised; the `.in` setup files are just plain text files);
- `.json` extension (in this case it is important to use this extension, for the parsing to work properly).

The distribution of LoKI-B includes some default configuration files (`'default_lokib_setup.in'` and `'default_lokib_pulse_setup.in'`; `'default_lokib_setup.json'` and `'default_lokib_pulse_setup.json'`), for the benefit of the user. By using one of these files, it is very easy to make a first run of the code, just following the sequence of steps below.
For example, using the `'default_lokib_setup.in'` setup file:

1. Open MATLAB®
2. Navigate to the `Code` folder of your local copy of the repository:
   `>> cd [repository folder]/LoKI-B_25.10/Code/`
3. Execute the following command in the MATLAB® console:
   `>> lokibcl('default_lokib_setup.in')`
4. The graphical user interface (GUI) should appear showing the solution(s) for the default setup file, represented in figure 2.

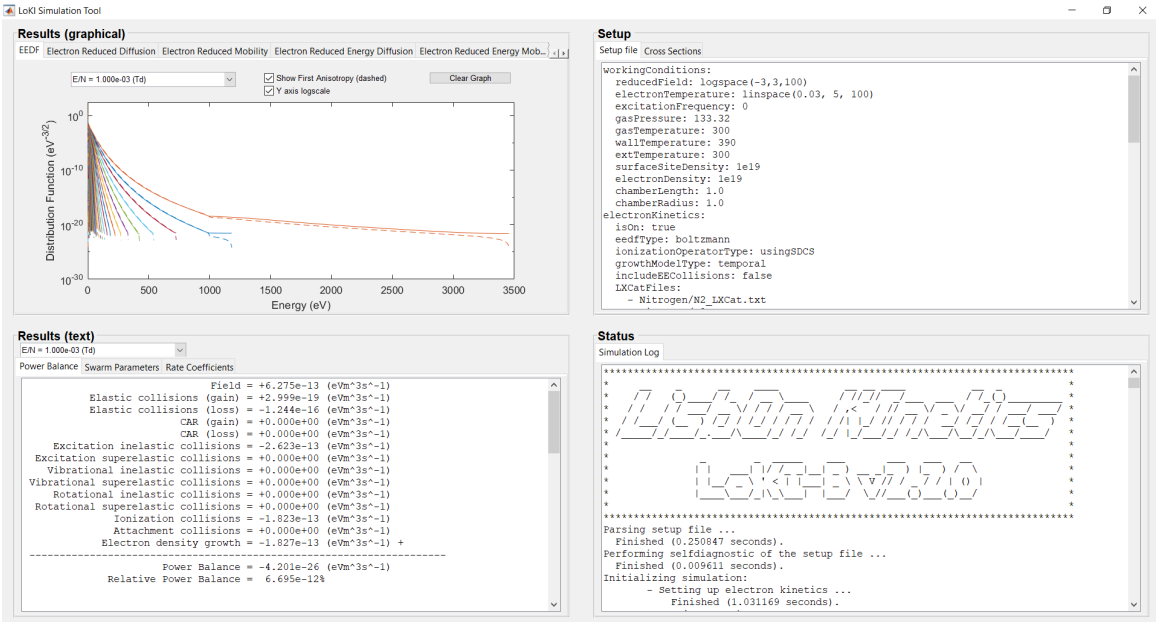


Figure 2: Graphical user interface output, for the default setup-file of LoKI-B (stationary simulations).

Congratulations! You have now concluded your first LoKI-B simulation.

Now, in order to run the code for different simulations, you just have to call the `lokibcl()` function with a different setup file, or a modified version of the default ones. The details on how to configure the setup file according to your needs are given in section 4.

3.2 Information about LoKI-B++

3.2.1 Prerequisites

To obtain, compile, and run LoKI-B++ you will need the following dependencies to be installed on your system,

- Git (https://git-scm.com/) to clone the repository,

- CMake (https://cmake.org/) and a suitable C/C++ compile for your operating system, i.e. GCC/G++ on Linux (https://gcc.gnu.org/), MSVC (https://visualstudio.microsoft.com/vs/features/cplusplus/) on Windows, and Apple Clang (https://developer.apple.com/xcode/cpp/) on Mac OS,
- LoKI-B uses Eigen (https://eigen.tuxfamily.org/) for linear algebra operations,
- `nlohmann-json` (https://github.com/nlohmann/json) for JSON handling,
- and `gnuplot` (http://gnuplot.info/) for plotting of results.

3.2.2 How to obtain the code

You can obtain the latest version of LoKI-B++ by cloning the "LoKI-B-cpp" github repository at https://github.com/LoKI-Suite/LoKI-B-cpp. This is done by issuing the command

```
git clone https://github.com/LoKI-Suite/LoKI-B-cpp.git
```

in a terminal. Alternatively, you can use a graphical user interface such as GitHub desktop (https://github.com/apps/desktop).

3.2.3 How to run the code

In general, to run C++ code you will first have to compile it. We provide multiple ways to do so depending on the operating system.

Running the code using Nix

The easiest way to run LoKI-B++ locally on Linux is to use the Nix package manager (https://nixos.org/). LoKI-B++ provides Nix expressions that take care of all the dependency management and building of the software. After installing the Nix package manager, you have to make sure to enable the `nix-command` and `flakes` experimental features. You can do this by setting `experimental-features nix-command flakes` in your `nix.conf` configuration file (https://nix.dev/manual/nix/2.18/command-ref/conf-file).

To run the default input file in LoKI-B++, you can now issue

```
nix run ./input/default_lokib_input.in
```

in the root of the repository. To plot the resulting EEDFs, you can pipe the output to `gnuplot`,

```
nix run ./input/default_lokib_input.in | gnuplot --persist
```

note that this does require `gnuplot` to be installed on your system. Alternatively, a development shell containing all the dependencies necessary to build and develop LoKI-B++ can be started by running

```
nix develop
```

in the root of the repository. Finally, you can also run LoKI-B++ from anywhere on your PC without the need to clone the Git repository using

```
nix run github:loki-suite/loki-b-cpp <path to input file>.
```

Standard compile and run on Linux

To compile LoKI-B++ on a machine running Linux, you can issue the following commands in the root of the repository.

1. Configure the build by running `cmake -DCMAKE_BUILD_TYPE=Release -D<BACKEND_FLAG>=ON -B build`. Here `<BACKEND_FLAG> = LOKIB_USE_MKL/LOKIB_USE_OPENBLAS` is a flag to set the backend to supply to Eigen, Intel MKL (https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-download.html), or OpenBLAS (http://www.openmathlib.org/OpenBLAS/), respectively. This flag can also be omitted to use pure Eigen routines.
2. To build the LoKI-B++ binary run `cmake --build build -j <NUM_JOBS> --target loki`, where `NUM_JOBS` is the maximum number of jobs to run simultaneously when compiling; just use the number of physical cores in your system. Omit this flag to use the default setting.

The binary will now be available at `./build/app/loki`. To run the default input file you can now run

```
./build/app/loki ./input/default_lokib_setup.in
```

in the root of the repository. To plot the resulting EEDFs, you can pipe the output to `gnuplot`,

```
./build/app/loki ./input/default_lokib_setup.in | gnuplot --persist
```

however this does require `gnuplot` to be installed on your system.

Standard compile and run on Windows

To compile LoKI-B++ on a machine running Windows, you can issue the following commands in the root of the repository.

1. Configure the build by running `cmake -D<BACKEND_FLAG>=ON -B build`.
   Here `<BACKEND_FLAG> = LOKIB_USE_MKL/LOKIB_USE_OPENBLAS` is a flag to set the backend to supply to Eigen, Intel MKL (https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-download.html), or OpenBLAS (http://www.openmathlib.org/OpenBLAS/), respectively. This flag can also be omitted to use pure Eigen routines.
2. To build the LoKI-B++ binary run `cmake --build build --config Release -j <NUM_JOBS> --target loki`, where `NUM_JOBS` is the maximum number of jobs to run simultaneously when compiling; just use the number of physical cores in your system. Omit this flag to use the default setting.

The binary will now be available at `.\build\app\loki.exe`. To run the default input file you can now run

```
.\build\app\loki.exe .\input\default_lokib_setup.in
```

in the root of the repository. To plot the resulting EEDFs, you can pipe the output to `gnuplot`,

```
.\build\app\loki.exe .\input\default_lokib_setup.in | gnuplot --persist
```

however this does require `gnuplot` to be installed on your system.

Using LoKI-B++ in the browser

LoKI-B++ can run natively in the browser by compilation to WebAssembly (https://webassembly.org/). A demo of the web deployment of LoKI-B++ is available at https://loki-suite.github.io/LoKI-Web/. It can be used to run LoKI-B++ in the browser on any operating system.

3.2.4 Code structure & testing

LoKI-B++ follows a standard structure for C++ project. The header files can be found in `./LoKI-B`, and the corresponding source files are in the `./source` directory. The unit and regression test files are in the `./tests` directory. The tests can be compiled by omitting the `loki` target in the build command, e.g.

```
cmake --build build -j<NUM_JOBS>
```

on Linux. You can then run the complete test suite using

```
ctest -test-dir ./build/tests.
```

3.3 How to contact us

After downloading LoKI-B / LoKI-B++, and especially if you intend to interact with us, you are invited to send a short message
to loki@tecnico.ulisboa.pt
with **subject** LoKI-B / LoKI-B++
just giving your **name** and **affiliation**.

3.4 How to reference the code

LoKI-B / LoKI-B++ are the result of the efforts of the Portuguese group N-Plasmas Reactive: Modeling and Engineering (N-PRiME) of *Instituto de Plasmas e Fusão Nuclear* with *Instituto Superior Técnico*, the *Departamento de Física* of *Facultad de Ciencias* with *Universidad de Córdoba* and the group Elementary Processes in Gas Discharges (EPG) of the Eindhoven University of Technology. These groups decided to share the outcome of its research on code development with the members of the Low-Temperature Plasmas community.

When using LoKI-B in your work, please give proper credits to the main developers, by adding the following citations

- Tejero A *et al The LisbOn KInetics Boltzmann solver* 2019 *Plasma Sources Sci. Technol.* **28** 043001.
- Tejero A *et al On the quasi-stationary approach to solve the electron Boltzmann equation in pulsed plasmas* 2021 *Plasma Sources Sci. Technol.* **30** 065008.

4 The setup file

The setup file serves as both the main input to LoKI-B and the primary user interface (UI) for interacting with the code. Setup files can be created either in text or JSON format.

In this file, the user can provide / select relevant information, namely

- about physical parameters
  - working conditions
  - collisional data
  - atomic and molecular data
  - species populations
- about physical models
  - mode of energy-sharing in ionisation collisions
  - type of growth model for the electron density
  - inclusion of electron-electron collisions
- about numerical details
  - energy grid for LoKI-B
  - nonlinear algorithm(s) to adopt in the simulations
  - convergence criteria for iterative cycles
  - integration times

The setup **text** file is an **indentation structured file**, *i.e.* it follows the "Off-side rule" [34] like in some programming languages such as Python. This means that it is **mandatory** that all the subfields within a field share the same level of indentation, otherwise the behaviour of the code is undefined.

The **setup text file** is organised according to the following syntax rules.

- The setup text file can include comments, signaled by the character `%`.
- Some of the fields in the setup text file can accept either one or more values.
  When only one value is to be specified, it can be given in the same line of the field, right after the colon, for example:

  ```
  mass: Databases/masses.txt
  ```

  Alternatively, when more than one value are to be specified, one can use several lines starting with a dash, for example

  ```
  LXCatFiles:            % cross section files
      - Nitrogen/N2_LXCat.txt
      - Nitrogen/N2_rot_LXCat.txt
  ```

- The setup text file can call either `*.txt` or `*.json` auxiliary input files.

The **setup JSON file** is organised similarly to the setup text file, following the standard JSON syntax rules.
The following should be noted:

- The calling of functions within the setup JSON file is passed with quotes "." .
- The setup JSON file can call either `*.txt` or `*.json` LXCat files.

Note that the value(s) of certain fields in the setup file can be defined specifying an "auxiliary input file" or a "property function".
All the "auxiliary input files" (*i.e.* LXCat files or property files) specified in the setup file should be located inside the input folder (`[repository folder]/LoKI-B_25.10/Code/Input`), eventually organised in subfolders with self-explanatory names (*e.g.* `Nitrogen`, `Databases`; see section 3.1.2). In this case, the name of the subfolder should be specified in the setup file (see examples above for `mass` and for `LXCatFiles`). For more information about auxiliary input files see section 5.
All the "property functions" specified in the setup file should be located in the property functions folder (`[repository folder]/LoKI-B_25.10/Code/Property Functions`; see section 3.1.2). For more information about property functions see section 6.1.

The setup file is organised into four different main fields, with self-explanatory designations:

- `workingConditions`
- `electronKinetics`
- `gui`
- `output`.

The next subsections present these fields in detail, using a **setup text file for nitrogen** as example.

4.1 Working conditions

In this field of the setup file the user can specify the values of the different working parameters for the simulation. Code 1 shows the section of the `'default_lokib_setup.in'` file that handles this configuration.

```
% --- configuration of the working conditions ---
workingConditions:
  reducedField:  logspace(-3,3,100)           % in Td
  electronTemperature: linspace(0.03, 5, 300) % in eV
  excitationFrequency: 0                      % in Hz
  gasPressure: 133.32                         % in Pa
  gasTemperature: 300                         % in K   (average gas temperature)
  wallTemperature: 390                        % in K   (wall temperature)
  extTemperature: 300                         % in K   (external temperature)
  surfaceSiteDensity: 1e19                    % in m-2 (used for surface kinetics)
  electronDensity: 1e19                       % in m-3
  chamberLength: 1.0                          % in m
  chamberRadius: 1.0                          % in m
%   dischargeCurrent: 2e-2                      % in A
%   dischargePowerDensity: 1e5                  % in W m-3
  totalSccmInFlow: 1                          % in sccm
  totalSccmOutFlow: ensureIsobaric            % a number (in sccm) or a model: totalSccmInFlow, ensureIsobaric
```

Code 1: Configuration of the working conditions of the simulation

IMPORTANT NOTES:

- The `reducedField` corresponds to E/N, where
  - $E = E_p$ is the amplitude of the applied electric field, in a DC case;
  - $E = E_p/\sqrt{2}$ is the root-mean-square (RMS) amplitude of the applied electric field, in a HF case.
- The `reducedField` can also contain information about a non-oscillatory ($\omega = 0$) electric-field pulse, in which case it reads

```
% --- configuration of the working conditions ---
workingConditions:
     reducedField: pulse@reducedFieldPulse, 1e-18, 2e-6, logspace, 100, 1e-6, 50
    % reducedField: pulse@reducedFieldRamp, 1e-18, 2e-6, logspace, 100, 1e-6, 25
    % pulse@functionName, firstStep, finalTime, samplingType, samplingPoints, additionalParameters
```

Code 2: Configuration of the reducedField for an electric-field pulse

where the field parameters are, in order,

- the name of the pulse function (providing E/N values in Td, as a function of time) The current distribution of LoKI-B includes the following pulse functions (see section 6.2):
  - `pulse@reducedFieldPulse`, corresponding to

$$\frac{E(t)}{N}(\text{Td}) = \frac{E_0}{N}\sqrt{\frac{t}{\tau_{\text{rise}}}}\,\exp\left(-\frac{t}{\tau_{\text{rise}}}\right) \quad . \tag{42}$$

  - `pulse@reducedFieldRamp`, corresponding to

$$\frac{E(t)}{N}\,(\text{Td}) = \begin{cases} \frac{E_0/N}{\tau_{\text{rise}}}t\,, & 0 \leq t \leq \tau_{\text{rise}} \\ \frac{E_0}{N}\,, & t > \tau_{\text{rise}} \end{cases} \quad . \tag{43}$$

- the first time step (in s) for the simulation output
- the final time (in s) for the simulations and the simulation output
- the type of sampling for the time steps used in the output (`logspace` or `linspace`)
- the number of sampling points in the simulation output
- any additional parameters (optional) assisting the construction of the pulse function (*e.g.* the values of variables E_0/N and τ_{rise} in equations (42) and (43)).

Note the following, **when adopting an electric-field pulse**:

- the `excitationFrequency` **MUST** be set to zero;
- the simulation runtimes can rapidly increase, especially for very long pulse-times, and/or high gas pressures, and/or a very large number of sampling points. User discretion is advised, when defining these parameters (the set of working conditions presented above for the `reducedField` is a good compromise example);
- users should be aware that keeping the graphical user interface activated (see section 4.3) can also degrade the simulation runtimes;
- it is recommended to adopt a constant energy grid (see section 4.2), chosen as to ensure the correct description of the EEDF for both the minimum $E(t)/N$ value in the simulations (considering low threshold mechanisms), and the maximum $E(t)/N$ value in the simulations (considering the bulk-to-tail drop in the EEDF);
- it is recommended to activate the optional parameters for the ode solver adopted in the `temporalIntegration` algorithm (see sections 4.2 and 7.5): absolute tolerance, relative tolerance, and maximum step-size. Users should understand the role of these parameters before activating them in the simulations.

- The `electronTemperature` corresponds to the electron temperature T_e.
- The user can choose simulations that solve the EBE or that adopt a generalized Maxwellian EEDF (see section 1.4 and section 4.2 for details on how to select the type of simulation for the EEDF):
  - the parameter `reducedField` is activated when the "boltzmann" simulation is selected;
  - the parameter `electronTemperature` is activated when the "prescribedEedf" simulation is selected.

  For the parameters `reducedField` and `electronTemperature`, it is possible to specify a single value, for a single simulation, or an array of values, for several simulations corresponding to the different values prescribed. The array of values can be defined using any expression understood by MATLAB® (*e.g.* `logspace(-4,1,100)`, `linspace(0,100,10)`, `[0 10 20 50]`, etc.), with the exception of those adopting the colon character ":" (*e.g.* `1:10` is not supported). In future releases of LoKI-B, this feature will probably be extended to other parameters.
- The `excitationFrequency` corresponds to the linear oscillation frequency $\omega/(2\pi)$.
- The `gasPressure` corresponds to the gas pressure p.
- The `gasTemperature` corresponds to the average gas temperature T_g.

- The `electronDensity` corresponds to the average electron density n_e.
- Not all the parameters in Code 1 are used in all the simulations. Indeed, the `gasTemperature` T_g is needed to calculate the elastic and the CAR operators (12b) and (12c), whereas the `gasPressure` p is required to calculate the total gas density only in the particular case of HF excitations (for obtaining the reduced frequency ω/N) and/or if electron-electron collisions are considered in the calculations (for obtaining the ionisation degree n_e/N, in which case it is also necessary to provide the `electronDensity` n_e). Moreover, the parameters `nearWallTemperature`, `wallTemperature`, `extTemperature`, `surfaceSiteDensity`, `chamberLength`, `chamberRadius`, `dischargeCurrent`, `dischargePowerDensity`, `totalSccmInFlow` and `totalSccmOutFlow` are not used in LoKI-B simulations.
  The user can comment the line(s) of Code 1 that specify the value(s) of parameter(s) not used in the simulations. Alternatively these lines can be kept uncommented, but in this case **it is mandatory to specify the values of the corresponding parameters**, regardless the fact that they are not used in the simulations (the values of these parameters will have no impact in the final results).
- Again, the code uses SI units for all physical quantities, except the energies that are expressed in eV (electron-volt) and the reduced fields that are expressed in Td (Townsend; 1 Td = 10^{-21} V m^2).

4.2 Electron kinetics

In this field of the setup file the user can define all the details related to the electron kinetics. Code 3 shows the section of the `'default_lokib_setup.in'` file that handles this configuration.

```
% --- configuration of the electron kinetics ---
electronKinetics:
  isOn: true             % true or false (to activate or deactivate the electron kinetics)
  eedfType: boltzmann    % boltzmann or prescribedEedf (generalized expression Maxwellian/Druyvesteyn)
%   shapeParameter: 1    % prescribedEedf shape parameter from 1 (Maxwellian) to 2 (Druyvesteyn)
  ionizationOperatorType: usingSDCS  % conservative, oneTakesAll, equalSharing or usingSDCS
  growthModelType: temporal          % temporal or spatial
  includeEECollisions: false         % true or false (to include / remove e-e collisions)
  LXCatFiles:                        % cross section files
    - Nitrogen/N2_LXCat.txt
    - Nitrogen/N2_rot_LXCat.txt
%  LXCatFilesExtra: extra_LXCat.txt  % extra cross section files
%  effectiveCrossSectionPopulations: % see doc
%    - Nitrogen/N2_effectivePop.txt
%  CARgases:                         % gases for which CAR is activated
%    - N2
  gasProperties:               % properties of the gases (S.I. Units)
    mass: Databases/masses.txt
    fraction:
      - N2 = 1
    harmonicFrequency: Databases/harmonicFrequencies.txt
    anharmonicFrequency: Databases/anharmonicFrequencies.txt
    rotationalConstant: Databases/rotationalConstants.txt
    electricQuadrupoleMoment: Databases/quadrupoleMoment.txt
    OPBParameter: Databases/OPBParameter.txt
  stateProperties:             % properties of the states (S.I. Units except for the energy [eV])
    energy:
      - N2(X,v=*) = harmonicOscillatorEnergy
      - N2(X,v=0,J=*) = rigidRotorEnergy
    statisticalWeight:
      - N2(X,v=*) = 1.0
      - N2(X,v=0,J=*) = rotationalDegeneracy_N2
    population:
      - N2(X) = 1.0
      - Nitrogen/N2_vibpop.txt
      - N2(X,v=0,J=*) = boltzmannPopulation@gasTemperature
  numerics:                    % configuration of numerical details of the simulation
    energyGrid:                % properties of the energy grid (in eV)
      maxEnergy: 1             % (use 18-20 for time-dependent simulations)
```

```
    cellNumber: 1000          % (use 1800-2000 for time-dependent simulations)
    smartGrid:              % configuration of the smart grid
      minEedfDecay: 20      % minimum number of decade-fall for the EEDF
      maxEedfDecay: 25      % maximum number of decade-fall for the EEDF
      updateFactor: 0.05    % factor used to increase or decrease the maximum value of the energy grid
  maxPowerBalanceRelError: 1e-9
  % threshold for the relative power balance warning message (use at least 100 for time-dependent simulations)
  nonLinearRoutines:
    algorithm: mixingDirectSolutions % mixingDirectSolutions or temporalIntegration
    mixingParameter: 0.7             % mixingDirectSolutions mixing parameter from 0 to 1
    maxEedfRelError: 1e-9 % maximum rel. variation for EEDF between two iterations (stop criterion)
%   odeSetParameters:       % optional parameters for ode solver of "temporalIntegration" algorithm
%     AbsTol: 1e-300
%     RelTol: 1e-6
%     MaxStep: 1e-7
```

Code 3: Configuration of the electron kinetics of the simulation

IMPORTANT NOTES:

- The `isOn` allows the user to activate (`true`) or deactivate (`false`) the simulations of LoKI-B.
- The `eedfType` allows the user to specify the type of simulation for the EEDF:
  - `boltzmann`, corresponding to the solution of the EBE under the classical two-term approximation;
  - `prescribedEedf`, corresponding to a prescribed generalised Maxwellian EEDF at given electron temperature T_e (see sections 1.4 and 4.1)
- The `shapeParameter` allows the user to specify the shape parameter s, when a `prescribedEedf` is selected (see section 1.4).
- The `ionizationOperatorType` allows the user to choose the type of ionisation operator with the EBE. The possible configurations are (see section 1.2):
  - `conservative`: the ionisation is treated as an excitation process;
  - `oneTakesAll`: the ionisation is treated as a non-conservative mechanism, with the new-born secondary electron introduced at zero kinetic-energy and the scattered electron taking the remaining energy after an ionisation event.
  - `equalSharing`: the ionisation is treated as a non-conservative mechanism, with the remaining energy after the event being equally shared between the scattered and the secondary electrons.
  - `usingSDCS`: the ionisation is treated as a non-conservative mechanism, described by considering a Single Differential Cross Section (SDCS) that depends on the energies of the primary and the secondary electrons.

    In general, the available experimental data for SDCS is scarce and often measured for only a few energy values of the primary electron. Here, we have adopted the comprehensive set of measurements of Opal *et al* [35], for the double differential cross section of different gases, resolved on both the energy and the angle of the secondary electron. This pioneering work proposes differential ionisation cross sections with a very reliable shape [36], which can be normalized from available experimental data for the corresponding integral cross sections (see (15)). Indeed, after integration over the angles, and by comparing the energy distributions of the secondary electrons for different energies of the primary electrons, these authors proposed a fitting expression for the SDCS

$$\sigma^{\text{sec}}_{i,\text{ion}}(u',u) = \frac{\sigma_{i,\text{ion}}(u')}{w\,\text{arctg}\left[\left(u'-V_{i,\text{ion}}\right)/(2w)\right]}\frac{1}{1+(u/w)^{\beta}} \quad , \tag{44}$$

    where $\beta \simeq 2$ and w is a parameter close to the ionisation threshold $V_{i,\text{ion}}$, which is set for each gas according to the original estimates [35]. Note that (44) satisfies exactly the expression (15) for the integral cross section $\sigma_{i,\text{ion}}$, and therefore it is used in LoKI-B to describe the energy distribution of the electrons involved in ionisation events, using available data and mitigating possible normalisation errors.

    The value of w is introduced via the `OPBParameter`, in the `gasProperties` section (see below), and

should be provided by the user if the choice of the `ionizationOperatorType` is `usingSDCS`. If no value is provided for this parameter, the code will take w for each state equal to the corresponding ionisation potential.

- The `growthModelType` allows the user to specify the configuration of the growth model, `spatial` or `temporal`, for the electron density (see section 1.2);

  The parameter `growthModelType` is **mandatory**, being used if the choice of the `ionizationOperatorType` is different than `conservative`, and/or in the case of electronegative gases (with non-conservative attachment mechanisms).
  Recall that, when choosing `prescribedEedf`, the electron power density transferred in collisional events is calculated assuming conservative mechanisms only, *i.e.* treating ionisation as an excitation mechanism and setting to zero the power-transfer due to attachment (see section 2.2). For this reason, **selecting a prescribed EEDF for studying electronegative gases is not recommended.**

  When choosing `growthModelType` equal to `spatial` the user **MUST** adopt a DC applied electric field (see section 2.1), hence `excitationFrequency` must be set equal to `0` in this case.

  Note that all non-conservative modes (`oneTakesAll`, `equalSharing`, `usingSDCS` and the presence of attachment collisions) activate non-linear operators for the creation / destruction of electrons, hence code performance is expected to decrease in these cases.

- The `includeEECollisions` allows the user to activate (`true`) or deactivate (`false`) the electron-electron collision operator in LoKI-B.

  The parameter `includeEECollisions` is **mandatory**.

  Note that the inclusion of electron-electron collisions activates a non-linear operator, hence code performance is expected to decrease in this case.

  When performing time-dependent simulations, it is possible to handle electron-electron collisions considering a time-dependent electron-density $n_e(t)$, calculated according to the solution of equation (116b) (see section 7.5), for an initial value n_0 defined by the parameter `electronDensity` in the `workingConditions`.
  The calculation of $n_e(t)$ is activated through the expression
  `boltzmann.eDensIsTimeDependent = true`
  in function `obtainTimeDependentSolution(boltzmann)` with the `Boltzmann.m` class. To deactivate this calculation the user should select the logical parameter `false`, instead.
  Note that the previous expression only activates the calculation of $n_e(t)$. To use the time-evolution of the electron density in the electron-electron collisions operator, the user must further activate electron-electron collisions by setting `includeEECollisions: true` in the setup file.

- In `LXCatFiles` the user can specify the text files (along with the corresponding path) containing the electron-scattering cross sections for the different gases used in the simulations, usually obtained from the open-access website LXCat [26]. From these files, the code obtains also information about all the gases and all the states that are targets and/or products in the electron collisions.

  Note that not every cross section datafile obtained from LXCat is accepted, since it must comply with certain format requirements for LoKI-B to retrieve the relevant data; see section 5.1 for more information.

- When `LXCatFilesExtra` is activated, the user can specify the text files (along with the corresponding path) containing information on "extra" electron-scattering cross sections. The "extra" refers to **cross sections that are not used in the solution to the EBE**, but only in the evaluation of the corresponding rate coefficients by integration over the EEDF.

  Usually, the "extra" files are obtained from the open-access website LXCat [26], provided they comply with certain format requirements for LoKI-B to retrieve the relevant data; see section 5.1 for more information.

- The `effectiveCrossSectionPopulations` allows the user to specify some particular populations to evaluate an "elastic" momentum-transfer cross section from an "effective" one. The populations can be provided through one or more "property files", located in the input folder or in any subfolder within it (see section 5.2).

For a gas k, the electron-neutral momentum-transfer cross section $\sigma_{k,c}(u)$ is constructed by weighting the contributions of the different types of collisional mechanisms to give

$$\sigma_{k,c}(u) = \sum_i \xi_{k_i} \sigma^{\text{el}}_{k_i,c}(u) + \sum_{i,j>i} \left[\xi_{k_i} \sigma_{k_{(i,j)},c}(u) + \xi_{k_j} \sigma_{k_{(j,i)},c}(u) \right] \quad , \tag{45}$$

where $\sigma^{\text{el}}_{k_i,c}$, $\sigma_{k_{(i,j)},c}$ and $\sigma_{k_{(j,i)},c}$ are the electron-neutral momentum-transfer cross sections for the elastic scattering of state k_i, the inelastic excitation $k_i \rightarrow k_j$, and the superelastic de-excitation $k_j \rightarrow k_i$, respectively. The latter cross sections, whose magnitudes are usually much smaller than those of $\sigma^{\text{el}}_{k_i,c}$, are often taken assuming an isotropic scattering (also due to the lack of data), in which case one can identify the momentum-transfer cross section with the integral cross section (integrated over all scattering angles), *i.e.* $\sigma_{k_{(i,j)},c} \simeq \sigma_{k_{(i,j)}}$. Also, the elastic momentum-transfer cross section is usually identified with that of the highly-populated ground-state of the gas. Therefore

$$\sigma_{k,c}(u) \simeq \sigma^{\text{el}}_{k,c}(u) + \sum_{i,j>i} \left[\xi_{k_i} \sigma_{k_{(i,j)}}(u) + \xi_{k_j} \sigma_{k_{(j,i)}}(u) \right] \quad . \tag{46}$$

Note that electron-neutral anisotropic scattering can indeed be handled in a two-term Boltzmann solver by using momentum-transfer cross sections in (11a)-(11b) and (45), **for mechanisms with low excitation-thresholds** (such as rotational excitations / deexcitations) [37]. The IST-Lisbon database on LXCat provides momentum-transfer cross sections for these mechanisms in the case of some gases (see section 5.1).

The solution of the EBE requires information on both the "elastic" and the "total" momentum-transfer cross sections for **all** the electronic target-states of the gas, but the way to obtain it depends on the momentum-transfer data available for each gas. If the "elastic" momentum-transfer cross section is available, equation (46) can be used directly to deduce the corresponding "total" $\sigma_{k,c}$, knowing the electron-scattering inelastic / superelastic cross sections for the transitions considered together with the relevant populations, to be self-consistently determined (if possible) within a kinetic model. Conversely, if the data available is for a "total" momentum-transfer cross section, which in databases is usually termed as "effective" $\sigma^{\text{eff}}_{k,c}$, then the elastic momentum-transfer cross section can be deduced from

$$\sigma^{\text{el}}_{k,c} = \sigma^{\text{eff}}_{k,c} - \sum_{i,j>i} \left[\xi^{\text{prescribed}}_{k_i} \sigma_{k_{(i,j)}} + \xi^{\text{prescribed}}_{k_j} \sigma_{k_{(j,i)}} \right] \quad , \tag{47}$$

where the populations $\xi^{\text{prescribed}}_{k_i}$ and $\xi^{\text{prescribed}}_{k_j}$ correspond to prescribed distributions for the vibrational states with the electronic ground-state of the gas and for the rotational states with the vibrational ground-state of the electronic ground-state of the gas. Ideally, the distributions should be coherent with the measured / estimated data of $\sigma^{\text{eff}}_{k,c}$, allowing to obtain a *unique* result (density independent) for the elastic cross section.

The prescription of populations $\xi^{\text{prescribed}}_{k_i,j}$ by the user is strongly discouraged and should be taken with extreme care, since the values of these populations are directly used in (47) without any checking or processing. In any case, this feature is to be used only when an "effective" momentum-transfer cross section is provided, instead of an "elastic". By default, LoKI-B adopts a Boltzmann distribution at 300 K for $\xi^{\text{prescribed}}_{k_i,j}$, but the user can modify this choice.

Note also the following:

- when the collisional data for a gas includes an electron-impact "effective" momentum-transfer cross section: : (i) it is assumed that it corresponds to an **electronic** momentum-transfer cross section; and (ii) the code evaluates an elastic momentum-transfer cross section and uses this **same** cross section for **all** the electronic target-states of the gas;
- when the collisional data for a gas includes an electron-impact "elastic" momentum-transfer cross section for a given state (electronic, vibrational or rotational), the user must provide **individual** elastic momentum-transfer cross sections for **each** target-sibling of that state.

  For example, if N_2(X,v=0-10) and N_2(A) are targets, the user can provide "elastic" momentum-transfer cross sections: (i) for the 11 vibrational states N_2(X,v=0-10) and for the electronic state N_2(A); or (ii) for the 2 electronic states N_2(X) and N_2(A). However, it cannot simply provide "elastic" momentum-transfer cross sections for N_2(X,v=0) and N_2(A).

- In `CARgases` the user can specify the gas or gases (indicated as a list of gas names) for which the continuous approximation for rotations (CAR) [17] should be activated in the simulations (see section 1.2). When the CAR is not to be used, this field must be removed or commented.

  Note that the CAR approach of (12c), to describe the effect of rotational inelastic/superelastic mechanisms, is valid only for some homonuclear diatomic molecules, such as N_2 and O_2 for which it was tested. In general, for rotations, the discrete approach of the collision operator (14a) should be adopted instead.

- In `gasProperties` the user can specify the relevant properties of the different gases used in the simulations, such as

  ```
  mass
  fraction
  harmonicFrequency
  anharmonicFrequency
  rotationalConstant
  electricQuadrupoleMoment
  OPBParameter
  ```

  - It is **mandatory** to specify the values of `mass` and `fraction`. The latter should add to 1, when summed over all gases used in the simulations.
  - `harmonicFrequency` and `anharmonicFrequency` are optional, and are used to evaluate the energy of the different vibrational states by assuming an harmonic / anharmonic oscillator model.
  - `rotationalConstant` is optional, and is used to evaluate the energy of the different rotational states by assuming a rigid-rotor model.
  - `rotationalConstant` and `electricQuadrupoleMoment` are optional, and are used when adopting the CAR approach for homonuclear diatomic molecules. In this case the code returns an error message if these parameters are not specified.
  - `OPBParameter` is optional and corresponds to the value of the w parameter to be used in (44), when adopting the `usingSDCS` mode for the ionisation operator.

  The values of the different gas properties can be specified using one of the following methods:

  - "Direct input" (see example below).

    ```
    N2 = 1
    ```

    When using the "direct input" method, properties must be set only for those gases used in the simulations, *i.e.* those listed in the specified LXCat files.

  - "Property file" (see example below; for more information about "property files" see section 5.2).

    ```
    mass: Databases/masses.txt
    ```

    When using the "property file" method, it is possible to specify the properties of gases that are not used in the simulations. LoKI-B handles input files as "databases", where it searches only for the information required.

- In `stateProperties` the user can specify the properties of the states of the different gases used in the simulations, in accordance with the information retrieved from the LXCat files. The properties available are

  ```
  energy
  statisticalWeight
  population
  ```

  - `energy` and `statisticalWeight` are optional, and can be used for the code to evaluate the `population` of some set of states, for example adopting a given distribution.
  - `statisticalWeight` is **mandatory** for those states appearing in the LXCat files as involved in second-kind collisions (mechanisms tagged with a double-arrow $\leftrightarrow$), in which case the Klein-Rosseland relation [22] is used to obtain the superelastic cross section from the corresponding inelastic.

In general, the `statisticalWeight` of a state corresponds to the product of the statistical weights g associated with the different partition functions (electronic, vibrational, rotation) of this state. For example, the statistical weight of states N_2(X,v=0,J) should be set as $g[N_2(X, v = 0, J)] = g_X \times g_{v=0} \times g_J = 1 \times 1 \times 3\left[1 + \frac{1}{2}\left(1 + (-1)^J\right)\right](2J + 1)$, where the rotational statistical weight g_J includes also the contribution of the hyperfine splitting resulting from the coupling between the electron and the nuclear spin. Note, however, that the statistical weights are used only to handle second-kind collisions (see (14a), (30b) and (78a)) and to define prescribed distributions of populations (e.g, see (54) and (58) in section (6.1)), where they always appear as ratios g_j/g_i with i, j corresponding to sibling states (i.e. both electronic, or both vibrational or both rotational). Therefore, the results are not changed if the `statisticalWeight` of a state is defined only for the partition function *intrinsic* to that state; for example, the statistical weight of *rotational* states N_2(X,v=0,J) can be defined simply as $g[N_2(X, v = 0, J)] = g_J = 3\left[1 + \frac{1}{2}\left(1 + (-1)^J\right)\right](2J + 1)$ (see also section (6.1)).

  - It is **mandatory** to specify the values of `population` for all states considered as **targets of electron-scattering mechanisms**, *e.g.* the vibrational distribution of an electronic ground-state.
    The user should **not** define populations for the states that are only involved in electron-scattering extra collisions (included in the `LXCatFilesExtra`), since the corresponding cross sections are not used in the calculation of the EEDF.
  - The values of the different populations should be properly normalized to 1, when summed over all sibling states.

  The values of the different properties of states can be specified using one of the following methods (note that the syntax supports the wildcard character `*`):

  - "Direct input" (see example below)

    ```
    N2(X,v=*) = 1.0
    ```

  - "Property functions" (see example below; for more information about "property functions" see section 6.1).

    ```
    N2(X,v=*) = harmonicOscillatorEnergy
    ```

    When using the "direct input" method or the "property functions" method, properties must be set only for those gases and states used in the simulations, *i.e.* those listed in the specified LXCat files.

  - "Property file" (see example below; for more information about "property files" see section 5.2).

    ```
    Nitrogen/N2_vibpop.txt
    ```

    When using the "property file" method, it is possible to specify the properties of gases and states that are not used in the simulations. LoKI-B handles input files as "databases", where it searches only for the information required.

- In `numerics` the user can specify the numerical details of the simulations. The configuration parameters available are

  ```
  energyGrid
  maxPowerBalanceRelError
  nonLinearRoutines
  ```

  - The `energyGrid` allows the user to specify the properties of the energy grid for simulations in LoKI-B (see section 7.1):
    * `maxEnergy`, corresponding to the value of $u_{\max}$ (in eV) (this is a **mandatory** parameter);
    * `cellNumber`, is an integer corresponding to the number of cells $\mathcal{N} > 0$ of the energy grid (this is a **mandatory** parameter);
    * `smartGrid`, activating a "smart grid" that will automatically adjust the maximum energy of the grid, as to ensure a user-prescribed number of decades in the fall of the EEDF, between f_1 and $f_{\mathcal{N}}$ (this is an optional parameter). This option is particularly relevant for simulations spanning different values of E/N or T_e. The configuration parameters are:

· `minEedfDecay`, is an integer corresponding to the minimum number of decade-fall for the EEDF;

· `maxEedfDecay`, is an integer corresponding to the maximum number of decade-fall for the EEDF (`maxEedfDecay` > `minEedfDecay`);

· `updateFactor`, corresponding to the factor by which the `maxEnergy` is increased or decreased, in the automatic adjustment of the energy grid (0 < `updateFactor` < 1).

Note that the "smart grid" is as "smart" as the user that control its properties. So, if you have doubts on how this works, it is better to ignore this feature. However, if you decide to activate the "smart grid", you should consider the following:

1. **When the smart grid is activated, the `maxEnergy` parameter is the initial value for the maximum energy of the grid. Please try to provide the best possible approximation for it, in order to avoid unnecessary iterations that will degrade the run time.**
2. **Increasing $u_{\max}$ significantly without a corresponding increase in the number of grid points may reduce calculation accuracy. Users should verify the grid convergence of the simulation results.**
3. **Do not specify very restrictive conditions for the EEDF, for example by setting `minEedfDecay` to 20 and `maxEedfDecay` to 21, since this condition can only be satisfied by imposing a very-small `updateFactor`, which will also degrade the run time. Typical values for the decade-fall are within 10 to 25, considering not only the use of double precision, but also the order of magnitude of the various quantities (kinetic energy, cross sections) multiplying the EEDF in the calculations.**
4. **Use a "reasonable" value for the `updateFactor` parameter. Very-small values will decrease the performance of the code, whereas very-large values will make extremely difficult for the code to satisfy the decade-fall imposed for the EEDF. Recommended values are usually between 0.05 and 0.2.**

– The `maxPowerBalanceRelError` allows the user to specify the threshold value for the error of the relative power balance, above which the code **returns a warning message**. The typical threshold value is $\simeq 10^{-9} - 10^{-8}$. (This is a **mandatory** parameter). Note that this parameter is not active in time-dependent simulations, in which case the "power balance" corresponds to the value of the intrinsic time-change of the electron mean energy, at unit gas density (see section 2.2).

– In `nonLinearRoutines` the user can specify several parameters for the configuration of the algorithms with the non-linear routines, dealing with the non-conservative collision operators (ionisation and attachment) and the electron-electron collisions operator (see section 7.5).

∗ `algorithm`: the user can choose the nonlinear algorithm to adopt in the simulations. (This is a **mandatory** parameter).

· `mixingDirectSolutions`, which uses two nested iterative algorithms, based on successive direct solutions.
This algorithm is recommended in general.

· `temporalIntegration`, which uses a relaxation method.
This algorithm is recommended only in the case of strong nonlinear effects (*e.g.* high electronegativity and/or high ionisation degrees).
This algorithm **MUST** be chosen for time-dependent simulations, *e.g.* when LoKI-B runs for electrons subjected to a pulsed electric-field.

∗ `mixingParameter`: the user **MUST** define the value (between 0 and 1) of a mixing parameter, when choosing the `mixingDirectSolutions` algorithm.
Note that the mixing parameter should not be zero, as the non-linear operators will not converge, in this case.

∗ `maxEedfRelError`: the user can define the stop criterion for the `nonLinearRoutines`, by defining the maximum value for the relative variations of the EEDF, between two consecutive iterations. The typical value is 10^{-9}. (This is a **mandatory** parameter). Note that this parameter is not active in time-dependent simulations, in which case the convergence criterion is defined by the relative tolerance `RelTol`, imposed as parameter of the `temporalIntegration` solver (see section 1.3).

* `odeSetParameters`: this is an optional field where the user can define specific options for the `temporalIntegration` solver. A detailed description of the different options can be found in the MATLAB® documentation [38]. Example of useful options are given below.
  - `AbsTol`: is the value of the absolute tolerance for the `temporalIntegration` algorithm. The typical value is 10^{-300}.
  - `RelTol`: is the value of the relative tolerance for the `temporalIntegration` algorithm. The typical value is 10^{-6}.
  - `MaxStep`: is the maximum value for the step of the advancing parameter with the `temporalIntegration` algorithm. The typical value is 10^{-7}.

4.3 Graphical user interface

In this field of the setup file the user can set the configuration of the Graphical User Interface (GUI). Code 4 shows the section of the `'default_lokib_setup.in'` file that handles this configuration.

```
% --- configuration of the grafical user interface ---
gui:
  isOn: true
  refreshFrequency: 1
```

Code 4: Configuration of the GUI of the simulation

IMPORTANT NOTES:

- The `isOn` parameter allows the user to activate (`true`) or deactivate (`false`) the GUI.
- In `refreshFrequency` the user can control the frequency with which the GUI is refreshed. The value of this parameter corresponds to the number of required simulations for the GUI to be updated, hence the larger this value the less frequently the GUI is updated.

  Note that a very frequent update of the GUI can considerably degrade the run time. Note further that the GUI is optimized for Full HD screen resolution. Adopting different resolutions may result in anomalous displays.

When running LoKI-B for time-independent simulations (DC or HF cases) with the GUI activated (`isOn: true`), the graphic interface appears as in figure 2.

- On the left-hand upper corner, for each value of E/N (or T_e) adopted in the simulations and depending on the selected tab, the GUI displays the **Results (graphical)** that can be found in the output files (see section 2), namely
  - in *EEDF*, the plot of the EEDF and the corresponding first anisotropy;
  - in *Electron Reduced Diffusion*, the plot of $D_e N$ as a function of E/N (or T_e);
  - in *Electron Reduced Mobility*, the plot of $\mu_e N$ (and also $\Re[\mu_{HF} N]$ and $-\Im[\mu_{HF} N]$ in a HF case), as a function of E/N (or T_e);
  - in *Electron Reduced Energy Diffusion*, the plot of $D_\varepsilon N$ as a function of E/N (or T_e);
  - in *Electron Reduced Energy Mobility*, the plot of $\mu_\varepsilon N$ as a function of E/N (or T_e);
  - in *Electron Energies*, the plot of $T_e \equiv (2/3)\varepsilon$ and u_k, as a function of E/N (or T_e);
  - in *Townsend Coefficient*, the plot of α/N as a function of E/N (or T_e);
  - in *Attachment Coefficient*, the plot of η/N as a function of E/N (or T_e);
  - in *Power Channels*, the plot of the electron energy transferred in different channels, relative to the reference power, $\Theta_x/\Theta_{\text{ref}}$, as a function of E/N (or T_e) (see sections 2.2 and 7.3.).

  In all the graphs (except for *EEDF* and *Power Channels*), the user can select/unselect a *X axis logscale* and/or a *Y axis logscale*.
- On the left-hand lower corner, for each value of E/N (or T_e) adopted in the simulations and depending on the selected tab, the GUI displays a summary of the **Results (text)** that can be found in the output files (see section 4.4), namely

  - in *Power balance*, details about the electron power balance;
  - in *Swarm Parameters*, a summary of the electron swarm parameters (the reduced-ionisation Townsend coefficient, the reduced-attachment Townsend coefficient, and the drift velocity are calculated and displayed only in a DC case);
  - in *Rate Coefficients*, a summary of the inelastic/superelastic electron rate coefficients, calculated using both the cross sections adopted for solving the EBE and the *extra* set of cross sections defined by the user as additional input data of LoKI-B (see section 4). It also displays the description of each mechanism.

- On the right-hand upper corner, depending on the selected tab, the GUI displays a summary of the input **Setup**, namely
  - in *Setup file*, the uncommented lines of the setup file;
  - in *Cross Sections*, the graphs of the cross sections uploaded for the simulations (including both those adopted for solving the EBE and the *extra* cross sections defined by the user as additional input data of LoKI-B). Here, the user can select in the appropriated tab which cross section(s) to display.
- On the right-hand lower corner, the GUI will display a summary of the **Status** of the *Simulation Log*, with information documenting events relevant to the code execution.

When running LoKI-B for time-dependent simulations (e.g. considering a time-dependent, non-oscillatory ($\omega = 0$) electric field) with the GUI activated (`isOn: true`), the graphic interface looks similar to that of figure 2. In this case

- the **Results (graphical)** and the **Results (text)** are displayed as a function of the time t (instead of E/N or T_e);
- the GUI displays also a *Pulse temporal integration progress bar* that depicts the progress of the simulation as a function of the *Integration time*, indicating also the current *Computational time*.

4.4 Output

In this field of the setup file the user can define the details of the simulation output. Code 5 shows the section of the `'default_lokib_setup.in'` file that handles this configuration.

```
% ---  configuration of the output files ---
output:
  isOn: false
  dataFormat: hdf5+txt           % txt or hdf5 or hdf5+txt
  folder: simulation_1
  dataSets:
%    - inputs
    - log
    - eedf
    - swarmParameters
    - rateCoefficients
    - powerBalance
    - lookUpTable
```

Code 5: Configuration of the output of the simulation

IMPORTANT NOTES:

- LoKI-B provides `log` information documenting events relevant to the code execution, that are displayed in the command window. This information can also be displayed in the GUI (if activated) and/or saved as text file and/or saved as hdf5 file, according to the `output` configuration (see below).
- The `isOn` parameter allows the user to activate (`true`) or deactivate (`false`) the output.

- The `dataFormat` parameter allows the user to select the output format: text files (`txt`), a single HDF5 file gathering all the information from the text files (`hdf5`), or both text and HDF5 files (`hdf5+txt`).

  Writing the output in HDF5 format is a new feature in `LoKI-B_25.10`. **In this case, the log and setup information are still written as text files.**

  MATLAB® has used HDF5 C library version 1.14.4.3 since release R2024b. Running older MATLAB versions may therefore prevent the use of the HDF5 format.

- In `folder` the user can specify the folder where the datasets of the simulation output will be saved. This is an optional parameter; if `folder` is removed, the code will generate a generic name for the output folder, with a time stamp.
  The output folder(s) will be located inside `[repository folder]/LoKI-B_25.10/Code/Output/`.

- When `isOn` is `true`, `dataSets` is a **mandatory** parameter, where the user can specify the information to be saved after each simulation.
  Note that in previous versions of LoKI-B this parameter was called `dataFiles`.

4.4.1 Datasets in text format

In text format (`txt`), the output is characterized by

- different E/N values, for `eedfType = boltzmann` and time-independent simulations. In this case the code generates as many output folders as the number of E/N values defined as `reducedField` in the `workingConditions`;
- different t values, for `eedfType = boltzmann` and time-dependent simulations. In this case the code generates as many output folders as the number of "sampling points" defined within `reducedField` in the `workingConditions`;
- different T_e values, for `eedfType = prescribedEedf`. In this case the code generates as many output folders as the number of T_e values defined as `electronTemperature` in the `workingConditions`.

The available options for `dataSets` are:

- `inputs`
  It copies all the input datafiles (*e.g* LXCat files; not functions) called from the setup file (and also an exact copy of the original setup file itself, including all commented / uncommented lines), using the corresponding structure of folders.
- `log`
  It writes the datafile `log.txt`, displaying the log information of the code execution.
- `eedf`
  It writes, for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e) simulation, the datafile `eedf.txt`, with three columns displaying the kinetic energy u, the EEDF f and the first anisotropy f^1 (only in the case of `boltzmann` simulations), respectively.
- `swarmParameters`
  It writes, for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e) simulation, the datafile `swarmParameters.txt`, displaying the values of the following quantities (see section 2.1):

  t (only for time-dependent simulations)
  E/N
  $D_e N$
  $\mu_e N$
  v_d (only in DC case)
  $\Re[\mu_{\mathrm{HF}} N] + \Im[\mu_{\mathrm{HF}} N]\, i$ (only in HF case)
  α/N (only in DC case)
  η/N (only in DC case)
  $D_\varepsilon N$
  $\mu_\varepsilon N$
  ε

u_k
$T_e \equiv (2/3)\varepsilon$.

– `rateCoefficients`
It writes, for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e) simulation, the datafile `rateCoefficients.txt` displaying information on rate coefficients (see section 2.1), organized into the following groups:

* e-Kinetics Rate Coefficients, including the inelastic / superelastic rate coefficients $C_{i,j}$ and $C_{j,i}$ calculated from the electron-impact cross sections between states i and $j > i$ provided as input data, as well as the threshold of the cross sections and the description of the reactions;
* e-Kinetics Extra Rate Coefficients, including the inelastic / superelastic rate coefficients $C_{i,j}$ and $C_{j,i}$ calculated from the electron-impact extra cross sections between states i and $j > i$ provided as input data, as well as the threshold of the cross sections and the description of the reactions.

– `powerBalance`
It writes, for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e) simulation, the datafile `powerBalance.txt`, displaying the values of the following electron power-density gains/losses, per electron at unit gas density (see section 2.2):

Θ_E/N
$\Theta_{\rm el}^{\rm gain}/N$, $\Theta_{\rm el}^{\rm loss}/N$, $\Theta_{\rm el}/N \equiv \Theta_{\rm el}^{\rm gain}/N + \Theta_{\rm el}^{\rm loss}/N$
$\Theta_{\rm CAR}^{\rm gain}/N$, $\Theta_{\rm CAR}^{\rm loss}/N$, $\Theta_{\rm CAR}/N \equiv \Theta_{\rm CAR}^{\rm gain}/N + \Theta_{\rm CAR}^{\rm loss}/N$
$\Theta_{\rm sup}^{\rm gain}/N$, $\Theta_{\rm inel}^{\rm loss}/N$, $\Theta_{\rm ele,vib,rot}/N \equiv \Theta_{\rm sup}^{\rm gain}/N + \Theta_{\rm inel}^{\rm loss}/N$ (for electronic, vibrational, rotational mechanisms)
$\Theta_{\rm ion}/N$
$\Theta_{\rm att}/N$
$\Theta_{\rm growth}/N$.

It also writes

the power balance $\Theta_{\rm growth}/N + \Theta_E/N + \Theta_{\rm coll}^{\rm gain}/N + \Theta_{\rm coll}^{\rm loss}/N$ (with "coll" referring to all collision types). Recall that, in time-dependent simulations, the "power balance" corresponds to the value of the intrinsic time-change of the electron mean energy, at unit gas density (see section 2.2);

the relative power balance (in %) $\left[\Theta_{\rm growth}/N + \Theta_E/N + \Theta_{\rm coll}^{\rm gain}/N + \Theta_{\rm coll}^{\rm loss}/N\right] / \left[\Theta_{\rm ref}/N\right] \times 100$. Recall that, in time-dependent simulations, the "relative power balance" corresponds to the value of the intrinsic time-change of the electron mean energy divided by the reference power-density per electron (see section 2.2);

It also writes, for each gas in the mixture, the gain/loss/net results for the electron power-density associated with the different types of collisions.

– `lookUpTable`
The `lookUpTable` is only written with `txt` output. It writes the following datasets

* `lookUpTableSwarm.txt`, containing a table with all the data in the file(s) `swarmParameters.txt` presented in columns, as a function of E/N, or t and E/N, or T_e (according to the type of simulations);
* `lookUpTableRateCoeff.txt`, containing a table with all the data in the file(s) `rateCoefficients.txt` presented in columns, as a function of E/N, or t and E/N, or T_e (according to the type of simulations);
* `lookUpTablePower.txt`, containing a table with all the data in the file(s) `powerBalance.txt` presented in columns, as a function of E/N, or t and E/N, or T_e (according to the type of simulations);
* `lookUpTableEedf.txt` **(only for time-dependent simulations)** containing a table with all the data in the file(s) `eedf.txt`, where the first column displays the values of t, the first row displays the values of u, and the remainder rows display the values of f for the different simulation times. Table 2 presents an extract of a typical `lookUpTableEedf.txt` file.
* `electronDensityEvolution.txt`, containing a table with the values of the electron density n_e and the electron density variation dn_e/dt presented in columns, as a function of t. This file is written only in the case of time-dependent simulations in the presence of electron-electron collisions, considering a time-dependent electron density $n_e(t)$ (see section 4.2).

Table 2: Extract of a typical `lookUpTableEedf.txt` file for a time-dependent simulation

	0.005	0.015	0.025	0.035	...
0.0000E+00	220.49	149.76	101.72	69.09	...
1.0000E-18	220.49	149.76	101.72	69.09	...
1.0233E-18	220.49	149.76	101.72	69.09	...
1.0471E-18	220.49	149.76	101.72	69.09	...
...	...	...	...	...	...

4.4.2 Datasets in HDF5 format

In HDF5 format (`hdf5`) [39], the output results are organized in a single file. The access to the data can be done through Python scripts, using one of the several interfaces available [40], or adopting filters on tools like MATLAB® or Origin®.
LoKI-B results are organized in a main `electronKinetics` group, under a root group ('/'). A *group* is conceptually equivalent to a folder with certain *attributes*, within which the *datasets* are organized. The attributes of the root group correspond to the relevant parameters of the working conditions (see figure 3).

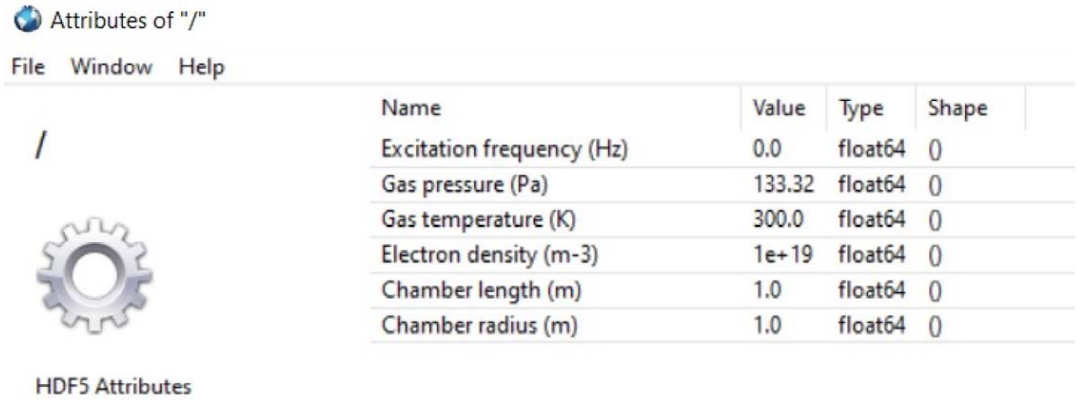


Figure 3: HDF5 attributes for LoKI-B simulations using the default setup-file.

The `electronKinetics` group contains six datasets:

- `reducedField`
  A table with the value(s) of the reduced electric field (or the corresponding electron temperature, see below). This table is linked to other datasets, corresponding to one of their axes. The shape and dimensions of the table depends on the problem under study.
  For `eedfType = boltzmann` and time-independent simulations, the table is composed by a single row with the different E/N values used in the simulations.
  For `eedfType = boltzmann` and time-dependent simulations, the table has two rows with the $E/N(t)$ and t values. In this case the code generates as many columns as the number of "sampling points" defined within `reducedField` in the `workingConditions`.
  For `eedfType = prescribedEedf`, the table is composed by a single row with the different T_e values used in the simulations. In this case the code generates as many columns as the number of T_e values defined as `electronTemperature` in the `workingConditions`.
  The units of each column are indicated as dataset attributes.
- `eedf`
  A 3d dataset, gathering the $\mathcal{N}$ values of the the kinetic energy u, the EEDF f and the first anisotropy f^1 (only in the case of `boltzmann` simulations), respectively, written in a three-column table (page)

for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e). The dataset's size is $N_{E/N} \times 3 \times \mathcal{N}$, with $N_{E/N}$ the number of E/N or T_e values considered in the output.
The units of each column are indicated in the dataset attributes.

– `swarmParameters`
A 2d table with the values of the swarm parameters, written for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e), in the following order:
ε, u_k, T_e or E/N, v_d, $\mu_e N$, $D_e N$, $\mu_\varepsilon N$, $D_\varepsilon N$, α/N, η/N.
T_e or E/N are written for `boltzmann` or `prescribedEedf` simulations, respectively.
The dataset's size is $10 \times N_{E/N}$ for a DC case or $8 \times N_{E/N}$ for an HF case, where v_d, α/N and η/N are not written.
The units of each column are indicated in the dataset attributes.

– `rateCoefficients`
A 3d dataset with information on the rate coefficients of the collisional processes considered in the electron kinetics, written in a table (page) for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e). The table has five columns displaying the process id, the inelastic rate coefficient, the corresponding superelastic rate coefficient, the cross section threshold, and the description of the collisional process. The dataset's size is $N_{\text{proc}} \times 5 \times N_{E/N}$, with N_{proc} the number of processes in the electron kinetics. The units of each column are indicated in the dataset attributes.

– `powerBalanceSummary`
A 3d dataset with the electron power-density gains/losses, per electron at unit gas density (see section 2.2), written for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e) in the following order (ten-column table):
`Field`, `Elastic`, `CAR`, `Rotational`, `Vibrational`, `Electronic`, `Ionisation`, `Attachment`, `eDensGrowth`, and `Balance`.
The dataset has three tables (pages) displaying, for each process (if applicable): the net power-density (gain+loss), the power-density gain, and the power-density loss, respectively. The power balance is displayed in only two pages, showing absolute and relative values, respectively. The dataset's size is $3 \times 10 \times N_{E/N}$. The units of each column are indicated in the dataset attributes.

– `powerBalanceGases`
A multidimensional dataset with the electron power-density gains/losses, per electron at unit gas density transferred to each gas in the mixture, written for each `boltzmann` (E/N or t) or `prescribedEedf` (T_e), for the following collisional processes (five columns):
`rotColl`, `vibColl`, `eleColl`, `ionColl`, and `attColl`.
The dataset has $N_{\text{gas}} \times 3$ tables (pages) displaying, for each gas (in a total of N_{gas}) and for each process (if applicable): the net power-density (gain+loss), the power-density gain, and the power-density loss, respectively.
The dataset's total size is $N_{\text{gas}} \times 3 \times 5 \times N_{E/N}$.

5 Auxiliary input files

This section gives detailed information on the configuration of the different auxiliary input files used by LoKI-B, namely on the notation to be adopted in the description of the species and the electron-scattering mechanisms involved in the simulations.

The auxiliary input files should be located in the input folder `[repository folder]/LoKI-B_25.10/Code/Input/` or in any subfolder within it. They are either plain text files or JSON files, although some can display extensions different than `*.txt` and `*.json`, for the sake of folder organisation.

5.1 LXCat files

By default, LoKI-B uses electron-scattering cross sections obtained from the LXCat open-access website [26], in the solution to the EBE or as *extra* data for integration over the calculated EEDF, to obtain electron macroscopic parameters. The cross sections can be **assembled into a single file or given from multiple files**, and if/when **appearing duplicated in the datafiles they are considered only once by LoKI-B**.

LXCat is organised to provide "data required for modeling low temperature plasmas" [26] in the most effective way. In the case of electron-neutral scattering cross sections, LXCat provides easy access to *complete sets* of cross sections, defined as those giving a good description of all electron energy and momentum losses [41], yielding electron swarm parameters in agreement with available measured data (within experimental uncertainties), when used in a two-term Boltzmann solver [27, 41]. Each *complete set* assembles cross sections for electron collisions with different neutral targets, usually associated with the electronic ground-state of each gas. To date, on LXCat these data are tagged with the name of the corresponding gas (*e.g.* `Ar`, `N2`, `O2`, `CO2`, $\cdots$) and are grouped under the category `Ground states`, where one can find cross sections for the collisions of electrons with the electronic / vibrational / rotational ground-states of a neutral gas. For example, the datasets `N2` may contain (i) the elastic / effective momentum-transfer cross section for N_2; (ii) excitation cross sections for different electronic states (including the ionisation state), from the electronic ground-state $N_2(X)$; (iii) excitation cross sections for different vibrational states $N_2(X,v)$, from the vibrational ground-state $N_2(X,v=0)$; (iv) a global rotational excitation cross section for $N_2(X)$.

This practical organisation of LXCat can create some difficulties when a detailed discrimination and handling of data is intended. The reason is of technical nature, and it relates with the fact that, to date, electron collisions with the `Ground state` of a gas are described as a collection of mechanisms `e + gas name ->` $\cdots$, with no possibility to specify the internal structure of the electronic ground-state of the gas. Meaning that the information, if any, about the different species involved in each electron collision is scattered across the LXCat files, depending on the strategy adopted by each particular LXCat contributor. Obviously, this makes impossible to develop a general parser to collect the full list of species (with internal structure), for any file downloaded from LXCat.

The various issues mentioned above, regarding the organisation of LXCat, as well as the detailed classification and handling of data, are being addressed by the platform's developers, as already demonstrated in the *demo* version of LXCat 3.0 [33].

Meanwhile users can modify any cross section **text-file** downloaded from LXCat, to make it compliant with the LoKI-B parser. For this, it is important to understand how LXCat structures the metadata associated with each cross section. In an LXCat text-file, each cross-section-table comes with a header including the following fields:

- `SPECIES`
- `PROCESS`
- `PARAM.`
- `COMMENT` (multiple lines)
- `UPDATED`
- `COLUMNS`.

In principle, LoKI-B should check the `PROCESS` field to obtain details on the full structure of reactants and products, for each electron-impact mechanism, but again this is not available in the current structure of LXCat metadata. To circumvent this limitation, the LoKI-B parser looks for this information in the first line of the free-field `COMMENT`, which is modified to meet the following structure:

```
COMMENT:␣[<electron>␣+␣<target state>␣<directionality>␣<products states>,␣<collision
                        type>,␣<collision subtype>] ...
```

where the square brackets and spaces are mandatory characters, and where the different elements in the expression have the following features:

- `electron:` can be represented by either characters `e` or `E` (see section 5.3 for more information about the description of the species involved in electron-scattering mechanisms).
- `target state:` can detail the electronic/vibrational/rotational state of the target species (see section 5.3 for more information about the description of the species involved in electron-scattering mechanisms).
- `directionality:` can be a forward arrow (`->`) for inelastic collisions, or a double arrow (`<->`) for inelastic/superelastic collisions.
- `product states:` sum of all the product(s) resulting from an electron collision, including electron(s) and heavy-species, eventually with indication of the electronic/vibrational/rotational level (see section 5.3 for more information about the description of the species involved in electron-scattering mechanisms).
- `collision type:`[7] one of the following collision types, supported by the LXCat database: `Elastic`, `Effective`, `Excitation`, `Vibrational`, `Rotational`, `Ionization` or `Attachment`. Note the **mandatory** capital letters, since the parser is case sensitive.
- `collision subtype:` (optional field) either empty for integral cross sections, or `momentum-transfer` for momentum-transfer cross section.
  Note the following exception: the collision types `Elastic` and `Effective` are automatically handled as `momentum-transfer`, even if the collision subtype if left empty.

The workaround described above is the practical solution to prepare any cross section **text-file** obtained from LXCat for compliance with the LoKI-B parser, and it is already implemented in the following files:

- the LXCat **text-files** distributed with the code, and located in `[repository folder]/LoKI-B_25.10/Code/Input/` (see section 3.1.2).

  Before modifying any other LXCat file, the user should check the text files distributed with the code, to become familiarised with the modifications needed.

- cross section **text-files**, **directly downloadable only from the IST-Lisbon database with the website LXCat, and to date only for the gases indicated below**[8]
  - Argon: `Ar`, for electron excitations from the ground-state Ar($1S_0$);
  - Helium: `He`, for electron excitations from the ground-state He(1^1S);
  - Nitrogen: `N2`, for electron excitations from the electronic ground-state $N_2(X)$ and the vibrational ground-state N_2(X,v=0); `N2-vib` for electron excitations from vibrational states N_2(X,v= $1-10$); `N2-rot` for electron excitations from rotational states N_2(X,v=0,J=0-30) $\Delta J = 2$.
    The latter two datagroups are clustered under `Nitrogen` in `State-specific and gas mixtures` at LXCat;
  - Atomic Nitrogen: `N`, for electron excitations from the ground-state N(^4S); `N-elec`, for electron excitations from the excited states N(^2D) and N(^2P).
    These datagroups are clustered under `Nitrogen` in `State-specific and gas mixtures` at LXCat;

[7] LXCat 3.0 will extend the collision types supported, (i) by adding: `MomentumTransfer`, `Electronic`, `Dissociative`, `Recombination`, `Dettachment` (to be introduced); and (ii) by allowing combinations between different collision types / subtypes, *e.g.* `Elastic MomentumTransfer`; `Electronic Excitation MomentumTransfer`.

[8] LoKI-B is distributed with only part of the data available on the IST-Lisbon database at LXCat, namely complete sets of cross sections for electron excitations from the ground-state of gases, of interest for swarm analyses.

- Oxygen: `O2`, for electron excitations from the electronic ground-state O_2(X) and the vibrational ground-state O_2(X,v=0); `O2-vib` for electron excitations from vibrational states O_2(X,v= 0 − 41); `O2-rot` for electron excitations from rotational states O_2(X,v=0,J=1-29) $\Delta J = 2$; `O2-elec` for electron excitations from excited electronic states of O_2, namely $O_2(a^1\Delta_g, b^1\Sigma_g^+)$.
  The latter three datagroups are clustered under `Oxygen` in `State-specific and gas mixtures` at LXCat;
- Atomic Oxygen: `O`, for electron excitations from the ground-state $O(^3P)$; `O-elec` for electron excitations from excited electronic states of O, namely O^-.
  These datagroups are clustered under `Oxygen` in `State-specific and gas mixtures` at LXCat;
- Ozone: `O3`, for electron excitations from the ground-state O_3(X).
  This datagroup is under `Oxygen` in `State-specific and gas mixtures` at LXCat;
- Hydrogen: `H2`, for electron excitations from the electronic ground-state H_2(X) and the vibrational ground-state H_2(X,v=0); `H2-vib` for electron excitations from vibrational states H_2(X,v= 1 − 15); `H2-rot` for electron excitations from rotational states H_2(X,v=0,J=0-5) $\Delta J = 2$; `H2-elec` for electron recombination of ionic states H_2^+ and H_3^+.
  The latter three datagroups are clustered under `Hydrogen` in `State-specific and gas mixtures` at LXCat;
- Atomic Hydrogen: `H`, for electron excitations from the ground-state H(1s); `H-elec`, for electron excitations from the excited states H(2s), H(2p), H(3), H(4) and H(5).
  These datagroups are clustered under `Hydrogen` in `State-specific and gas mixtures` at LXCat;
- Carbon dioxide: `CO2`, for electron excitations from the electronic ground-state CO_2(X) and the vibrational ground-state CO_2(X,v=000);
- Carbon monoxide: `CO`, for electron excitations from the electronic ground-state CO(X) and the vibrational ground-state CO(X,v=0); `CO-rot` for electron excitations from rotational states CO(X,v=0,J=0-17) $\Delta J = 1$;
- Carbon monoxide (with anisotropic scattering)[9]: `CO_anis`, for electron excitations from the electronic ground-state CO(X) and the vibrational ground-state CO(X,v=0), to be adopted when anisotropic scattering for rotational collisions is also considered; `CO_dipint-rot` delivering dipole-integral cross sections for electron excitations from rotational states CO(X,v=0,J=0-25) $\Delta J = 1$; `CO_dipmt-rot` delivering dipole-momentum-transfer cross sections for electron excitations from rotational states CO(X,v=0,J=0-25) $\Delta J = 1$; `CO_quadint-rot` delivering quadrupole-integral cross sections for electron excitations from rotational states CO(X,v=0,J=0-24) $\Delta J = 2$.
  These datagroups are clustered under `CO_anis` in `State-specific and gas mixtures` at LXCat.

In the folder `[repository folder]/LoKI-B_25.10/Code/OtherAuxFunctions` the user can find the script `LXCat2LoKI.m` that interactively assists the modification of LXCat **text-files**, for compliance with the LoKI-B parser. Code 6 shows an example of a modified header, for a cross-section-table extracted from the `N2_LXCat.txt` file distributed with LoKI-B.

```
EXCITATION
N2 -> N2 (v=0 - v=1)
 3.000000e-1
SPECIES: e / N2
PROCESS: E + N2 -> E + N2 (v=0 - v=1), Excitation
PARAM.:  E = 0.3 eV, complete set
COMMENT: [e + N2(X,v=0) <-> e + N2(X,v=1), Vibrational] Pitchford L C and Phelps A V 1982
COMMENT: Bull. Am. Phys. Soc. 27 109 Tachibana K and Phelps A V 1979 JCP 71 3544 Schulz G J 1962
COMMENT: Phys. Rev. 125 229 Schulz G J 1964 Phys. Rev. 135 A988 Schulz G J 1973 Rev. Mod. Phys.
COMMENT: 45 423 Engelhardt A G, Phelps A V and Risk C G 1964 Phys. Rev. 135 A1566 Pavlovic Z,
COMMENT: Boness M J W, Herzenberg A and Schulz G J 1972 Phys. Rev. A 6 676.
UPDATED: 2017-09-03 10:54:40
COLUMNS: Energy (eV) | Cross section (m2)
-----------------------------
 3.000000e-1  0.000000e+0
 4.000000e-1  3.000000e-23
 5.000000e-1  4.000000e-23
```

[9] This complete set of cross sections is the result of the work of L. Vialetto *et al* [37], which proposes a model of anisotropic scattering for rotational collisions, to be used in a two-term Boltzmann solver.

```
 ...
 9.653000e+2   1.238000e-24
 9.654000e+2   0.000000e+0
 1.000000e+3   0.000000e+0
-----------------------------
```

Code 6: Extract from the `N2_LXCat.txt` file distributed with LoKI-B (cf the modified `COMMENT` field)

LXCat 3.0 enables downloading cross-section files in both text and JSON formats, both of which are now supported by `LoKI-B_25.10`. The LXCat **JSON-files** are fully compatible with the LoKI-B parser, which no longer uses the information in the first line of the free-field `COMMENT`, reading instead the following fields:

- `state`, `reversible` and `typeTags`, within `processes > reaction`;
- `threshold` and `values > data`, within `processes > info`.

Currently, collision type `Electronic` (supported by LXCat 3.0) is automatically converted by `LoKI-B_25.10` into `Excitation`, as supported by current version of LXCat. Work is in progress to consolidate the transcription of LXCat data between versions 2.0 and 3.0, and the corresponding data parsing in numerical codes.

5.2 Property files

Property files are text files with the properties of gases and states used in the simulations. Some property files are distributed with LoKI-B, but in general these files can be prepared by the user, considering the following rules:

- for organisation sake, all "property files" specified in the setup file should be located inside the input folder (`[repository folder]/LoKI-B_25.10/Code/Input`), eventually organised in subfolders with self-explanatory names (*e.g.* `Nitrogen`, `Databases`; see section 3.1.2);
- property files can include comments, signaled by the character `%`;
- property files **MUST** be two-column-formatted, separated by any white-space character.
  The first column should include the gas/state name (see section 5.3 for more information about the description of the species used in the simulations), and the second column should include the value of the property for that particular gas/state;
- property files accept basic mathematical operations (such as addition, subtraction, multiplication and division), in the second column with the value of the property (see Code 7 as an example);
- property files can include the properties of gases/states not used in the simulations. LoKI-B handles input files as "databases", where it searches only for the information required.

Codes 7 and 8 show two examples of property files distributed with LoKI-B.

```
% masses of different gases expressed in SI units (Kg)
N3    3*14.007*1.660539040e-27
N4    4*14.007*1.660539040e-27
N2    2*14.007*1.660539040e-27
N     14.007*1.660539040e-27
O2    2*15.999*1.660539040e-27
O     15.999*1.660539040e-27
O3    3*15.999*1.660539040e-27
H2    2*1.008*1.660539040e-27
H     1.008*1.660539040e-27
CO2   (12.0107+2*15.999)*1.660539040e-27
CO    (12.0107+15.999)*1.660539040e-27
Ar    39.948*1.660539040e-27
He    4.002602*1.660539040e-27
```

```
He2   8.005204*1.660539040e-27
H2O   (2*1.008+15.999)*1.660539040e-27
CH4   16.0427*1.660539040e-27
C     12.0107*1.660539040e-27
```

Code 7: Property file with the masses of different gases (`Databases/masses.txt`)

```
% populations of the different vibrational states of N2
N2(X,v=0)   1
N2(X,v=1)   0
N2(X,v=2)   0
N2(X,v=3)   0
N2(X,v=4)   0
N2(X,v=5)   0
N2(X,v=6)   0
N2(X,v=7)   0
N2(X,v=8)   0
N2(X,v=9)   0
N2(X,v=10)  0
```

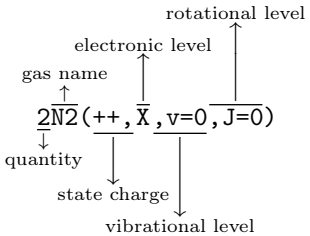
Code 8: Property file with a vibrational distribution of N_2 (`Nitrogen/N2_vibpop.txt`)

5.3 Species description

This section describes the ontology, including syntax rules, used in LXCat [26] for representing the different species in LoKI-B simulations. Note that LoKI-B now supports the new ontology adopted in the *demo* version of LXCat 3.0 [33] for describing species and reactions when using JSON files; however, the parsing scheme is expected to continue evolving. Work is in progress to consolidate the transcription of LXCat data between versions 2.0 and 3.0, and the corresponding data parsing in numerical codes.

LoKI-B handles two different types of species: electrons and heavy species (atomic or molecular). All species are represented by a **string without any white-space characters** and when specified in an electron collision **they are separated by plus signs** (`+`).
The representation of these species follows the syntax rules below:

- **Electrons:** can be represented by either characters `e` or `E`, which can appear multiplied by an integer, *e.g.* `2e` which is equivalent to `e + e`.
- **Heavy species:** refer to the different atomic or molecular species used in the simulations. **For electron-collision target species, it is mandatory to provide information on its configuration (see below)**.
  The scheme below shows the general structure to adopt, when representing the configuration of a particular species:



  where

  - quantity: is an **optional** field composed only by integers. It represents the number of species to be considered.
  - gas name: is a **mandatory** field composed by letters (English alphabet), integers and underscore characters (`\_`). It can not start with an integer and is case sensitive.
  - state charge: is an **optional** field composed by plus (`+`) and minus (`-`) characters. It represents the charge of an ionic state and has to terminate with a comma (`,`).

- electronic level: is a **mandatory** field composed by letters (English alphabet), integers, underscore characters (_), plus characters (+), minus characters (-), asterisk characters (*), apostrophe characters (’), square brackets ([]) and periods (.).
- vibrational level: is an **optional** field composed by letters (English alphabet), integers, underscore characters (_), plus characters (+), minus characters (-) and asterisk characters (*). It has to start with the string "`,v=`".
- rotational level: is an **optional** field composed by letters (English alphabet), integers, underscore characters (_), plus characters (+), minus characters (-) and asterisk characters (*). It has to start with the string "`,J=`" and it must necessarily come after the vibrational level field.

IMPORTANT NOTE:

The asterisk character ($\star$) is supported **only** in the setup file, **not in the LXCat auxiliary input files**.

6 Auxiliary functions

This section describes several auxiliary functions that can be used in the simulations of LoKI-B.

6.1 Property functions

Property functions are `*.m` functions that can be used to specify the properties of gases and states used in the simulations. Some property functions are distributed with LoKI-B, but in general these functions can be prepared by the user, according to the needs for new functionalities. In this case, the user is advised to mimic the structure of any property function already provided with the code.

The following should be considered:

- for organisation sake, all "property functions" specified in the setup file should be located in the property functions folder (`[repository folder]/LoKI-B_25.10/Code/PropertyFunctions`; see section 3.1.2);
- property functions receive the arguments (`state`, `argumentArray`, `workCond`), where
  - `state`: is a handle to the state-object to which the property is assigned.
    For example, when using `N2(X,v=*)` the property is assigned to all the vibrational states with the electronic ground-state of N_2;
  - `argumentArray`: is an array of arguments corresponding to the list of parameters to be used in the setup file (see below);
  - `workCond`: is a handle to the working-conditions-object of the simulation.
- property functions are called from the setup file (see section 4) using the following structure:

  `<function name>@[list of parameters]`
  (`@[list of parameters]`: Optional)

  where the `list of parameters` is a list of comma-separated values with all the parameters called by the particular property function. The parameters sent to the property functions can be mapped to the values of the different working conditions previously defined in the setup file (see Code 3 for examples, and therein the implementation of property function `boltzmannPopulation@gasTemperature`);
- property functions **MUST** be set only for those gases and states used in the simulations, *i.e.* those listed in the specified LXCat files.

Table 3 lists the property functions distributed with `LoKI-B_25.10`, along with the required parameters and dependencies.

The property functions in table 3 correspond to the following physical models.

`harmonicOscillatorEnergy`
It calculates the energy of (vibrational) state j of a diatomic molecule, adopting the harmonic oscillator model

$$E_j = \frac{\hbar\omega_{\text{vib}}}{e}\left(j+\frac{1}{2}\right) \quad , \tag{48}$$

where ω_{vib} is the `harmonicFrequency` (angular frequency) of the molecule.

`morseOscillatorEnergy`
It calculates the energy of (vibrational) state j of a diatomic molecule, adopting the Morse oscillator model

$$E_j = \frac{\hbar\omega_{vib}}{e}\left(j+\frac{1}{2}\right) - \frac{\hbar\omega_{\text{vib}}\,\chi_e}{e}\left(j+\frac{1}{2}\right)^2 \quad , \tag{49}$$

where $\omega_{\text{vib}}\,\chi_e$ is the first `anharmonicFrequency` of the molecule (with χ_e the anharmonicity constant).

Table 3: List of property functions distributed with LoKI-B_25.10

Property	Function name	Parameters [units]	Dependencies
energy	harmonicOscillatorEnergy	n/a	harmonicFrequency
	morseOscillatorEnergy	n/a	harmonicFrequency anharmonicFrequency
	rigidRotorEnergy	n/a	rotationalConstant
statisticalWeight	rotationalDegeneracy	n/a	n/a
	rotationalDegeneracy_N2	n/a	n/a
	rotationalDegeneracy_H2	n/a	n/a
population	boltzmannPopulation	Temperature [K]	energy statisticalWeight
	boltzmannRotationalPopulation_H2	Temperature [K]	energy statisticalWeight
	boltzmannPopulationRotationalCutoff	Temperature [K] J_{MAX}	energy statisticalWeight
	boltzmannPopulationVibrationalCutoff	Temperature [K] v_{MAX}	energy statisticalWeight
	treanorPopulation	Temperature0 [K] Temperature1 [K]	energy statisticalWeight
	treanorGordietsPopulation	Temperature0 [K] Temperature1 [K]	energy statisticalWeight anharmonicFrequency

`rigidRotorEnergy`

It calculates the energy of (rotational) state J of a diatomic molecule, adopting the rigid-rotor model

$$E_J = BJ(J+1) \quad , \tag{50}$$

where B is the rotational constant of the molecule.

These property functions should be called within the `energy` property of the setup file, for the corresponding vibrational and rotational states.

For example:

`N2(X,v=*) = harmonicOscillatorEnergy`

to set the energies of all the vibrational states with the electronic ground-state of N_2.

`N2(X,v=0,J=*) = rigidRotorEnergy`

to set the energies of all the rotational states of the vibrational state v=0, with the electronic ground-state of N_2.

`rotationalDegeneracy`

It calculates the statistical weight of the *intrinsic* partition function of a (generic) rotational state J [17]

$$g_J = 2J + 1 \quad . \tag{51}$$

`rotationalDegeneracy_N2`

It calculates the statistical weight of the *intrinsic* partition function of a rotational state J of the N_2 molecule [17]

$$g_J = 3\left[1 + \frac{1}{2}\left(1 + (-1)^J\right)\right](2J+1) \quad . \tag{52}$$

`rotationalDegeneracy_H2`

It calculates the statistical weight of the *intrinsic* partition function of a rotational state J of the H_2 molecule [17]

$$g_J = \left[2 - (-1)^J\right](2J+1) \quad . \tag{53}$$

These property functions should be called within the `statisticalWeight` property of the setup file, for the corresponding rotational states (see also section 4.2).
For example:
`N2(X,v=0,J=*) = rotationalDegeneracy_N2`
to set the statistical weights of all the rotational states of the vibrational state v=0, with the electronic ground-state of N_2.

`boltzmannPopulation`
It calculates the population of state j, adopting the Boltzmann distribution

$$\xi_j = \frac{g_j \, \exp\left(-\frac{E_j}{k_B T}\right)}{\sum_{i \in \text{siblings}} g_i \, \exp\left(-\frac{E_i}{k_B T}\right)} \quad , \tag{54}$$

where T is the characteristic temperature of the distribution.

`boltzmannRotationalPopulation_H2`
It calculates the population of rotational state j of H_2, adopting Boltzmann distributions for the orto/para configurations, with weights 25% and 75%, respectively

$$\xi_{j_{\text{even}}} = 0.25 \frac{g_{j_{\text{even}}} \, \exp\left(-\frac{E_{j_{\text{even}}}}{k_B T}\right)}{\sum_{i \in \text{siblings}} g_{i_{\text{even}}} \, \exp\left(-\frac{E_{i_{\text{even}}}}{k_B T}\right)} \tag{55a}$$

$$\xi_{j_{\text{odd}}} = 0.75 \frac{g_{j_{\text{odd}}} \, \exp\left(-\frac{E_{j_{\text{odd}}}}{k_B T}\right)}{\sum_{i \in \text{siblings}} g_{i_{\text{odd}}} \, \exp\left(-\frac{E_{i_{\text{odd}}}}{k_B T}\right)} \quad , \tag{55b}$$

where T is the characteristic temperature of the distribution.

`boltzmannPopulationRotationalCutoff`
It calculates the population of rotational state J, adopting a Boltzmann distribution truncated at level J_{MAX}

$$\xi_J = \frac{g_J \, \exp\left(-\frac{E_J}{k_B T}\right)}{\sum_{J'=0}^{J_{\text{MAX}}} g_{J'} \, \exp\left(-\frac{E_{J'}}{k_B T}\right)} \quad , \tag{56}$$

where T is the characteristic temperature of the distribution.
This distribution allows defining the populations of rotational states to be considered in the solution to the electron Boltzmann calculation, irrespectively of the full list of states involved in electron collision events (for example, appearing also in extra cross section files).

`boltzmannPopulationVibrationallCutoff`
It calculates the population of vibrational state v, adopting a Boltzmann distribution truncated at level v_{MAX}

$$\xi_v = \frac{g_v \, \exp\left(-\frac{E_v}{k_B T}\right)}{\sum_{v'=0}^{v_{\text{MAX}}} g_{v'} \, \exp\left(-\frac{E_{v'}}{k_B T}\right)} \quad , \tag{57}$$

where T is the characteristic temperature of the distribution.
This distribution allows defining the populations of vibrational states to be considered in the solution to the electron Boltzmann calculation, irrespectively of the full list of states involved in electron collision events (for example, appearing also in extra cross section files).

`treanorPopulation`
It calculates the population of state j, adopting the Treanor distribution [42, 43]

$$\xi_j = \frac{g_j \, \exp\left(\frac{j(E_1-E_0)-(E_j-E_0)}{k_B T_0}\right) \, \exp\left(-\frac{j(E_1-E_0)}{k_B T_1}\right)}{\sum_{i \in \text{siblings}} g_i \, \exp\left(\frac{i(E_1-E_0)-(E_i-E_0)}{k_B T_0}\right) \, \exp\left(-\frac{i(E_1-E_0)}{k_B T_1}\right)} \quad , \tag{58}$$

where T_0 and T_1 are the characteristic temperatures of the distribution.

`treanorGordietsPopulation`

It calculates the population of state j, adopting the Treanor-Gordiets distribution [42, 43]

$$\xi_j = \begin{cases} \dfrac{g_j \exp\left(\frac{j(E_1-E_0)-(E_j-E_0)}{k_B T_0}\right) \exp\left(-\frac{j(E_1-E_0)}{k_B T_1}\right)}{\sum_{i\in\text{siblings}} \xi_i} & , j \leq j^\star \\ \dfrac{\xi_{j^\star} \frac{j^\star}{j}}{\sum_{i\in\text{siblings}} \xi_i} & , j > j^\star \end{cases} , \tag{59}$$

where T_0 and T_1 are the characteristic temperatures of the distribution and

$$j^\star \equiv \frac{1}{2}\left[1 + \frac{E_1 - E_0}{\hbar\omega_{\text{vib}}\chi_e}\frac{T_0}{T_1}\right]$$

is the vibrational level corresponding to the minimum of the Treanor distribution (with $\omega_{\text{vib}}\chi_e$ the anharmonic frequency of the oscillator characterizing the distribution).

These property functions should be called within the `population` property of the setup file, for the corresponding states, using

`boltzmannPopulation@<temperature>`
`boltzmannPopulationRotationalCutoff@<temperature>,<J`$_{\text{MAX}}$`>`
`treanorPopulation@<temperature0>,<temperature1>`
where `temperature,temperature0,temperature1` can be temperature values or the dynamic variables `gasTemperature` or `electronTemperature`.
For example
`N2(X,v=*) = treanorPopulation@gasTemperature,4000`
to set the population of all the vibrational states with the electronic ground-state of N_2;
`N2(X,v=0,J=*) = boltzmannPopulation@gasTemperature`
to set the population of all the rotational states of the vibrational state v=0, with the electronic ground-state of N_2.

6.2 Other auxiliary functions

In OtherAuxFunctions the user can find `*.m` functions

- for interactively assist the modification of LXCat **text-files**, for compliance with the LoKI-B parser. In the current distribution of LoKI-B, this function is `LXCat2LoKI.m`;
- for plotting two different EEDFs in the same MATLAB® graph. In the current distribution of LoKI-B, this function is `compareEEDFs.m`;
- for calculating working conditions that can evolve during the simulations. In the current distribution of LoKI-B, this functionality is for calculating the `reducedField` in the case of step electric fields (function `reducedFieldRamp.m`) or electric field pulses (function `reducedFieldPulse.m`).

In general these functions can be prepared by the user, according to the needs for new functionalities. In this case, the user is advised to mimic the structure of any auxiliary function already provided with the code.

7 Numerical solution

The following sections provide essential information on the numerical solution of LoKI-B. They cover the discretisation of the EBE, the calculation and discretisation of the electron particle-balance and power-balance equations, and the solution methods employed to solve the EBE.

This material is intended primarily for developers, as much of the content is too technical or detailed for the majority of users.

7.1 Discretisation of the EBE

The EBE is numerically solved in an uniform energy grid defined in the energy interval $[0, u_{\max}]$, with *cells*

$$u_n^{\text{cell}} \equiv u_n = (n-1/2)\Delta u, \;\; n = 1, 2, \cdots \mathcal{N} \tag{60}$$

of fixed step-size $\Delta u = u_{\max}/\mathcal{N}$, each cell n being limited by *nodes* $n-1/2$ and $n+1/2$, each node n having energy

$$u_n^{\text{node}} = u_{n-1/2} = (n-1)\Delta u, \;\; n = 1, 2, \cdots \mathcal{N}+1 \tag{61}$$

Figure 4 presents a graphic representation of the energy grid as defined in (60)-(61). The user can define the

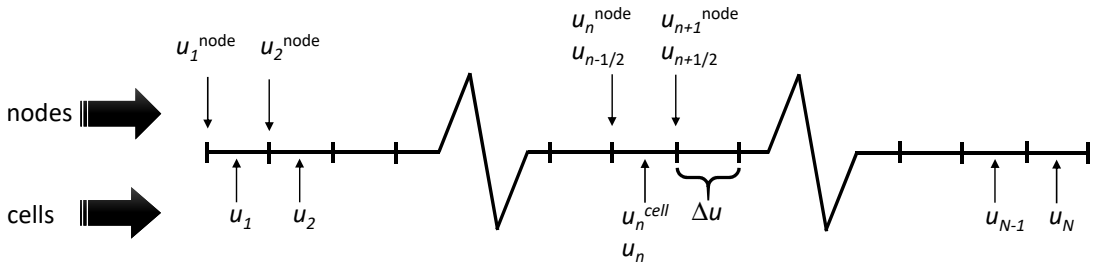


Figure 4: Graphic representation of the energy grid as defined in (60)-(61).

energy grid in two ways: (i) by setting the maximum energy $u_{\max}$ and the number of cells $\mathcal{N}$; (ii) by prescribing a number of decades in the fall of the EEDF, between f_1 and $f_{\mathcal{N}}$, as to ensure an automatic adjustment of the energy grid. Typical values for the decade-fall are within 10 to 25, considering not only the use of double precision, but also the order of magnitude of the various quantities (kinetic energy, cross sections) multiplying the EEDF in the calculations. The user is responsible for adequately defining the energy grid, namely by noticing that small energy steps are required to resolve any relevant structure in the cross sections, such as near-threshold variations and resonances, as well as situations involving the presence of a Ramsauer minimum. A significant increase in $u_{\max}$ without a corresponding increase in the number of grid points can reduce calculation accuracy. Therefore, the user should always verify the grid convergence of their simulation results.

The EBE equations (7a)-(8b) are written for each **grid-cell**, where the isotropic and anisotropic functions f and f^1 are defined, with the corresponding upfluxes G_x ($x = E, \text{el}, \text{CAR}, ee$) and cross sections σ_y ($y = k, c; k_{(i,j)}$) being set on the **grid-nodes**. The cross sections are interpolated linearly from the data provided by the user, setting to zero all the values up to the corresponding energy threshold, *i.e.* $\sigma_{y_n} = 0$ for $u_n^{\text{node}} \leq V_{i,j} \Longrightarrow n \leq$ `INT` $(V_{i,j}/\Delta u) + 1$.

The EBE is discretised adopting the centered finite differences scheme presented in [44]. The result is a system of algebraic equations that can be written under matrix form, with tridiagonal elements coming from the left-hand sides of (7a) and (8a), and some sparse off-tridiagonal elements due to the electron-neutral collisional operator in the right-hand sides.

The emphasis of the discretisation is with the definition of the upflux functions $G(u)$ on the grid-nodes, i.e. on the general boundaries $u_{n\pm1/2}^{\text{cell}}$ of a cell n centered at energy u_n^{cell}. The discretisation method can be described as a finite-volume-like procedure, by applying Gauss' theorem to an integral over a one-dimensional energy volume around u_n^{cell} as follows

$$-\frac{1}{N}\frac{1}{\gamma}\left.\frac{dG_x(u)}{du}\right|_{n;\, x=E,\text{el},\text{CAR}} = -\frac{1}{N}\frac{1}{\gamma}\frac{1}{\Delta u}\int_{u_{n-1/2}}^{u_{n+1/2}}\frac{dG_x(u)}{du}du = -\frac{1}{N}\frac{1}{\gamma}\frac{G_{x_{n+1/2}} - G_{x_{n-1/2}}}{\Delta u} \quad , \tag{62a}$$

and therefore (see equations (13a)-(13d))

$$
\begin{aligned}
-\frac{1}{N}\frac{1}{\gamma}\left.\frac{dG_x(u)}{du}\right|_{n;\,x=E,\mathrm{el,CAR}} &= \left\{\frac{d}{du}\left[g_x(u)\left(d_x f(u)+c_x\frac{df(u)}{du}\right)\right]\right\}_n \\
&\simeq \frac{g_{x_{n+1/2}}}{\Delta u}\left(\frac{d_x}{\Delta u}\int_{u_n^{\mathrm{cell}}}^{u_{n+1}^{\mathrm{cell}}} f(u)du+\frac{c_x}{\Delta u}\int_{u_n^{\mathrm{cell}}}^{u_{n+1}^{\mathrm{cell}}}\frac{df(u)}{du}du\right) \\
&- \frac{g_{x_{n-1/2}}}{\Delta u}\left(\frac{d_x}{\Delta u}\int_{u_{n-1}^{\mathrm{cell}}}^{u_n^{\mathrm{cell}}} f(u)du+\frac{c_x}{\Delta u}\int_{u_{n-1}^{\mathrm{cell}}}^{u_n^{\mathrm{cell}}}\frac{df(u)}{du}du\right) \\
&\simeq \frac{g_{x_{n+1/2}}}{\Delta u}\left(\frac{d_x}{2}(f_n+f_{n+1})+\frac{c_x}{\Delta u}(f_{n+1}-f_n)\right) \\
&- \frac{g_{x_{n-1/2}}}{\Delta u}\left(\frac{d_x}{2}(f_{n-1}+f_n)+\frac{c_x}{\Delta u}(f_n-f_{n-1})\right) \\
&= \left[\frac{g_{x_{n-1/2}}}{\Delta u}\left(\frac{c_x}{\Delta u}-\frac{d_x}{2}\right)\right]f_{n-1} \\
&- \left[\frac{g_{x_{n+1/2}}}{\Delta u}\left(\frac{c_x}{\Delta u}-\frac{d_x}{2}\right)+\frac{g_{x_{n-1/2}}}{\Delta u}\left(\frac{c_x}{\Delta u}+\frac{d_x}{2}\right)\right]f_n \\
&+ \left[\frac{g_{x_{n+1/2}}}{\Delta u}\left(\frac{c_x}{\Delta u}+\frac{d_x}{2}\right)\right]f_{n+1} \quad , \qquad (62b)
\end{aligned}
$$

where c_x and d_x are constants that can be obtained straightforwardly from (12a)-(12c), and

$$
\begin{aligned}
f_n &\equiv f(u_n^{\mathrm{cell}}) \\
g_{x_{n+1/2}} &\equiv g_x(u_{n+1/2}^{\mathrm{cell}}) \equiv g_x(u_{n+1}^{\mathrm{node}}) \quad .
\end{aligned}
$$

In addition to the terms proportional to $dG_{x=E,\mathrm{el,CAR}}/du$, equations (7a)-(8b) are further composed by terms proportional to f, df/du and dG_{ee}/du whose discretised forms can be generally summarised as follows (the latter receiving a treatment similar to equations (62a)-(62b); see also equations (12d) and (13e))

$$
\begin{aligned}
[a(u)f(u)]_n &= a_n f_n & (63a)\\
[a'(u)f(u\pm V_{i,j})]_n &= a'_n f_{n\,\pm\,\mathrm{INT}(V_i,j/\Delta u)} & (63b)\\
\left[b(u)\frac{df(u)}{du}\right]_n &= b_n\frac{f_{n+1}-f_{n-1}}{2\Delta u} & (63c)\\
\left.-\frac{1}{N}\frac{1}{\gamma}\frac{dG_{ee}(u)}{du}\right|_n &= \left\{\frac{d}{du}\left[g_{ee}(u)\left(I(u)f(u)+J(u)\frac{df(u)}{du}\right)\right]\right\}_n \quad , & (63d)
\end{aligned}
$$

where $a(u)$, $a'(u)$ and $b(u)$ are generic functions of u; and $I(u)$ and $J(u)$ are the relevant Spitzer integrals [12,21] for the electron-electron collision operator.

The detailed discretized form of each term is given below[10], with the cross sections defined on the grid-nodes as

$$
\begin{aligned}
\sigma_{y_n} &\equiv \sigma_y(u_n^{\mathrm{node}}) \\
\sigma_y(u_n^{\mathrm{cell}}) &= \frac{1}{2}\left(\sigma_{y_n}+\sigma_{y_{n+1}}\right) \quad .
\end{aligned}
$$

- **Time-dependent term (only for time-dependent simulations, see section 1.3)**

$$
\left.\frac{1}{N}\frac{1}{\gamma}\sqrt{u}\frac{\partial f(u,t)}{\partial t}\right|_n = \frac{1}{N}\frac{1}{\gamma}u_n^{1/2}\frac{\partial f_n(t)}{\partial t} \quad . \qquad (64)
$$

- **Temporal growth term**

$$
\left.\frac{1}{\gamma\sqrt{u}}\frac{\langle\nu_{\mathrm{eff}}\rangle}{N}uf(u)\right|_n = \frac{1}{\gamma}\frac{\langle\nu_{\mathrm{eff}}\rangle}{N}u_n^{1/2}f_n \quad . \qquad (65)
$$

[10] The discretized expressions aim to replicate as closely as possible those employed in the code's numerical implementation.

- **Spatial growth term (see equation (13b))**[11].

$$\begin{aligned}\frac{1}{3}\frac{\alpha_{\text{eff}}}{N}uf^1(u)\Big|_n &\simeq -\frac{\alpha_{\text{eff}}}{N}\left\{\frac{\alpha_{\text{eff}}}{N}\left(\frac{u}{3\Omega_{\text{SST}}(u)}f(u)\right)_n\right.\\ &\left.+\frac{E}{N}\frac{1}{2}\left[\left(\frac{u}{3\Omega_{\text{SST}}(u)}\frac{df(u)}{du}\right)^{\text{node}}_{n+1}+\left(\frac{u}{3\Omega_{\text{SST}}(u)}\frac{df(u)}{du}\right)^{\text{node}}_{n}\right]\right\}\\ &= -\frac{\alpha_{\text{eff}}}{N}\left\{\frac{\alpha_{\text{eff}}}{N}D_nf_n+\frac{E}{N}\left[D^{\text{node}}_{n+1}\frac{f_{n+1}-f_n}{2\Delta u}+D^{\text{node}}_{n}\frac{f_n-f_{n-1}}{2\Delta u}\right]\right\}\\ &= \left[\frac{\alpha_{\text{eff}}}{N}\frac{E}{N}\frac{D_{n-1/2}}{2\Delta u}\right]f_{n-1}\\ &-\left[\left(\frac{\alpha_{\text{eff}}}{N}\right)^2D_n+\frac{\alpha_{\text{eff}}}{N}\frac{E}{N}\left(\frac{D_{n-1/2}}{2\Delta u}-\frac{D_{n+1/2}}{2\Delta u}\right)\right]f_n\\ &-\left[\frac{\alpha_{\text{eff}}}{N}\frac{E}{N}\frac{D_{n+1/2}}{2\Delta u}\right]f_{n+1}\end{aligned}\quad , \tag{66}$$

with (see equation (13b))

$$D_n = \frac{u_n}{\frac{3}{2}\left[\Omega_{\text{SST}_n}+\Omega_{\text{SST}_{n+1}}\right]} \tag{67a}$$

$$D^{\text{node}}_n = D_{n-1/2}=\frac{u_{n-1/2}}{3\Omega_{\text{SST}_n}} \quad . \tag{67b}$$

- **Electric field operator (see equations (12a) and (13a)-(13b))**

$$g_E^{x=\text{PT,SST}}(u) = \frac{u}{3\Omega_x(u)} \tag{68a}$$

$$c_E = \left(\frac{E}{N}\right)^2 \tag{68b}$$

$$d_E = \begin{cases} 0 \ , \text{ for a PT situation} \\ \dfrac{\alpha_{\text{eff}}}{N}\dfrac{E}{N} \ , \text{ for a SST situation} \end{cases} \tag{68c}$$

For a PT situation

$$-\frac{1}{N}\frac{1}{\gamma}\left.\frac{dG_E^{\text{PT}}(u)}{du}\right|_n = \left(\frac{E}{N}\right)^2\left\{\left[\frac{g^{\text{PT}}_{E_{n-1/2}}}{(\Delta u)^2}\right]f_{n-1}-\left[\frac{g^{\text{PT}}_{E_{n-1/2}}+g^{\text{PT}}_{E_{n+1/2}}}{(\Delta u)^2}\right]f_n+\left[\frac{g^{\text{PT}}_{E_{n+1/2}}}{(\Delta u)^2}\right]f_{n+1}\right\} \quad . \tag{69a}$$

For an SST situation

$$\begin{aligned}-\frac{1}{N}\frac{1}{\gamma}\left.\frac{dG_E^{\text{SST}}(u)}{du}\right|_n &= \left(\frac{E}{N}\right)^2\left\{\left[\frac{g^{\text{SST}}_{E_{n-1/2}}}{(\Delta u)^2}\right]f_{n-1}-\left[\frac{g^{\text{SST}}_{E_{n-1/2}}+g^{\text{SST}}_{E_{n+1/2}}}{(\Delta u)^2}\right]f_n+\left[\frac{g^{\text{SST}}_{E_{n+1/2}}}{(\Delta u)^2}\right]f_{n+1}\right\}\\ &+\frac{\alpha_{\text{eff}}}{N}\frac{E}{N}\left\{-\left[\frac{g^{\text{SST}}_{E_{n-1/2}}}{2\Delta u}\right]f_{n-1}-\left[\frac{g^{\text{SST}}_{E_{n-1/2}}-g^{\text{SST}}_{E_{n+1/2}}}{2\Delta u}\right]f_n+\left[\frac{g^{\text{SST}}_{E_{n+1/2}}}{2\Delta u}\right]f_{n+1}\right\}\end{aligned} \quad . \tag{69b}$$

- **Elastic operator**

$$g_{\text{el}}(u) = 2u^2\sum_k\frac{m_e}{M_k}\chi_k\sigma^{\text{el}}_{k,c}(u) \tag{70a}$$

$$c_{\text{el}} = \frac{k_BT_g}{e} \tag{70b}$$

$$d_{\text{el}} = 1 \tag{70c}$$

[11] The spatial growth term is discretized as follows: (i) directly in the cells, for the component proportional to $f(u)$; (ii) in the nodes, for the component proportional to $df(u)/du$. This discretisation allows satisfying the particle and the energy conservation equations - see sections 7.2-7.3

$$-\frac{1}{N}\frac{1}{\gamma}\left.\frac{dG_{\rm el}(u)}{du}\right|_n = \left[\frac{g_{{\rm el}_{n-1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}-\frac{1}{2}\right)\right]f_{n-1} - \left[\frac{g_{{\rm el}_{n-1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}+\frac{1}{2}\right)+\frac{g_{{\rm el}_{n+1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}-\frac{1}{2}\right)\right]f_n + \left[\frac{g_{{\rm el}_{n+1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}+\frac{1}{2}\right)\right]f_{n+1} \quad . \tag{71}$$

- **CAR operator**

$$g_{\rm CAR}(u) = 4u\sum_k B_k\chi_k\sigma_{0,k}(u) \tag{72a}$$

$$c_{\rm CAR} = \frac{k_BT_g}{e} \tag{72b}$$

$$d_{\rm CAR} = 1 \tag{72c}$$

$$-\frac{1}{N}\frac{1}{\gamma}\left.\frac{dG_{\rm CAR}(u)}{du}\right|_n = \left[\frac{g_{{\rm CAR}_{n-1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}-\frac{1}{2}\right)\right]f_{n-1} - \left[\frac{g_{{\rm CAR}_{n-1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}+\frac{1}{2}\right)+\frac{g_{{\rm CAR}_{n+1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}-\frac{1}{2}\right)\right]f_n + \left[\frac{g_{{\rm CAR}_{n+1/2}}}{\Delta u}\left(\frac{k_BT_g}{e\Delta u}+\frac{1}{2}\right)\right]f_{n+1} \quad . \tag{73}$$

- **Electron-electron collisions operator**

[In the case of time-dependent simulations, see section 1.3, the electron density $n_e(t)$ and the EEDF $f(u,t)$ introduce a time-dependence into $g_{ee}(u,t)$, $I(u,t)$ and $J(u,t)$, which are written for a given t.]

$$g_{ee}(u) = \frac{2}{\gamma}\frac{\nu_{ee}(u)}{N}u^{3/2} \tag{74a}$$

$$I(u) = \int_0^u f(u')u'^{1/2}du' \tag{74b}$$

$$J(u) = \frac{2}{3}\left[\int_0^u f(u')u'^{3/2}du' + u^{3/2}\int_u^\infty f(u')du'\right] \tag{74c}$$

$$-\frac{1}{N}\frac{1}{\gamma}\left.\frac{dG_{ee}(u)}{du}\right|_n = \left\{\frac{d}{du}\left[g_{ee}(u)\left(I(u)f(u)+J(u)\frac{df(u)}{du}\right)\right]\right\}_n \quad . \tag{75}$$
$$= A_{n-1}f_{n-1} - (A_n+B_n)f_n + B_{n+1}f_{n+1}$$

$$A_n \equiv A_{I_n}+A_{J_n} = -\frac{2}{\gamma}\frac{\nu_{ee_{n+1/2}}}{N}u_{n+1/2}^{3/2}\left[\frac{I_{n+1/2}}{2\Delta u}-\frac{J_{n+1/2}}{(\Delta u)^2}\right] \equiv \sum_{m=1}^{\mathcal{N}} a_{n,m}f_m \tag{76a}$$

$$B_n \equiv B_{I_n}+B_{J_n} = \frac{2}{\gamma}\frac{\nu_{ee_{n-1/2}}}{N}u_{n-1/2}^{3/2}\left[\frac{I_{n-1/2}}{2\Delta u}+\frac{J_{n-1/2}}{(\Delta u)^2}\right] \equiv \sum_{m=1}^{\mathcal{N}} b_{n,m}f_m \tag{76b}$$

$$A_{I_n} = -\frac{2}{\gamma}\frac{\nu_{ee_{n+1/2}}}{N}u_{n+1/2}^{3/2}\frac{I_{n+1/2}}{2\Delta u} \quad , \quad \text{etc} \tag{76c}$$

$$I_{n+1/2} = \sum_{m=1}^{n} f_m u_m^{1/2}\Delta u \tag{76d}$$

$$J_{n+1/2} = \frac{2}{3}\left[\sum_{m=1}^{n} f_m u_m^{3/2}\Delta u + u_{n+1/2}^{3/2}\sum_{m=n+1}^{\mathcal{N}} f_m\Delta u\right] \tag{76e}$$

$$a_{n,m} = -\frac{2}{\gamma}\frac{\nu_{ee_{n+1/2}}}{N}u_{n+1/2}^{3/2}\times\begin{cases}\frac{1}{2}u_m^{1/2}-\frac{2}{3}\frac{u_m^{3/2}}{\Delta u} & ,\ m\le n\\ -\frac{2}{3}\frac{u_{n+1/2}^{3/2}}{\Delta u} & ,\ m>n\end{cases} \tag{76f}$$

$$b_{n,m} = \frac{2}{\gamma}\frac{\nu_{ee_{n-1/2}}}{N}u_{n-1/2}^{3/2} \times \begin{cases} \frac{1}{2}u_m^{1/2} + \frac{2}{3}\frac{u_m^{3/2}}{\Delta u} & , \; m \leq n-1 \\ \frac{2}{3}\frac{u_{n-1/2}^{3/2}}{\Delta u} & , \; m > n-1 \end{cases} . \tag{76g}$$

- **Discrete collision operator**

$$S_n = \sum_{i,j>i} S_{(i,j)_n} + \sum_i S_{(i,\mathrm{ion})_n} + \sum_i S_{(i,\mathrm{att})_n} \tag{77a}$$

$$l_{i,j} \equiv \mathtt{INT}\left(V_{i,j}/\Delta u\right) \tag{77b}$$

$$\sigma_{i,\mathrm{ion}}(u_n^{\mathrm{cell}}) = \sum_{m=1}^{(n-l_{i,\mathrm{ion}})/2} \sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(n,m)}} \tag{77c}$$

$$\begin{aligned} S_{(i,j)_n} &= \delta_i \left[u_{n+l_{i,j}} \frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2} f_{n+l_{i,j}} - u_n \frac{\sigma_{(i,j)_n} + \sigma_{(i,j)_{n+1}}}{2} f_n \right] \\ &+ \delta_j \frac{g_i}{g_j} \left[u_n \frac{\sigma_{(i,j)_n} + \sigma_{(i,j)_{n+1}}}{2} f_{n-l_{i,j}} - u_{n+l_{i,j}} \frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2} f_n \right] \end{aligned} \tag{78a}$$

$$\begin{aligned} S_{(i,\mathrm{ion})_n} &= \delta_i \left\{ \sum_{m=n+l_{i,\mathrm{ion}}+1}^{2n+l_{i,\mathrm{ion}}} u_m \sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,m-n-l_{i,\mathrm{ion}}+1/2)}} f_m \Delta u \right. \\ &+ \left. \sum_{m=2n+l_{i,\mathrm{ion}}}^{\mathcal{N}} u_m \sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,n+1/2)}} f_m \Delta u - u_n \sigma_{(i,\mathrm{ion})_{n+1/2}} f_n \right\} \end{aligned} \tag{78b}$$

$$S_{(i,\mathrm{att})_n} = -\delta_i u_n \frac{\sigma_{(i,\mathrm{att})_n} + \sigma_{(i,\mathrm{att})_{n+1}}}{2} f_n \quad , \tag{78c}$$

where

$$\sigma_{i,\mathrm{ion}}(u) = \int_0^{(u-V_{i,\mathrm{ion}})/2} \sigma^{\mathrm{sec}}_{i,\mathrm{ion}}(u,u')du' \longrightarrow \sigma_{i,\mathrm{ion}_{m+1}} = \sum_{n=1}^{(m-l_{i,\mathrm{ion}})/2} \sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1,n+1/2)}} \Delta u \tag{79a}$$

$$\sigma_{i,\mathrm{ion}_{m+1/2}} = \frac{\sigma_{i,\mathrm{ion}_m} + \sigma_{i,\mathrm{ion}_{m+1}}}{2} \tag{79b}$$

The integrals in the ionisation operator are discretized using the mid-point quadrature rule, for example

$$\begin{aligned} \int_{a(u)}^{b(u)} u'\sigma^{\mathrm{sec}}_{(i,\mathrm{ion})}(u',u)f(u')du' &\longrightarrow \int_{a(u_{k+1}^{\mathrm{node}})}^{b(u_{n+1}^{\mathrm{node}})\Delta u} u'\sigma^{\mathrm{sec}}_{(i,\mathrm{ion})}(u',u_{n+1/2}^{\mathrm{node}})f(u')du' \\ &= \int_{k\Delta u}^{n\Delta u} u'\sigma^{\mathrm{sec}}_{(i,\mathrm{ion})}(u',u_n^{\mathrm{cell}})f(u')du' \\ &= \sum_{m=k}^{n} u_m \sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,n+1/2)}} f_m \Delta u \quad . \end{aligned} \tag{80}$$

such that the cross section values, usually defined at the grid-nodes, are **now taken at the grid-cells** (with the indexes m and n in $\sigma^{\mathrm{sec}}_{(i,\mathrm{ion})}$ referring to node numbers).

In particular,

$$u + V_{i,\mathrm{ion}} \longrightarrow u_{n+1}^{\mathrm{node}} + l_{i,\mathrm{ion}}\Delta u = (n + l_{i,\mathrm{ion}})\,\Delta u \tag{81a}$$

$$2u + V_{i,\mathrm{ion}} \longrightarrow 2u_{n+1}^{\mathrm{node}} + l_{i,\mathrm{ion}}\Delta u = (2n + l_{i,\mathrm{ion}})\,\Delta u \tag{81b}$$

$$\frac{u - V_{i,\mathrm{ion}}}{2} \longrightarrow \frac{u_{n+1}^{\mathrm{node}} - l_{i,\mathrm{ion}}\Delta u}{2} = \frac{(n - l_{i,\mathrm{ion}})\,\Delta u}{2} \quad . \tag{81c}$$

Note that, in the first term of equation (78b), the lower index in the summation becomes $n + l_{i,\mathrm{ion}} + 1$, because for $m = n + l_{i,\mathrm{ion}}$

$$\sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,1/2)}} = 0$$

since it would involve an undefined 0-cell.

The previous equations are subjected to the following conditions [44]:

- the flux boundary conditions $G(0) = G(u_{\max}) = 0$, which are implemented here for *each* operator as $g_{x_{1/2}} = g_{x_{\mathcal{N}+1/2}} = 0$ and $A_0 = A_{\mathcal{N}} = B_1 = B_{\mathcal{N}+1} = 0$.[12]
  Note that the boundary conditions $g^{\text{SST}}_{E_{1/2}} = g^{\text{SST}}_{E_{\mathcal{N}+1/2}} = 0$ result in $D_{1/2} = D_{\mathcal{N}+1/2} = 0$ (see equation (13b));
- the imposition of a Maxwellian EEDF for $dG_{ee}/du = 0$, which is ensured by setting $a_{n,m} = \sqrt{a_{n,m} \times a_{m-1,n+1}}$ (see section 7.4.2).
- the energy conservation condition for dG_{ee}/du, which yields $b_{m,n} = a_{n,m}$ (see section 7.4.1).

7.2 The particle balance equation

The EBE (7a)-(8b) and (19a)-(19b) can be integrated in energy space to yield the particle balance equation, *i.e.*

- For a PT situation

$$\frac{1}{\gamma}\frac{\langle\nu_{\text{eff}}\rangle}{N}\int_0^\infty \sqrt{u}f(u)du + \frac{1}{N}\frac{1}{\gamma}\int_0^\infty \frac{dG(u)}{du}du = \int_0^\infty S(u)du$$

  and therefore

$$\frac{\langle\nu_{\text{eff}}\rangle}{N} = \gamma\int_0^\infty S(u)du \quad , \tag{82}$$

  where we have used the normalisation condition and the boundary conditions for the upflux $G(u)$.

- For a SST situation

$$-\frac{1}{3}\left(\frac{\alpha_{\text{eff}}}{N}\right)^2\int_0^\infty \frac{u}{\Omega_{\text{SST}}(u)}f(u)du - \frac{1}{3}\frac{\alpha_{\text{eff}}}{N}\frac{E}{N}\int_0^\infty \frac{u}{\Omega_{\text{SST}}(u)}\frac{df(u)}{du}du + \frac{1}{N}\frac{1}{\gamma}\int_0^\infty \frac{dG(u)}{du}du = \int_0^\infty S(u)du$$

  and therefore (cf. equation (37))

$$-(D_e N)\left(\frac{\alpha_{\text{eff}}}{N}\right)^2 + (\mu_e N)\frac{E}{N}\frac{\alpha_{\text{eff}}}{N} = \frac{\langle\nu_{\text{eff}}\rangle}{N} \quad , \tag{83}$$

  where we have used equation (82), the boundary conditions for the upflux $G(u)$, and the expressions (26a) and (26b) of the DC reduced transverse diffusion coefficient and the DC reduced mobility, respectively.

- For a time-dependent situation

$$\frac{1}{N}\frac{1}{\gamma}\frac{d}{dt}\int_0^\infty \sqrt{u}f(u,t)du + \frac{1}{\gamma}\frac{\langle\nu_{\text{eff}}\rangle(t)}{N}\int_0^\infty \sqrt{u}f(u,t)du + \frac{1}{N}\frac{1}{\gamma}\int_0^\infty \frac{\partial G(u,t)}{\partial u}du = \int_0^\infty S(u,t)du$$

  and therefore

$$\frac{\langle\nu_{\text{eff}}\rangle(t)}{N} = \gamma\int_0^\infty S(u,t)du \quad , \tag{84}$$

  where, similarly to the PT situation, we have used the normalisation condition and the boundary conditions for the upflux $G(u)$.

[12] Note that the *strict* boundary conditions should be $G(0) = G(u_{\max}) = \sum_x G_x(0) = \sum_x G_x(u_{\max}) = 0$. Work on their numerical implementation is underway.

Note that the integration of the right-hand sides of equations (7a), (8a) and (19a) yields

$$\begin{aligned}
\int_0^\infty S_{\text{inel}}(u)du &= \sum_{i,j>i}\int_0^\infty S_{(i,j),\text{inel}}(u)du = \\
&= \sum_{i,j>i}\int_0^\infty \delta_i\left[\sigma_{i,j}(u+V_{i,j})f(u+V_{i,j})(u+V_{i,j}) - \sigma_{i,j}(u)f(u)u\right]du \\
&= \sum_{i,j>i}\delta_i\left[\int_{V_{i,j}}^\infty \sigma_{i,j}(u')f(u')u'du' - \int_0^\infty \sigma_{i,j}(u)f(u)udu\right] \\
&= \sum_{i,j>i}\delta_i\left[\int_0^\infty \sigma_{i,j}(u)f(u)udu - \int_0^\infty \sigma_{i,j}(u)f(u)udu\right] \\
&= 0 \quad , \qquad (85)
\end{aligned}$$

where we have considered that $\sigma_{i,j}(u) = 0$ for $u \leq V_{i,j}$.

7.2.1 Discretisation of the particle balance equation

The numerical discretisation of the EBE must ensure the electron-particle conservation, *i.e.* it must allow retrieving the numerical version of equations (82)-(83). In particular:

- normalisation condition
  The EEDF is calculated as to satisfy the normalisation condition

$$\int_0^\infty f(u)\sqrt{u}du = 1 \longrightarrow \sum_{n=1}^{\mathcal{N}} f_n\sqrt{u_n}\Delta u_n = 1 \quad ; \qquad (86)$$

- DC free-diffusion coefficient
  The DC reduced transverse free-diffusion coefficient is calculated using (see equations (26a), (66) and (67a))

$$\begin{aligned}
D_eN &= \frac{\gamma}{3}\int_0^\infty \frac{u}{\Omega_{\text{SST}}(u)}f(u)du \\
&\longrightarrow \frac{\gamma}{3}\sum_{n=1}^{\mathcal{N}}\int_{u_{n-1/2}}^{u_{n+1/2}} \frac{u}{\Omega_{\text{SST}}(u)}f(u)du \\
&\longrightarrow \gamma\Delta u\sum_{n=1}^{\mathcal{N}}\frac{u_n}{3\Omega_{\text{SST}_n}}f_n \\
&\longrightarrow \gamma\Delta u\sum_{n=1}^{\mathcal{N}} D_nf_n \quad ; \qquad (87)
\end{aligned}$$

- DC reduced mobility
  The DC reduced mobility is calculated using (see equations (26b), (66) and (67b); recall that $D_{1/2} =$

$D_{\mathcal{N}+1/2} = 0$)

$$
\begin{aligned}
\mu_e N &= -\frac{\gamma}{3}\int_0^\infty \frac{u}{\Omega_{\mathrm{SST}}(u)}\frac{df(u)}{du}du \\
&\longrightarrow -\frac{\gamma}{3}\sum_{n=1}^{\mathcal{N}}\int_{u_{n-1/2}}^{u_{n+1/2}} \frac{u}{\Omega_{\mathrm{SST}}(u)}\frac{df(u)}{du}du \\
&\longrightarrow -\gamma\sum_{n=1}^{\mathcal{N}}\frac{1}{2}\left[\left(\frac{u}{3\Omega_{\mathrm{SST}}(u)}\frac{df(u)}{du}\right)_{n+1}^{\mathrm{node}} + \left(\frac{u}{3\Omega_{\mathrm{SST}}(u)}\frac{df(u)}{du}\right)_{n}^{\mathrm{node}}\right]\Delta u \\
&\longrightarrow -\gamma\sum_{n=1}^{\mathcal{N}}\frac{1}{2}\left[D_{n+1/2}\left(\frac{f_{n+1}-f_n}{\Delta u}\right) + D_{n-1/2}\left(\frac{f_n-f_{n-1}}{\Delta u}\right)\right]\Delta u \\
&\longrightarrow -\gamma\frac{1}{2}\left[\sum_{n=1}^{\mathcal{N}-1} D_{n+1/2}\frac{f_{n+1}-f_n}{\Delta u} + \sum_{n=2}^{\mathcal{N}} D_{n-1/2}\frac{f_n-f_{n-1}}{\Delta u}\right]\Delta u \\
&\longrightarrow -\gamma\sum_{n=2}^{\mathcal{N}} D_{n-1/2}\frac{f_n-f_{n-1}}{\Delta u}\Delta u = -\gamma\sum_{n=1}^{\mathcal{N}} D_{n-1/2}\frac{f_n-f_{n-1}}{\Delta u}\Delta u \quad ;
\end{aligned}
\tag{88}
$$

- flux conservation
  The integration of each upflux term ($x = E, \mathrm{el}, \mathrm{CAR}$) yields a zero-identity (see equation (62b); recall that $f_0 = f_{\mathcal{N}+1} = 0$ and $g_{x_{1/2}} = g_{x_{\mathcal{N}+1/2}} = 0$)

$$
\begin{aligned}
-\frac{1}{N}\frac{1}{\gamma}\int_0^\infty \frac{dG_x(u)}{du}du = 0 \longrightarrow -\frac{1}{N}\frac{1}{\gamma}\sum_{n=1}^{\mathcal{N}}\left(\frac{dG_x(u)}{du}\right)_n \Delta u \\
&= \sum_{n=1}^{\mathcal{N}}\left\{\frac{g_{x_{n-1/2}}}{\Delta u}\left[\frac{c_x}{\Delta u}-\frac{d_x}{2}\right]\right\} f_{n-1}\Delta u \\
&- \sum_{n=1}^{\mathcal{N}}\left\{\frac{g_{x_{n+1/2}}}{\Delta u}\left[\frac{c_x}{\Delta u}-\frac{d_x}{2}\right] + \frac{g_{x_{n-1/2}}}{\Delta u}\left[\frac{c_x}{\Delta u}+\frac{d_x}{2}\right]\right\} f_n\Delta u \\
&+ \sum_{n=1}^{\mathcal{N}}\left\{\frac{g_{x_{n+1/2}}}{\Delta u}\left[\frac{c_x}{\Delta u}+\frac{d_x}{2}\right]\right\} f_{n+1}\Delta u \\
&= \sum_{n=1}^{\mathcal{N}-1}\left\{g_{x_{n+1/2}}\left[\frac{c_x}{\Delta u}-\frac{d_x}{2}\right]\right\} f_n \\
&- \sum_{n=1}^{\mathcal{N}-1}\left\{g_{x_{n+1/2}}\left[\frac{c_x}{\Delta u}-\frac{d_x}{2}\right]\right\} f_n - \sum_{n=2}^{\mathcal{N}}\left\{g_{x_{n-1/2}}\left[\frac{c_x}{\Delta u}+\frac{d_x}{2}\right]\right\} f_n \\
&+ \sum_{n=2}^{\mathcal{N}}\left\{g_{x_{n-1/2}}\left[\frac{c_x}{\Delta u}+\frac{d_x}{2}\right]\right\} f_n \\
&= 0 \quad ;
\end{aligned}
\tag{89}
$$

- flux conservation for electron-electron collisions
  The integration of the upflux term for electron-electron collisions yields also a zero-identity (see equa-

tion (75); recall that $f_0 = f_{\mathcal{N}+1} = 0$ and $A_0 = A_{\mathcal{N}} = B_1 = B_{\mathcal{N}+1} = 0$)

$$
\begin{aligned}
-\frac{1}{N}\frac{1}{\gamma}\int_0^\infty \frac{dG_{ee}(u)}{du}du = 0 \longrightarrow -\frac{1}{N}\frac{1}{\gamma}\sum_{n=1}^{\mathcal{N}}\left(\frac{dG_{ee}(u)}{du}\right)_n \Delta u \\
&= \sum_{n=1}^{\mathcal{N}} \left[A_{n-1}f_{n-1} - (A_n + B_n)\, f_n + B_{n+1}f_{n+1}\right] \\
&= \sum_{n=2}^{\mathcal{N}} A_{n-1}f_{n-1} - \sum_{n=1}^{\mathcal{N}-1} A_n f_n - \sum_{n=2}^{\mathcal{N}} B_n f_n + \sum_{n=1}^{\mathcal{N}-1} B_{n+1}f_{n+1} \\
&= \sum_{n=1}^{\mathcal{N}-1} A_n f_n - \sum_{n=1}^{\mathcal{N}-1} A_n f_n - \sum_{n=1}^{\mathcal{N}-1} B_{n+1}f_{n+1} + \sum_{n=1}^{\mathcal{N}-1} B_{n+1}f_{n+1} \\
&= 0 \quad .
\end{aligned}
\tag{90}
$$

- inelastic/superelastic collisions
  The integration of the inelastic and superelastic collision terms yields a zero identify (see equations (78a))

$$
\begin{aligned}
\int_0^\infty S_{\text{inel}}(u)du = 0 \longrightarrow \sum_{n=1}^{\mathcal{N}} S_n^{\text{inel}}\Delta u = \sum_{i,j>i}\sum_{n=1}^{\mathcal{N}} S_{(i,j)_n}^{\text{inel}}\Delta u \\
&= \sum_{i,j>i}\sum_{n=1}^{\mathcal{N}} \delta_i \left[u_{n+l_{i,j}}\sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_{n+l_{i,j}} - u_n \sigma_{(i,j)_{n+1/2}}\, f_n\right]\Delta u \\
&= \sum_{i,j>i} \delta_i \left[\sum_{n=1}^{\mathcal{N}-l_{i,j}} u_{n+l_{i,j}}\sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_{n+l_{i,j}} - \sum_{n=1+l_{i,j}}^{\mathcal{N}} u_n \sigma_{(i,j)_{n+1/2}}\, f_n\right]\Delta u \\
&= \sum_{i,j>i} \delta_i \left[\sum_{n=1}^{\mathcal{N}-l_{i,j}} u_{n+l_{i,j}}\sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_{n+l_{i,j}} - \sum_{n=1}^{\mathcal{N}-l_{i,j}} u_{n+l_{i,j}} \sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_{n+l_{i,j}}\right]\Delta u \\
&= 0 \quad \text{for } \sigma_{(i,j)_{n+1/2}} = 0, n \le l_{i,j} \quad ;
\end{aligned}
\tag{91a}
$$

$$
\begin{aligned}
\int_0^\infty S_{\text{sup}}(u)du = 0 \longrightarrow \sum_{n=1}^{\mathcal{N}} S_n^{\text{sup}}\Delta u = \sum_{i,j>i}\sum_{n=1}^{\mathcal{N}} S_{(i,j)_n}^{\text{sup}}\Delta u \\
&= \sum_{i,j>i}\sum_{n=1}^{\mathcal{N}} \delta_j \frac{g_i}{g_j} \left[u_n\sigma_{(i,j)_{n+1/2}}\, f_{n-l_{i,j}} - u_{n+l_{i,j}} \sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_n\right]\Delta u \\
&= \sum_{i,j>i} \delta_j \frac{g_i}{g_j} \left[\sum_{n=1+l_{i,j}}^{\mathcal{N}} u_n\sigma_{(i,j)_{n+1/2}}\, f_{n-l_{i,j}} - \sum_{n=1}^{\mathcal{N}-l_{i,j}} u_{n+l_{i,j}} \sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_n\right]\Delta u \\
&= \sum_{i,j>i} \delta_j \frac{g_i}{g_j} \left[\sum_{n=1}^{\mathcal{N}-l_{i,j}} u_{n+l_{i,j}}\sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_n - \sum_{n=1}^{\mathcal{N}-l_{i,j}} u_{n+l_{i,j}} \sigma_{(i,j)_{n+l_{i,j}+1/2}}\, f_n\right]\Delta u \\
&= 0 \quad \text{for } \sigma_{(i,j)_{n+1/2}} = 0, n \le l_{i,j} \quad .
\end{aligned}
\tag{91b}
$$

- ionisation

The integration of the ionisation collision terms yields the reduced-ionisation frequency (see equation (78b))

$$\begin{aligned}
\int_0^\infty S_{\rm ion}(u)du = 0 \longrightarrow \sum_{n=1}^{\mathcal{N}} S_n^{\rm ion}\Delta u &= \sum_i \sum_{n=1}^{\mathcal{N}} S_{(i,{\rm ion})_n}\Delta u \\
&= \sum_i \sum_{n=1}^{\mathcal{N}} \delta_i \left\{ \sum_{m=n+l_{i,\rm ion}+1}^{2n+l_{i,\rm ion}} u_m \sigma^{\rm sec}_{(i,{\rm ion})_{(m+1/2,m-n-l_{i,\rm ion}+1/2)}} f_m \Delta u \right. \\
&\left. + \sum_{m=2n+l_{i,\rm ion}}^{\mathcal{N}} u_m \sigma^{\rm sec}_{(i,{\rm ion})_{(m+1/2,n+1/2)}} f_m \Delta u - u_n \sigma_{(i,{\rm ion})_{n+1/2}} f_n \right\} \Delta u \\
&= \sum_i \delta_i \left\{ \sum_{m=1}^{\mathcal{N}} u_m f_m \left[\sum_{k=1}^{(m-l_{i,\rm ion})/2} \sigma^{\rm sec}_{(i,{\rm ion})_{(m+1/2,k+1/2)}} \Delta u \right] \right. \\
&\left. + \sum_{m=1}^{\mathcal{N}} u_m f_m \left[\sum_{n=1}^{(m-l_{i,\rm ion})/2} \sigma^{\rm sec}_{(i,{\rm ion})_{(m+1/2,n+1/2)}} \Delta u \right] - \sum_{n=1}^{\mathcal{N}} u_n f_n \sigma_{(i,{\rm ion})_{n+1/2}} \right\} \Delta u \\
&= \sum_i \delta_i \left\{ \sum_{m=1}^{\mathcal{N}} u_m f_m \sigma_{(i,{\rm ion})_{m+1/2}} + \sum_{m=1}^{\mathcal{N}} u_m f_m \sigma_{(i,{\rm ion})_{m+1/2}} - \sum_{n=1}^{\mathcal{N}} u_n f_n \sigma_{(i,{\rm ion})_{n+1/2}} \right\} \Delta u \\
&= \sum_i \delta_i \sum_{n=1}^{\mathcal{N}} u_n f_n \sigma_{(i,{\rm ion})_{n+1/2}} = \frac{1}{\gamma} \sum_i \delta_i \frac{\langle \nu_{i,\rm ion} \rangle}{N} = \frac{1}{\gamma} \frac{\langle \nu_{\rm ion} \rangle}{N} \quad ,
\end{aligned} \tag{92}$$

where we have used equations (31a) and (79a), and considered

$$\begin{aligned}
& n + l_{i,\rm ion} + 1 \le m \le 2n + l_{i,\rm ion} \\
& \quad \Longrightarrow 1 \le m - n - l_{i,\rm ion} \equiv k \le n \le \frac{m - l_{i,\rm ion}}{2} \quad , \quad (\text{for } m = 2n + l_{i,\rm ion}) \\
& 2n + l_{i,\rm ion} \le m \le \mathcal{N} \Longrightarrow 1 \le n \le \frac{m - l_{i,\rm ion}}{2} \le \mathcal{N} \quad .
\end{aligned}$$

- attachment
  The integration of the attachment collision term yields the reduced-attachment frequency (see equation (78c))

$$\begin{aligned}
\int_0^\infty S_{\rm att}(u)du = 0 \longrightarrow \sum_{n=1}^{\mathcal{N}} S_n^{\rm att}\Delta u &= \sum_i \sum_{n=1}^{\mathcal{N}} S_{(i,{\rm att})_n}\Delta u \\
&= -\sum_i \delta_i \sum_{n=1}^{\mathcal{N}} u_n \frac{\sigma_{(i,{\rm att})_n} + \sigma_{(i,{\rm att})_{n+1}}}{2} f_n \Delta u \\
&= \frac{1}{\gamma} \sum_i \delta_i \frac{\langle \nu_{i,\rm att} \rangle}{N} = \frac{1}{\gamma} \frac{\langle \nu_{\rm att} \rangle}{N} \quad .
\end{aligned} \tag{93}$$

The discretized forms of the particle balance equations are verified by using equations (86)-(93) in the expressions (82), (83) and (84).

7.3 The power balance equation

The balance equation for the electron power-density, per electron at unit gas density, is obtained (depending on the particular PT/SST situation considered) by multiplying either (38a) or (38b) by γu and integrating it over all energies; or by multiplying either (38c) or (38d) by $u^{3/2}/N$ and integrating it over all energies (see section 2.2). The general result writes (in eV s^{-1} m^3)

$$\frac{\Theta(t)}{N} = \frac{\Theta_{\rm growth}}{N} + \frac{\Theta_E}{N} + \frac{\Theta_{\rm coll}}{N} \quad , \tag{94a}$$

where the term in the left-hand side represents the intrinsic time-variation of the power density, and the different terms on the right-hand side represent, in order, the variation of the power density due to non-conservative

mechanisms inducing a time/space net growth of the electron density, the power density gained from the applied electric field, and the power density gained/lost in collisional ("coll") events, which can be separately written as (note that all "gain" terms are positively defined and all "loss" terms are negatively defined).

$$\frac{\Theta_{\rm coll}}{N} \equiv \frac{\Theta_{\rm coll}^{\rm gain}}{N} + \frac{\Theta_{\rm coll}^{\rm loss}}{N} \quad . \tag{94b}$$

7.3.1 Discretisation of the power balance equation

The discretized form of (94a) can be obtained using the same finite-volume-like procedure as in section 7.1, applying Gauss' theorem to integrals over one-dimensional energy volumes around each n-cell centered at energy u_n (see (62a)-(62b)).

The different terms in (94a) are written as follows.

- intrinsic time-variation term
  The left-hand side of (39a) is

$$\frac{\Theta(t)}{N} = \begin{cases} 0 \ , \text{ for stationary and steady-state simulations} \\ \dfrac{1}{N}\displaystyle\int_0^\infty u^{3/2}\frac{\partial f(u,t)}{\partial t}du = \frac{1}{N}\frac{d\varepsilon(t)}{dt} \ , \text{ for time-dependent simulations} \end{cases} . \tag{95}$$

  The quantity $\Theta(t)/N$ is numerically calculated by summing all the power terms on the right-hand side of (94a), and on output the value of this quantity is tagged as *Power balance* (see section 4.4).

- time/space net growth term

$$\frac{\Theta_{\rm growth}}{N} = \begin{cases} -\dfrac{\langle\nu_{\rm eff}\rangle}{N}\displaystyle\int_0^\infty \frac{1}{\sqrt{u}}u^2 f(u)du = -\frac{\langle\nu_{\rm eff}\rangle}{N}\varepsilon \\ \quad \to -\dfrac{\langle\nu_{\rm eff}\rangle}{N}\displaystyle\sum_{n=1}^{\mathcal{N}} u_n^{3/2} f_n \Delta u \ , \\ \quad \text{for a PT situation} \\ -\dfrac{\alpha_{\rm eff}}{N}\dfrac{\gamma}{3}\displaystyle\int_0^\infty u^2 f^1(u)du \\ \quad = -\dfrac{\alpha_{\rm eff}}{N}\dfrac{\gamma}{3}\displaystyle\int_0^\infty u^2\left[-\frac{E/N}{\Omega_{\rm SST}(u)}\frac{df(u)}{du} - \frac{\alpha_{\rm eff}/N}{\Omega_{\rm SST}(u)}f(u)\right]du = -\dfrac{\alpha_{\rm eff}}{N}\left(\mu_\varepsilon E - D_\varepsilon \alpha_{\rm eff}\right) \\ \quad \to -\dfrac{\alpha_{\rm eff}}{N}\gamma\displaystyle\sum_{n=1}^{\mathcal{N}}\left[-\frac{E}{N}u_{n-1/2}D_{n-1/2}\frac{f_n - f_{n-1}}{\Delta u} - \frac{\alpha_{\rm eff}}{N}u_n D_n f_n\right]\Delta u, \\ \quad \text{for a SST situation (see equations (28a)-(28b) and (87)-(88))} \end{cases} \tag{96}$$

- terms involving upflux functions $G_{x=E,{\rm el},{\rm CAR}}(u)$
  The $\Theta_{x=E,{\rm el},{\rm CAR}}/N$ terms involving upflux functions $G_x(u)$ are handled as follows (see equations (13a)-(13b) and (68b)-(68c); (13c) and (70b)-(70c); (13d) and (72b)-(72c))

$$\begin{aligned} \frac{\Theta_{x=E,{\rm el},{\rm CAR}}}{N} &= -\frac{1}{N}\sum_n \int_{u_{n-1/2}}^{u_{n+1/2}} u\frac{dG_x(u)}{du}du = -\frac{1}{N}\sum_n u_n\left[G_{x_{n+1/2}} - G_{x_{n-1/2}}\right] \\ &= \gamma\sum_n\left\{u_n\left[g_{x_{n+1/2}}\left(\frac{d_x}{2}(f_n + f_{n+1}) + \frac{c_x}{\Delta u}(f_{n+1} - f_n)\right)\right.\right. \\ &- \left.\left. g_{x_{n-1/2}}\left(\frac{d_x}{2}(f_{n-1} + f_n) + \frac{c_x}{\Delta u}(f_n - f_{n-1})\right)\right]\right\} \\ &= \gamma\left\{\sum_{n=1}^{\mathcal{N}} u_n\left[g_{x_{n+1/2}}\left(\frac{d_x}{2} - \frac{c_x}{\Delta u}\right) - g_{x_{n-1/2}}\left(\frac{d_x}{2} + \frac{c_x}{\Delta u}\right)\right]f_n\right. \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{n=1}^{\mathcal{N}} u_n \left[g_{x_{n+1/2}} \left(\frac{d_x}{2} + \frac{c_x}{\Delta u} \right) \right] f_{n+1} - \sum_{n=1}^{\mathcal{N}} u_n \left[g_{x_{n-1/2}} \left(\frac{d_x}{2} - \frac{c_x}{\Delta u} \right) \right] f_{n-1} \Bigg\} \\
&= \gamma \Bigg\{ \sum_{n=1}^{\mathcal{N}} u_n \left[g_{x_{n+1/2}} \left(\frac{d_x}{2} - \frac{c_x}{\Delta u} \right) - g_{x_{n-1/2}} \left(\frac{d_x}{2} + \frac{c_x}{\Delta u} \right) \right] f_n \\
&+ \sum_{n=2}^{\mathcal{N}+1} u_{n-1} \left[g_{x_{n-1/2}} \left(\frac{d_x}{2} + \frac{c_x}{\Delta u} \right) \right] f_n - \sum_{n=0}^{\mathcal{N}-1} u_{n+1} \left[g_{x_{n+1/2}} \left(\frac{d_x}{2} - \frac{c_x}{\Delta u} \right) \right] f_n \Bigg\} \\
&= \gamma \left[\sum_{n=1}^{\mathcal{N}} (u_n - u_{n+1}) \, g_{x_{n+1/2}} \left(\frac{d_x}{2} - \frac{c_x}{\Delta u} \right) - (u_n - u_{n-1}) \, g_{x_{n-1/2}} \left(\frac{d_x}{2} + \frac{c_x}{\Delta u} \right) \right] f_n \\
&= \gamma \left[\sum_{n=1}^{\mathcal{N}} g_{x_{n+1/2}} \left(c_x - d_x \frac{\Delta u}{2} \right) - g_{x_{n-1/2}} \left(c_x + d_x \frac{\Delta u}{2} \right) \right] f_n \quad , \qquad (97)
\end{aligned}
$$

where we have used the boundary conditions $g_{x_{1/2}} = g_{x_{\mathcal{N}+1/2}} = 0$.

Note that the power absorbed from the electric field can also be written as

- for a PT situation (see equations (13a), (68b)-(68c), (67a) and (97))

$$
\begin{aligned}
\frac{\Theta_E^{\mathrm{PT}}}{N} &= \gamma \left(\frac{E}{N} \right)^2 \sum_{n=1}^{\mathcal{N}} \left[g^{\mathrm{PT}}_{E_{n+1/2}} - g^{\mathrm{PT}}_{E_{n-1/2}} \right] f_n \\
&= \gamma \left(\frac{E}{N} \right)^2 \left[\sum_{n=2}^{\mathcal{N}+1} g^{\mathrm{PT}}_{E_{n-1/2}} f_{n-1} - \sum_{n=1}^{\mathcal{N}} g^{\mathrm{PT}}_{E_{n-1/2}} f_n \right] \\
&= -\gamma \left(\frac{E}{N} \right)^2 \sum_{n=1}^{\mathcal{N}} g^{\mathrm{PT}}_{E_{n-1/2}} (f_n - f_{n-1}) \\
&= \left[-\gamma \sum_{n=1}^{\mathcal{N}} \frac{u_{n-1/2}}{3\Omega_{\mathrm{PT}_{n-1/2}}} (f_n - f_{n-1}) \right] \left(\frac{E}{N} \right)^2 \\
&= \left[-\gamma \sum_{n=1}^{\mathcal{N}} D^{\mathrm{PT}}_{n-1/2} (f_n - f_{n-1}) \right] \left(\frac{E}{N} \right)^2 = (\mu_e N) \left(\frac{E}{N} \right)^2 \quad , \qquad (98a)
\end{aligned}
$$

where we have used the boundary conditions $g^{\mathrm{PT}}_{E_{1/2}} = g^{\mathrm{PT}}_{E_{\mathcal{N}+1/2}} = 0$, and equation (88).

- for a SST situation (see equations (13b), (68b)-(68c), (67a)-(67b) and (97))

$$
\begin{aligned}
\frac{\Theta_E^{\mathrm{SST}}}{N} &= \gamma \left(\frac{E}{N} \right)^2 \sum_{n=1}^{\mathcal{N}} \left[g^{\mathrm{SST}}_{E_{n+1/2}} - g^{\mathrm{SST}}_{E_{n-1/2}} \right] f_n - \gamma \frac{E}{N} \frac{\alpha_{\mathrm{eff}}}{N} \frac{1}{2} \sum_{n=1}^{\mathcal{N}} \left[g^{\mathrm{SST}}_{E_{n+1/2}} + g^{\mathrm{SST}}_{E_{n-1/2}} \right] f_n \Delta u \\
&= \gamma \Bigg[\left(\frac{E}{N} \right)^2 \sum_{n=2}^{\mathcal{N}+1} g^{\mathrm{SST}}_{E_{n-1/2}} f_{n-1} - \left(\frac{E}{N} \right)^2 \sum_{n=1}^{\mathcal{N}} g^{\mathrm{SST}}_{E_{n-1/2}} f_n \\
&\quad - \frac{E}{N} \frac{\alpha_{\mathrm{eff}}}{N} \frac{1}{2} \sum_{n=2}^{\mathcal{N}+1} g^{\mathrm{SST}}_{E_{n-1/2}} f_{n-1} \Delta u - \frac{E}{N} \frac{\alpha_{\mathrm{eff}}}{N} \frac{1}{2} \sum_{n=1}^{\mathcal{N}} g^{\mathrm{SST}}_{E_{n-1/2}} f_n \Delta u \Bigg] \\
&= -\gamma \left(\frac{E}{N} \right)^2 \sum_{n=1}^{\mathcal{N}} g^{\mathrm{SST}}_{E_{n-1/2}} (f_n - f_{n-1}) - \gamma \frac{E}{N} \frac{\alpha_{\mathrm{eff}}}{N} \sum_{n=1}^{\mathcal{N}} g^{\mathrm{SST}}_{E_{n-1/2}} \frac{f_{n-1} + f_n}{2} \Delta u \\
&= \left[-\gamma \sum_{n=1}^{\mathcal{N}} \frac{u_{n-1/2}}{3\Omega_{\mathrm{SST}_{n-1/2}}} (f_n - f_{n-1}) \right] \left(\frac{E}{N} \right)^2 - \left[\gamma \sum_{n=1}^{\mathcal{N}} \frac{u_{n-1/2}}{3\Omega_{\mathrm{SST}_{n-1/2}}} f_{n-1/2} \Delta u \right] \frac{E}{N} \frac{\alpha_{\mathrm{eff}}}{N} \\
&= \left[-\gamma \sum_{n=1}^{\mathcal{N}} D^{\mathrm{SST}}_{n-1/2} (f_n - f_{n-1}) \right] \left(\frac{E}{N} \right)^2 - \left[\gamma \sum_{n=1}^{\mathcal{N}} D^{\mathrm{SST}}_{n-1/2} f_{n-1/2} \Delta u \right] \frac{E}{N} \frac{\alpha_{\mathrm{eff}}}{N} \\
&= (\mu_e N) \left(\frac{E}{N} \right)^2 - (D_e N) \frac{E}{N} \frac{\alpha_{\mathrm{eff}}}{N} \quad , \qquad (98b)
\end{aligned}
$$

where we have used the boundary conditions $g^{\rm SST}_{E_{1/2}} = g^{\rm SST}_{E_{\mathcal{N}+1/2}} = 0$, and equations (87)-(88).

Furthermore, the discretized form of the collisional gain-terms writes (see equation (40e))

$$\frac{\Theta^{\rm gain}_{x={\rm el,CAR}}}{N}\gamma\int_0^\infty u\frac{d}{du}\left[g_x(u)c_x\frac{df(u)}{du}\right]du \to \gamma\sum_{n=1}^{\mathcal{N}} c_x\left[g_{x_{n+1/2}} - g_{x_{n-1/2}}\right]f_n \quad . \tag{99}$$

- terms involving the upflux function $G_{\rm ee}(u)$
  The $\Theta_{\rm ee}/N$ terms involving the upflux function $G_{\rm ee}(u)$ are specifically handled as follows (see equation (75))

$$\begin{aligned}
\frac{\Theta_{ee}}{N} &= -\frac{1}{N}\sum_n\int_{u_{n-1/2}}^{u_{n+1/2}} u\frac{dG_{\rm ee}(u)}{du}du \\
&= \gamma\sum_{n=1}^{\mathcal{N}} u_n\left[A_{n-1}f_{n-1} - (A_n+B_n)f_n + B_{n+1}f_{n+1}\right]\Delta u \\
&= \gamma\left[\sum_{n=0}^{\mathcal{N}-1} u_{n+1}A_nf_n - \sum_{n=1}^{\mathcal{N}} u_n(A_n+B_n)f_n + \sum_{n=2}^{\mathcal{N}+1} u_{n-1}B_nf_n\right]\Delta u \\
&= \gamma\left[\sum_{n=1}^{\mathcal{N}-1}(u_{n+1}-u_n)A_nf_n - \sum_{n=2}^{\mathcal{N}}(u_n-u_{n-1})B_nf_n\right]\Delta u \\
&= \gamma\sum_{n=1}^{\mathcal{N}}(A_n-B_n)f_n(\Delta u)^2 \qquad (100a) \\
\frac{\Theta^{\rm gain}_{ee}}{N} &= \gamma\sum_n\int_{u_{n-1/2}}^{u_{n+1/2}} u\frac{d}{du}\left[g_{ee}(u)J(u)\frac{df(u)}{du}\right]du \\
&= \gamma\sum_{n=1}^{\mathcal{N}}(A_{J_n}-B_{J_n})f_n(\Delta u)^2 \quad , \qquad (100b)
\end{aligned}$$

  where we have used the boundary conditions $A_0 = A_{\mathcal{N}} = B_1 = B_{\mathcal{N}+1} = 0$.

- inelastic/superelastic collision terms
  The terms $\Theta^{\rm loss}_{\rm inel}/N$ and $\Theta^{\rm gain}_{\rm sup}/N$, relative to the power lost/gained in inelastic/superelastic collisions are obtained as follows (see equation (78a))

$$\begin{aligned}
\frac{\Theta^{\rm loss}_{(i,j),{\rm inel}}}{N} &= \gamma\sum_n\int_{u_{n-1/2}}^{u_{n+1/2}} S_{(i,j)}\big|_{\rm inel}(u)udu \\
&= \delta_i\gamma\sum_{n=1}^{\mathcal{N}} u_n\left[u_{n+l_{i,j}}\frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2}f_{n+l_{i,j}} - u_n\frac{\sigma_{(i,j)_n}+\sigma_{(i,j)_{n+1}}}{2}f_n\right]\Delta u \\
&= \delta_i\gamma\left[\sum_{n=l_{i,j}+1}^{l_{i,j}+\mathcal{N}} u_{n-l_{i,j}}u_n\frac{\sigma_{(i,j)_n}+\sigma_{(i,j)_{n+1}}}{2}f_n - \sum_{n=1}^{\mathcal{N}} u_nu_n\frac{\sigma_{(i,j)_n}+\sigma_{(i,j)_{n+1}}}{2}f_n\right]\Delta u \\
&= \delta_i\gamma\sum_{n=1}^{\mathcal{N}}\left(u_{n-l_{i,j}}-u_n\right)u_n\frac{\sigma_{(i,j)_n}+\sigma_{(i,j)_{n+1}}}{2}f_n\Delta u \\
&= -\delta_i\gamma\sum_{n=1}^{\mathcal{N}}\left[u_n\frac{\sigma_{(i,j)_n}+\sigma_{(i,j)_{n+1}}}{2}f_n\Delta u\right]V_{i,j} = -\delta_iC_{i,j}V_{i,j} \qquad (101a) \\
\frac{\Theta^{\rm loss}_{\rm inel}}{N} &= \sum_{i,j>i}\frac{\Theta^{\rm loss}_{(i,j),{\rm inel}}}{N} = \sum_{i,j>i} -\delta_iC_{i,j}V_{i,j} \quad ; \qquad (101b)
\end{aligned}$$

$$
\begin{aligned}
\frac{\Theta^{\text{gain}}_{(i,j),\text{sup}}}{N} &= \gamma \sum_n \int_{u_{n-1/2}}^{u_{n+1/2}} S_{(i,j)}\big|_{\text{sup}}(u) u du \\
&= \delta_j \frac{g_i}{g_j} \gamma \sum_{n=1}^{\mathcal{N}} u_n \left[u_n \frac{\sigma_{(i,j)_n} + \sigma_{(i,j)_{n+1}}}{2} f_{n-l_{i,j}} - u_{n+l_{i,j}} \frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2} f_n \right] \Delta u \\
&= \delta_j \frac{g_i}{g_j} \gamma \left[\sum_{n=l_{i,j}+1}^{\mathcal{N}} u_n u_n \frac{\sigma_{(i,j)_n} + \sigma_{(i,j)_{n+1}}}{2} f_{n-l_{i,j}} - \sum_{n=1}^{\mathcal{N}-l_{i,j}} u_n u_{n+l_{i,j}} \frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2} f_n \right] \Delta u \\
&= \delta_j \frac{g_i}{g_j} \gamma \sum_{n=1}^{\mathcal{N}} \left(u_{n+l_{i,j}} - u_n \right) u_{n+l_{i,j}} \frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2} f_n \Delta u \\
&= \delta_j \frac{g_i}{g_j} \gamma \sum_{n=1}^{\mathcal{N}} \left[u_{n+l_{i,j}} \frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2} f_n \Delta u \right] V_{i,j} = \delta_j C_{j,i} V_{i,j} \qquad (102a) \\
\frac{\Theta^{\text{gain}}_{\text{sup}}}{N} &= \sum_{i,j>i} \frac{\Theta^{\text{gain}}_{(i,j),\text{sup}}}{N} = \sum_{i,j>i} \delta_j C_{j,i} V_{i,j} \quad . \qquad (102b)
\end{aligned}
$$

In the previous equations we have used the discretized form of the inelastic / superelastic rate coefficients (see equations (30a)-(30b))

$$
\begin{aligned}
C_{i,j} &= \gamma \sum_{n=1}^{\mathcal{N}} u_n \frac{\sigma_{(i,j)_n} + \sigma_{(i,j)_{n+1}}}{2} f_n \Delta u \qquad (103a) \\
C_{j,i} &= \gamma \frac{g_i}{g_j} \sum_{n=1}^{\mathcal{N}} u_{n+l_{i,j}} \frac{\sigma_{(i,j)_{n+l_{i,j}}} + \sigma_{(i,j)_{n+l_{i,j}+1}}}{2} f_n \Delta u \quad , \qquad (103b)
\end{aligned}
$$

as well as the boundary conditions $\sigma_{(i,j)_n} = 0$ for $n \leq l_{i,j}$, and $f_n = 0$ for $n > \mathcal{N}$ and $n < 1$.

- ionisation collision term
  The term Θ_{ion}/N, relative to the power lost in ionisation collisions is obtained as follows (see equations (78b) and (92))

$$
\begin{aligned}
\frac{\Theta_{i,\text{ion}}}{N} &= \gamma \sum_n \int_{u_{n-1/2}}^{u_{n+1/2}} S_{i,\text{ion}}(u) u du \\
&= \delta_i \gamma \sum_{n=1}^{\mathcal{N}} u_n \left\{ \sum_{m=n+l_{i,\text{ion}}+1}^{2n+l_{i,\text{ion}}} u_m \sigma^{\text{sec}}_{(i,\text{ion})_{(m+1/2, m-n-l_{i,\text{ion}}+1/2)}} f_m \Delta u \right. \\
&+ \left. \sum_{m=2n+l_{i,\text{ion}}}^{\mathcal{N}} u_m \sigma^{\text{sec}}_{(i,\text{ion})_{(m+1/2,n+1/2)}} f_m \Delta u - u_n \sigma_{(i,\text{ion})_{n+1/2}} f_n \right\} \Delta u \\
&= \delta_i \gamma \left\{ \sum_{m=1}^{\mathcal{N}} u_m f_m \left[\sum_{k=1}^{(m-l_{i,\text{ion}})/2} u_{m-k-l_{i,\text{ion}}} \sigma^{\text{sec}}_{(i,\text{ion})_{(m+1/2,k+1/2)}} \Delta u \right] \right. \\
&+ \left. \sum_{m=1}^{\mathcal{N}} u_m f_m \left[\sum_{n=1}^{(m-l_{i,\text{ion}})/2} u_n \sigma^{\text{sec}}_{(i,\text{ion})_{(m+1/2,n+1/2)}} \Delta u \right] - \sum_{n=1}^{\mathcal{N}} u_n^2 f_n \sigma_{(i,\text{ion})_{n+1/2}} \right\} \Delta u \\
&= \delta_i \gamma \left\{ \sum_{m=1}^{\mathcal{N}} u_m f_m \left[\sum_{k=1}^{(m-l_{i,\text{ion}})/2} \left(m - \frac{1}{2} - k + \frac{1}{2} - l_{i,\text{ion}} - \frac{1}{2} \right) \Delta u \, \sigma^{\text{sec}}_{(i,\text{ion})_{(m+1/2,k+1/2)}} \Delta u \right] \right. \\
&+ \left. \sum_{m=1}^{\mathcal{N}} f_m \left[\sum_{n=1}^{(m-l_{i,\text{ion}})/2} u_m u_n \sigma^{\text{sec}}_{(i,\text{ion})_{(m+1/2,n+1/2)}} \Delta u \right] - \sum_{n=1}^{\mathcal{N}} u_n^2 f_n \sigma_{(i,\text{ion})_{n+1/2}} \right\} \Delta u
\end{aligned}
$$

$$
\begin{aligned}
&= \delta_i\gamma\left\{\sum_{m=1}^{\mathcal{N}} u_m^2 f_m\left[\sum_{k=1}^{(m-l_{i,\mathrm{ion}})/2}\sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,k+1/2)}}\Delta u\right] - \sum_{m=1}^{\mathcal{N}} f_m\left[\sum_{k=1}^{(m-l_{i,\mathrm{ion}})/2} u_m u_k \sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,k+1/2)}}\Delta u\right]\right.\\
&- \sum_{m=1}^{\mathcal{N}} u_m f_m\left[\sum_{k=1}^{(m-l_{i,\mathrm{ion}})/2}\left(l_{i,\mathrm{ion}}+\frac{1}{2}\right)\Delta u\,\sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,k+1/2)}}\Delta u\right]\\
&+ \left.\sum_{m=1}^{\mathcal{N}} f_m\left[\sum_{n=1}^{(m-l_{i,\mathrm{ion}})/2} u_m u_n \sigma^{\mathrm{sec}}_{(i,\mathrm{ion})_{(m+1/2,n+1/2)}}\Delta u\right] - \sum_{n=1}^{\mathcal{N}} u_n^2 f_n \sigma_{(i,\mathrm{ion})_{n+1/2}}\right\}\Delta u\\
&= \delta_i\gamma\left\{\sum_{m=1}^{\mathcal{N}} u_m^2 f_m \sigma_{(i,\mathrm{ion})_{m+1/2}} - u_{l_{i,\mathrm{ion}}+1}\sum_{m=1}^{\mathcal{N}} u_m f_m \sigma_{(i,\mathrm{ion})_{n+1/2}} - \sum_{n=1}^{\mathcal{N}} u_n^2 f_n \sigma_{(i,\mathrm{ion})_{n+1/2}}\right\}\Delta u\\
&= -\delta_i u_{l_{i,\mathrm{ion}}+1}\gamma\sum_{m=1}^{\mathcal{N}} u_m f_m \sigma_{(i,\mathrm{ion})_{n+1/2}}\Delta u = -\delta_i C_{i,\mathrm{ion}} V^{\star}_{i,\mathrm{ion}} \qquad (104)\\
\frac{\Theta_{\mathrm{ion}}}{N} &= \sum_i \frac{\Theta_{i,\mathrm{ion}}}{N} = \sum_i -\delta_i C_{i,\mathrm{ion}} V^{\star}_{i,\mathrm{ion}} \quad . \qquad (105)
\end{aligned}
$$

In the previous equations, the ionisation potential is modified by an extra shift of $\Delta u/2$, due to the discretization of the ionisation collisional operator,

$$
V^{\star}_{i,\mathrm{ion}} \equiv u_{l_{i,\mathrm{ion}}+1} = l_{i,\mathrm{ion}}\Delta u + \frac{\Delta u}{2} \quad , \qquad (106)
$$

and we have used the discretized form of the ionisation rate coefficient (see equations (31a) and (103a))

$$
C_{i,\mathrm{ion}} = \gamma\sum_{n=1}^{\mathcal{N}} u_n \sigma_{(i,\mathrm{ion})_{n+1/2}} f_n \Delta u \quad . \qquad (107)
$$

- attachment collision term
  The term Θ_{att}/N, relative to the power lost in attachment collisions is obtained as follows (see equation (78c))

$$
\begin{aligned}
\frac{\Theta_{i,\mathrm{att}}}{N} &= \gamma\sum_n \int_{u_{n-1/2}}^{u_{n+1/2}} S_{i,\mathrm{att}}(u) u du\\
&= -\gamma\sum_{n=1}^{\mathcal{N}} \delta_i \left(u_n\right)^2 \frac{\sigma_{(i,\mathrm{att})_n} + \sigma_{(i,\mathrm{att})_{n+1}}}{2} f_n \Delta u \qquad (108)\\
\frac{\Theta_{\mathrm{att}}}{N} &= \sum_i \frac{\Theta_{i,\mathrm{att}}}{N} \quad . \qquad (109)
\end{aligned}
$$

The term Θ_{coll}/N in equation (94a) is numerically obtained from expressions (97), (100b), (101b), (102b), (105) and (109) as

$$
\frac{\Theta_{\mathrm{coll}}}{N} = \frac{\Theta_{\mathrm{el}}}{N} + \frac{\Theta_{\mathrm{CAR}}}{N} + \frac{\Theta_{\mathrm{ee}}}{N} + \frac{\Theta_{\mathrm{inel}}}{N} + \frac{\Theta_{\mathrm{sup}}}{N} + \frac{\Theta_{\mathrm{ion}}}{N} + \frac{\Theta_{\mathrm{att}}}{N} \quad . \qquad (110)
$$

In stationary and steady-state simulations, the code evaluates the relative power balance equation, dividing (39a) by the *reference power-density* (per electron at unit gas density) Θ_{ref}/N, with the latter defined as the sum of all power-gain terms. Usually, the stationary / steady-state power balance is satisfied to at least $\sim 10^{-10}\%$, and the user can define a threshold value for the relative power balance above which the code returns a warning message (see section 4.2).

In time-dependent simulations, equation (39a) is as accurate as the numerical solution of the EBE, being satisfied to at least $\sim 1\%$ (according to the relative tolerance imposed as parameter to the ode solver, see section 4.2).

7.4 Conservation conditions for the electron-electron collision operator

7.4.1 Energy conservation for electron-electron collisions

Electron-electron collisions should not affect the net gain/loss of energy by the electrons, which means that

$$\begin{aligned}\frac{\Theta_{ee}}{N} &= -\frac{1}{N}\int_0^\infty u\frac{dG_{ee}(u)}{du}du = \frac{1}{N}\int_0^\infty G_{ee}(u)du \\ &= -\gamma\int_0^\infty\left[g_{ee}(u)\left(I(u)f(u)+J(u)\frac{df(u)}{du}\right)\right]du \\ &= -\gamma g_{ee}\left\{\int_0^\infty\left[I(u)f(u)-\frac{dJ(u)}{du}f(u)\right]du+\left[J(u)f(u)\right]_0^\infty\right\}=0 \quad , \end{aligned} \tag{111a}$$

where in the last expression we have used the fact that

$$g_{ee}\propto\frac{u^{3/2}}{v^3}=\text{const} \quad , \tag{111b}$$

and performed an integration by parts of the last term.

The Spitzer integral J, after an integration by parts, can be written as (see (74c))

$$J(u)=\frac{2}{3}\left[\int_0^u f(u')u'^{3/2}du'+u^{3/2}\int_u^\infty f(u')du'\right]=\int_0^u u'^{1/2}\left[\int_{u'}^\infty f(u'')du''\right]du' \quad , \tag{111c}$$

which yields

$$\frac{dJ(u)}{du} = u^{1/2}\int_u^\infty f(u')du' \tag{111d}$$

$$\left[J(u)f(u)\right]_0^\infty = \int_0^\infty u'^{1/2}\left[\int_{u'}^\infty f(u'')du''\right]du'\times f(\infty)-0\longrightarrow 0 \quad , \tag{111e}$$

where we have used Leibnitz law and the fact that the EEDF vanishes at ∞.

Moreover, noting that the derivative of the Spitzer integral I yields (see (74b))

$$I(u)=\int_0^u f(u')u'^{1/2}du' \Longrightarrow \frac{dI(u)}{du}=f(u)u^{1/2} \quad , \tag{111f}$$

one can write using (111d),

$$\begin{aligned}\int_0^\infty\frac{dJ(u)}{du}f(u)du &= \int_0^\infty f(u)u^{1/2}\int_u^\infty f(u')du'du \\ &= \int_0^\infty\frac{dI(u)}{du}\int_u^\infty f(u')du'du \\ &= \left[I(u)\int_u^\infty f(u')du'\right]_0^\infty+\int_0^\infty I(u)f(u)du \quad , \end{aligned} \tag{111g}$$

where the last expression was obtained with an integration by parts.

Finally, using (111e) and (111g) in (111a) one obtains

$$\begin{aligned}\frac{\Theta_{ee}}{N} &= -\gamma g_{ee}\left\{\int_0^\infty\left[I(u)f(u)-I(u)f(u)\right]du-\left[I(u)\int_u^\infty f(u')du'\right]_0^\infty\right\} \\ &= \gamma g_{ee}\left\{\int_0^\infty f(u)u^{1/2}du\times 0-0\times\int_0^\infty f(u)du\right\}\longrightarrow 0 \quad , \end{aligned} \tag{111h}$$

thus confirming the result of equation (111a).

Numerically, this result introduces a relationship between the matrix terms of the electron-electron collision operator. From (100b) and (76a)- (76b), one has

$$\gamma \sum_{n=1}^{\mathcal{N}} (A_n - B_n) f_n (\Delta u)^2 = 0 \implies \gamma \sum_{n=1}^{\mathcal{N}} \left[\sum_{m=1}^{\mathcal{N}} a_{n,m} - \sum_{m=1}^{\mathcal{N}} b_{n,m} \right] f_m f_n (\Delta u)^2 = 0$$
$$\implies \sum_{n,m=1}^{\mathcal{N}} (a_{n,m} - b_{n,m}) f_m f_n (\Delta u)^2 = 0 \quad , \tag{112a}$$

which means that the matrix multiplying the EEDFs should be antisymmetric satisfying

$$a_{n,m} - b_{n,m} = -a_{m,n} + b_{m,n} \tag{112b}$$

or

$$a_{n,m} - b_{m,n} = -a_{m,n} + b_{n,m} = 0 \Longrightarrow b_{m,n} = a_{n,m} \quad . \tag{112c}$$

7.4.2 Maxwellian solution for collisional equilibrium

Under the exclusive action of electron-electron collisions, the EBE (7a), (8a) or (19a) writes

$$\frac{dG_{ee}(u)}{du} = 0 \implies G_{ee}(u) = 0$$
$$\implies g_{ee}(u) \left(I(u) f(u) + J(u) \frac{df(u)}{du} \right) = 0 \quad , \tag{113a}$$

where we have used the boundary conditions $G_{ee}(0) = G_{ee}(\infty) = 0$ and equation (75).

Equation (113a) can be rewritten using (74b), (111b) and (111c) to obtain

$$f(u) \int_0^u u'^{1/2} f(u') du' + \frac{df(u)}{du} \int_0^u u'^{1/2} \left[\int_{u'}^{\infty} f(u'') du'' \right] du' = 0 \quad . \tag{113b}$$

This identity is satisfied if f is a Maxwellian distribution function, since in this case

$$f(u) = a \int_u^{\infty} f(u') du' \Longrightarrow \frac{df(u)}{du} = -a f(u) \Longrightarrow f(u) = C \exp(-au) \quad , \tag{113c}$$

with a and C constants.

Numerically, this equilibrium situation implies that (see (75) and (112c))

$$\sum_{m=1}^{\mathcal{N}} (a_{n-1,m} f_m) f_{n-1} - \sum_{m=1}^{\mathcal{N}} (a_{n,m} f_m + b_{n,m} f_m) f_n + \sum_{m=1}^{\mathcal{N}} (b_{n+1,m} f_m) f_{n+1} = 0$$
$$\Longrightarrow \sum_{m=1}^{\mathcal{N}} (a_{n-1,m} f_m) f_{n-1} - \sum_{m=1}^{\mathcal{N}} (a_{n,m} f_m + a_{m,n} f_m) f_n + \sum_{m=1}^{\mathcal{N}} (a_{m,n+1} f_m) f_{n+1} = 0$$
$$\Longrightarrow \sum_{m=1}^{\mathcal{N}} a_{n-1,m} f_m f_{n-1} - \sum_{m=1}^{\mathcal{N}} a_{n,m} f_m f_n - \sum_{m=1}^{\mathcal{N}-1} a_{m,n} f_m f_n + \sum_{m=1}^{\mathcal{N}-1} a_{m,n+1} f_m f_{n+1} = 0 \tag{114a}$$
$$\Longrightarrow \sum_{m=1}^{\mathcal{N}} a_{n-1,m} f_m f_{n-1} - \sum_{m=2}^{\mathcal{N}} a_{m-1,n} f_{m-1} f_n - \sum_{m=1}^{\mathcal{N}} a_{n,m} f_m f_n + \sum_{m=2}^{\mathcal{N}} a_{m-1,n+1} f_{m-1} f_{n+1} = 0$$
$$\Longrightarrow \sum_{m=1}^{\mathcal{N}} [a_{n-1,m} f_m f_{n-1} - a_{m-1,n} f_{m-1} f_n] - \sum_{m=1}^{\mathcal{N}} [a_{n,m} f_m f_n - a_{m-1,n+1} f_{m-1} f_{n+1}] = 0 \quad ,$$

where we have used the boundary conditions $A_0 = A_{\mathcal{N}} = 0$ or, equivalently, $a_{0,k} = a_{\mathcal{N},k} = 0$, and where the EEDFs are Maxwellian distribution functions satisfying

$$f_{m-1} f_{n+1} = C^2 \exp(-a u_{m-1}) \exp(-a u_{n+1})$$
$$= C^2 \exp(-a u_m) \exp(-a u_n) \exp(a \Delta u) \exp(-a \Delta u)$$
$$= f_m f_n \quad . \tag{114b}$$

Equations (114a)-(114b) allow defining a relationship between the upper/lower matrix terms a_{mn} of the electron-electron collision operator

$$a_{n,m} = a_{m-1,n+1}\,,\ 1 \le n \le \mathcal{N}-1 \quad ;\, 1 \le m \le \mathcal{N}$$

or to balance out the contributions from the upper/lower matrix terms

$$a_{n,m} = \sqrt{a_{n,m} \times a_{m-1,n+1}}\,,\ 1 \le n \le \mathcal{N}-1 \quad ;\, 1 \le m \le \mathcal{N}\,. \tag{115}$$

7.5 Solution of LoKI-B

After discretisation, the isotropic part of the EBE can be written as $\sum_{m=1}^{\mathcal{N}} C_{n,m}(f) f_m = 0$, which is solved coupled to the normalisation condition $\sum_{m=1}^{\mathcal{N}} f_m \sqrt{u_m}\Delta u = 1$.

The matrix $C(f)$ is nonlinear due to the presence of the following terms.

- The temporal/spatial growth-term of the electron density, associated with non-conservative mechanisms, involves either the effective ionisation rate coefficient $\langle\nu_{\text{eff}}\rangle/N$ or the reduced-effective Townsend coefficient α_{eff}/N, both being functions of the EEDF (see section 2.1).
- The expression of the electron-electron upflux G_{ee} involves linear functions of the EEDF, as confirmed by (12d) and (76a)-(76b).

The EBE is solved as a balance equation, written as in (38a), (38b), (38c), (38d), using the solution methods indicated below (see section 4.2 for details about the numerical calling of the solution methods)

- direct solution method (successively applied in the presence of nonlinear terms), using `mixingDirectSolutions` as `algorithm`
  - for steady-state simulations with a temporal growth model (equation (38a))
  - for steady-state simulations with a spatial growth model (equation (38b));
- relaxation method, using `temporalIntegration` as `algorithm`
  - for steady-state simulations with temporal growth model (equation (38a))
  - for time-dependent simulations with temporal or spatial growth models (equations (38c), (38d)).

For electropositive gases, when choosing `ionizationOperatorType` as `conservative` (see sections 1.2 and 4.2), the first term on the right-hand side of equations (38a), (38b), (38c), (38d) is set to zero.

The solution method adopted in solving the EBE depends also on its linear/nonlinear features.

- In the absence of nonlinear effects (*e.g.*, for electropositive gases, with the ionisation taken as a conservative mechanism and no electron-electron collisions), the EBE is solved adopting a very effective direct solution method. In this case the first equation of the Boltzmann system is added to the normalisation condition of the EEDF, *i.e.* $\sum_{m=1}^{\mathcal{N}} C_{1,m} f_m = \sum_{m=1}^{\mathcal{N}} \sqrt{u_m}\Delta u f_m = 1$, and the resulting non-homogeneous equation system becomes ready for direct inversion.
- When the matrix $C(f)$ contains nonlinear terms, the EBE is solved using an iterative method.
  - In the case of weak nonlinear effects (*e.g.*, low electronegativity and low ionisation degree), the solution can be obtained by using two iterative algorithms, based on successive direct solutions. The algorithm is represented in the simplified flowchart of figure 5, and it adopts:
    - a mixing of solutions to converge over the temporal growth parameter $\langle\nu_{\text{eff}}\rangle/N$ or the spatial growth parameter α_{eff}/N (see sections 2.1 and 7.2). In this case the convergence criterion looks for relative differences, between two consecutive iterations, below 10^{-10} for the growth parameter and below a user-prescribed value (typically 10^{-9}) for the EEDF;
    - a Newton-Raphson-based approach to converge over the electron-electron collision operator. In this case the convergence criterion looks for relative differences, between two consecutive iterations, below $\sim 10^{-9} - 10^{-10}$ for the ratio of the power "dissipated" in electron-electron collisions to the reference power $\Theta_{ee}/\Theta_{\text{ref}}$ (see section 2.2 for the definition of Θ_{ref}), and below a user-prescribed value (typically 10^{-9}) for the EEDF.

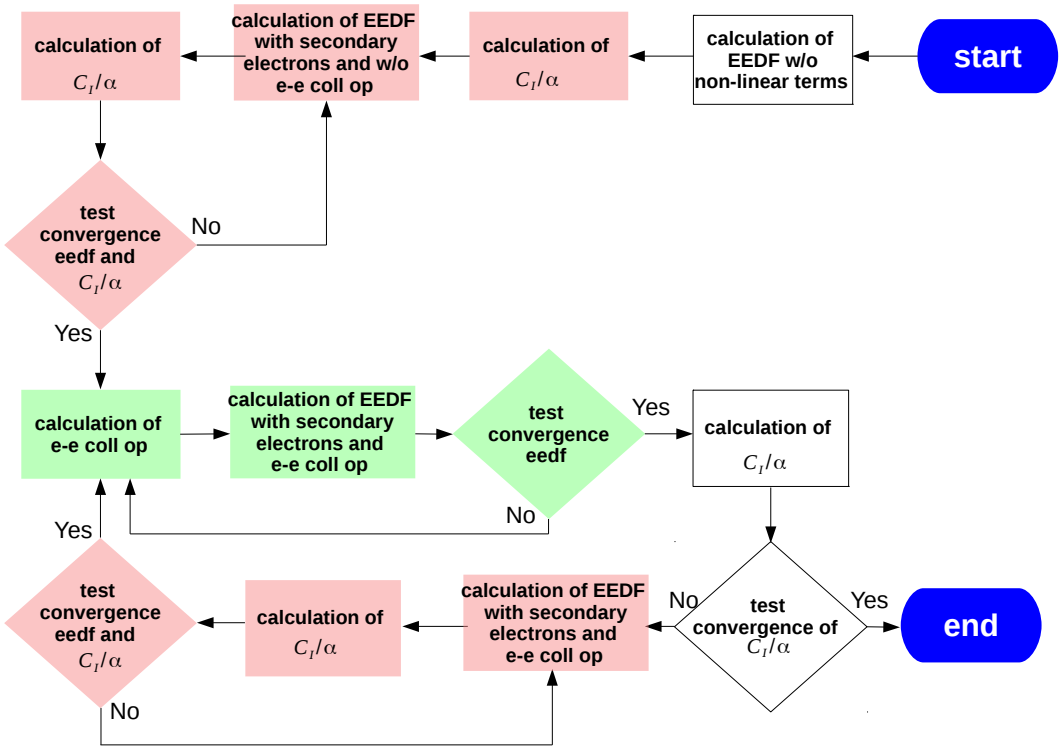


Figure 5: Simplified flowchart of LoKI-B, in the case of weak nonlinear effects. In red are the steps done within the ionisation routine. In green are the steps done within the electron-electron collisions routine.

When both algorithms are needed, the code ensures that the relative differences between the final EEDFs, obtained in each of the algorithms, are below a user-prescribed value for the EEDF.

– in the presence of strong nonlinear effects, the user is recommended to choose a relaxation method. In this case the EBE is solved with a time-dependent algorithm, using MATLAB's ordinary differential equation solver `ode15s` [45], and the user can specify the (optional) parameters contained in the structure of the solver, *e.g.* maximum step-size (typical value 10^{-7}). The default convergence criterion demands relative variations of the EEDF below a user-prescribed value (typically 10^{-9}), between two consecutive iterations.

The solution of the time-dependent EBE adopts also a relaxation method, using MATLAB's ordinary differential equation solver `ode15s` [45]. When electron-electron collisions are included in the calculations, LoKI-B can solve the system of equations

$$\frac{\partial f(t)}{\partial t} = C[f(t)]f(t) \tag{116a}$$

$$\frac{dn_e(t)}{dt} = \langle \nu_{\text{eff}}(t) \rangle n_e(t) \quad , \tag{116b}$$

subjected to some initial conditions $f(t=0) \equiv f_0$ and n_0. **Although this is implemented in LoKI-B, the option is not available by default. If needed, it can be easily activated by the user directly in the code.** The solver adopts a variable-step, variable-order algorithm suited for stiff problems, based on a set of highly-stable implicit integration formulas of orders 1 to 5 (optionally, it can also use the backward differentiation formulas). Specifically, the integrator estimates the error at each iteration and then adjusts the timestep Δt to improve either the accuracy (*i.e.* by decreasing the timestep) or the efficiency (*i.e.* by increasing the timestep). At each time t the solution is updated using $f(t) = f(t-\Delta t) + (\partial f(t)/\partial t)\Delta t$ and $n_e(t) = n_e(t-\Delta t) + (dn_e(t)/dt)\Delta t$. Again, the user can specify the (optional) parameters contained in the structure of the solver, *e.g.* absolute tolerance (typical value 10^{-300}), relative tolerance (typical value 10^{-6}) and maximum step-size (typical value 10^{-7}), which are applied by the solver to adjust Δt for each instant t. The relative tolerance works as convergence criterion, imposing the maximum value of the relative variations of the EEDF, between two consecutive iterations.

8 Validity limits and control

As any numerical tool, LoKI-B provides a mere representation of the energy distribution of the electrons in a LTP, because it is based on a specific formulation with embedded approximations. Therefore, the tool should be used only in situations fitting its validity limits, and users are advised to carefully read the extensive literature on the two-term EBE [14], for obtaining details about its solution.

The two-term approximation adopted in LoKI-B to solve the EBE is valid in a situation of small anisotropies [14], corresponding to electron–neutral mean-free-paths much smaller than any characteristic dimension of the plasma container, and an energy gain from the electric field, between collisions, much smaller than the electron kinetic energy. For a homogeneous scenario like the one described here, the latter condition is the most relevant and can be approximately written as (see (32a) and (40c))

$$\Theta_E \sim \mu_e E^2 \ll \varepsilon \nu_c \quad , \tag{117}$$

which yields, for a Maxwellian EEDF at 1 eV temperature and a typical reduced collision frequency $\nu_c/N \sim \sigma_c \langle v \rangle \sim 10^{-19} \times 10^6 = 10^{-13}$ m^3 s^{-1}, $E/N \ll \sqrt{m_e \varepsilon / e}(\nu_c/N) \sim 300$ Td. This is a conservative limit for the reduced electric-field, often exceeded in calculations because the two-term approximation yields fairly good results still.
However, users are advised to check the EEDF results when taking E/N values much above the estimated validity limit, namely by comparing the solution obtained for the isotropic and the anisotropic parts of the electron distribution function.

Also, HF calculations are valid only if the oscillation frequency of the electric field is much larger than the typical frequency for the electron energy relaxation, which can be approximately written as (see (12b))

$$\omega \gg (m_e/M) N (\nu_c/N) \quad , \tag{118}$$

corresponding to $\omega/(2\pi) \gg 20$ MHz - 20 kHz for $\nu_c/N \sim 10^{-13}$ m^3 s^{-1}, $m_e/M \sim 5 \times 10^{-5}$, $T_g = 300$ K and $p \sim 10^5$ Pa - 130 Pa.
Again, users are advised to check the validity of the HF approximation for the particular working conditions considered.

LoKI-B contains several control features upon the input data and the calculation results, and in some cases it **returns warning messages** for the benefit of users. In particular:

- the code controls the maximum value $u_{\max}$ of the energy grid adopted, comparing it with the cutoff energy u_{cutoff} of the electron-neutral elastic momentum-transfer cross section.
  When $u_{\text{cutoff}} < u_{\max}$ a warning message is returned, and if the user accepts to proceed with the calculations these are carried out by setting to zero the values of any cross section from its corresponding u_{cutoff} and up to $u_{\max}$. **LoKI-B does not extrapolate cross sections beyond the cutoff energies defined in the input data-files**, again because it adopts an ontology that separates the tool and the data. The user is recommended to always careful examine the cross sections being used and, if needed, he/she should adopt adequate laws to extrapolate them, chosen according to the specific processes considered (*e.g.* dipole allowed or dipole forbidden electron excitation, ionisation, etc);

- the code **returns a warning message** if the calculation of the elastic momentum-transfer cross section from the corresponding effective **produces negative values for** $\sigma_{k,c}^{\text{el}}$ (see section 4.2), which is then set to zero. **In this case, users are advised to check the cross section data adopted**;

- **in stationary and steady-state simulations**, the code controls the quality of the Boltzmann-matrix inversion, by monitoring the error upon the fractional electron power-balance. The latter is obtained from (39a), by normalizing this equation with respect to a *reference power-density* Θ_{ref},**defined for each calculation as the sum of all power-gain terms** (see section 2.2).
  The relative error $\left[\Theta_E + \Theta_{\text{coll}}^{\text{gain}} + \Theta_{\text{coll}}^{\text{loss}} + \Theta_{\text{growth}}\right]/\Theta_{\text{ref}}$ **(with typical values $\sim 10^{-9} - 10^{-10}$, and user-defined `maxPowerBalanceRelError` threshold for producing a warning message, see section 4.2) degrades for very-low E/Ns**, due to the smallness of the power-density values for all the power-transfer channels, **and for very-high E/Ns**, if the $u_{\max}$ grid-limit adopted in the calculations is not large enough to prevent ill-conditioned EBE-matrices.

Concerning cross section data, the user is further advised to carefully check

- if the electron-scattering cross sections are set to zero at the threshold energy, as it is coherently assumed by LoKI-B. Note that, excluding errors, the LXCat files obtained from the IST-Lisbon database comply with this recommendation;
- the maximum value $u_{\max}$ of the energy grid adopted, comparing it with the highest threshold $u_{\mathrm{th-max}}$ of the collisional cross sections associated with reactions for which superelastic mechanisms are activated. Note that $u_{\max} \gg u_{\mathrm{th-max}}$ should be ensured, to properly account for the superelastic effects.

9 Comparison between LoKI-B++ and LoKI-B

LoKI-B MATLAB and LoKI-B++ aim to produce equal results. However, there are some differences in how the codes can be used, and the features they support. This section aims to clarify these differences.

9.1 General differences

There are a few general differences between the two codes. First and foremost, they are written in different languages and thus require different methods of running the code. Section 3.2 provides information on how to obtain and run LoKI-B++. What follows is a list of differences between the two codes. Second, LoKI-B++ does not currently come with a local graphical user interface, although a preliminary web interface is available. Therefore, it is typically run in the browser, or from the command line. There are some additional differences in terms of supported output formats. LoKI-B++ supports JSON-based output, whereas LoKI-B MATLAB supports HDF5 output. Finally, LoKI-B++ is almost at feature parity with LoKI-B MATLAB. Two routines are currently missing, support for anisotropic inelastic collisions as mentioned in Section 4.2, and a routine to solve the time-dependent electron Boltzmann equation introduced in Section 1.3.

9.2 Differences in the input file

There are two main differences in the input file structures of LoKI-B MATLAB and LoKI-B++. First, as LoKI-B++ does not currently have a local graphical user interface, the `gui` section of the input file is superfluous, and it can be omitted. Second, the different output formats of LoKI-B are still in active development, which leads to some differences in how the `output` section is configured. Code 9 shows the `output` section of a LoKI-B++ input file with the available configuration options.

```
output:
  isOn: true                  % [Boolean] Whether to enable output collection.
  writeJSON: true             % [Boolean] Whether to write JSON output.
  JSONFile: output-file.json  % [String] The file to write the JSON output to
                              % (required when ‘writeJSON‘ is ‘true‘).
  writeText: true             % [Boolean] Whether to write text-based output.
  folder: default-temporal    % [String] The folder to write the test files to
                              % (required when ‘writeText‘ is ‘true‘).
  dataFiles:                  % [Array of String] Defines the type of text-based
    - log                     % output files that are written.
    - eedf
    - swarmParameters
    - rateCoefficients
    - powerBalance
    - lookUpTable
```

Code 9: The ‘output‘ section of a LoKI-B++ input file.

9.3 Quantitative comparison of swarm parameters

The output of LoKI-B MATLAB and LoKI-B++ has been compared and validated for all included input files. The absolute relative error in output parameter p is defined as

$$\mathrm{ARE}[p] = \left| \frac{p^{\mathrm{MATLAB}} - p^{\mathrm{C++}}}{p^{\mathrm{MATLAB}}} \right|, \tag{119}$$

where p^{MATLAB} is the value of p produced by LoKI-B MATLAB, and $p^{\mathrm{C++}}$ is the value of p produced by LoKI-B++. The error metric for comparing EEDFs is the point-to-point mean absolute relative error,

$$\mathrm{MARE}[f] = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{f_i^{\mathrm{MATLAB}} - f_i^{\mathrm{C++}}}{f_i^{\mathrm{MATLAB}}} \right|, \tag{120}$$

where N is the number of cells in the simulation domain.

Figures 7 to 12 present the results. LoKI-B MATLAB and LoKI-B++ show excellent agreement up to a maximum ARE of $\approx 10^{-10}$ in the EEDF and swarm parameters. Note that these results cover many different gases, collision types, temporal growth, and spatial growth across a wide range of E/N.

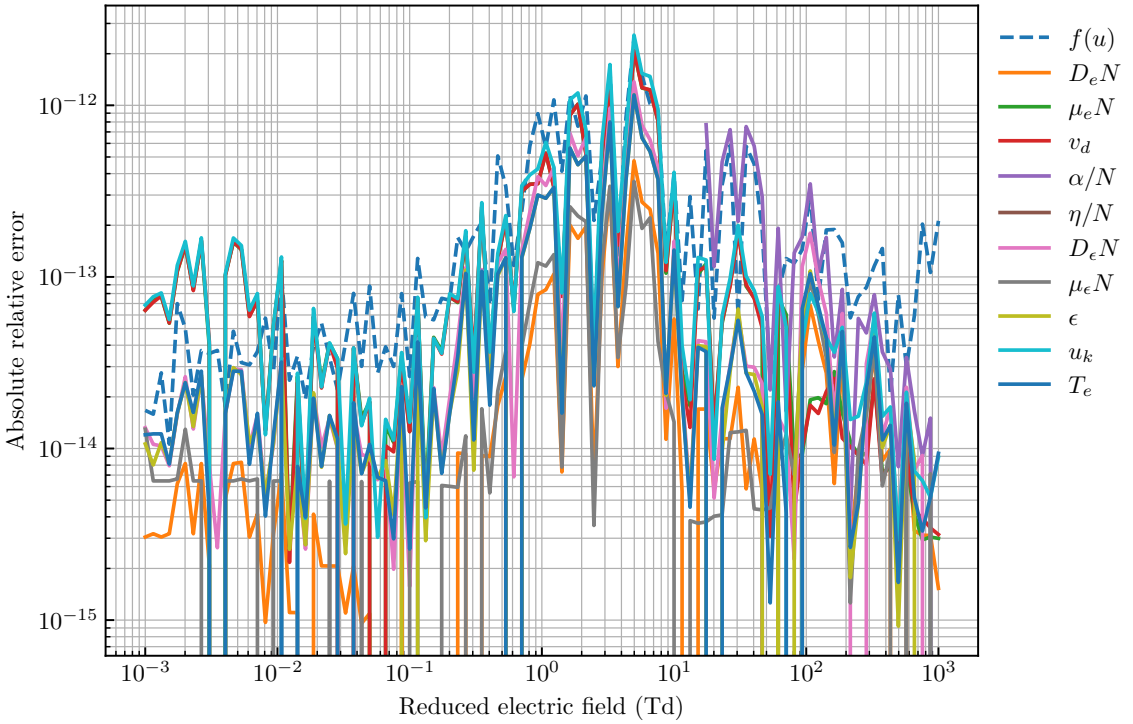


Figure 6: A comparison of N_2 output parameters between LoKI-B MATLAB and LoKI-B++ for the default input file `default_lokib_input.in`.

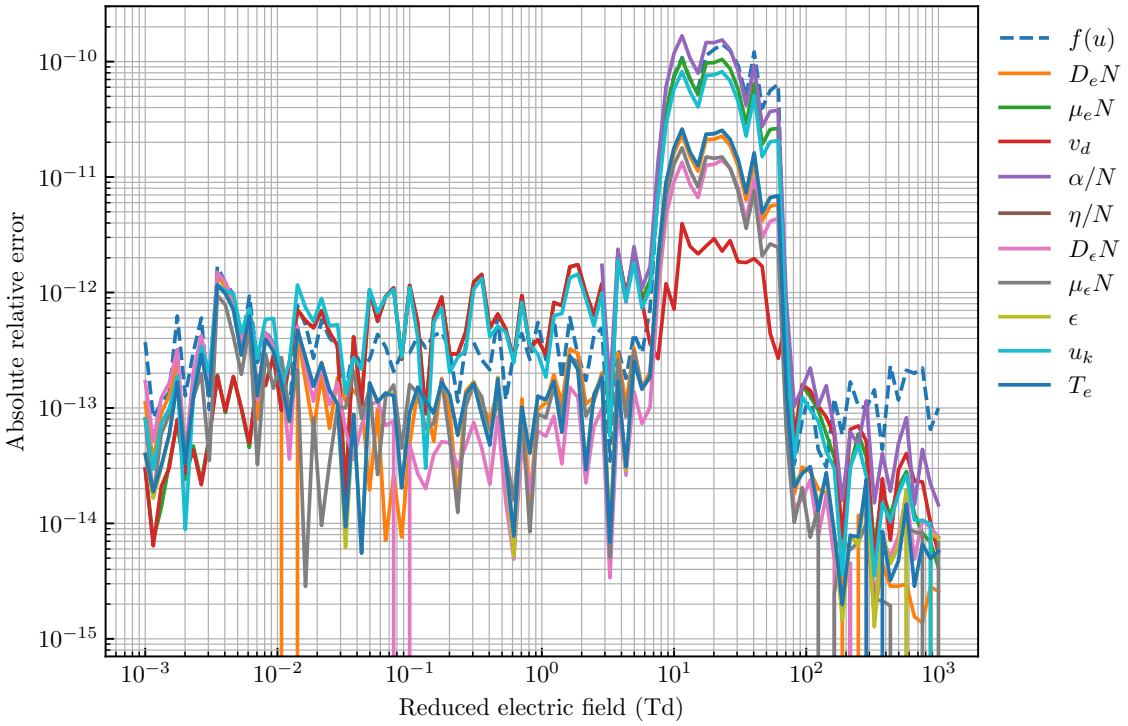


Figure 7: A comparison of Ar output parameters between LoKI-B MATLAB and LoKI-B++ for the `Ar_swarm_setup.in` input file.

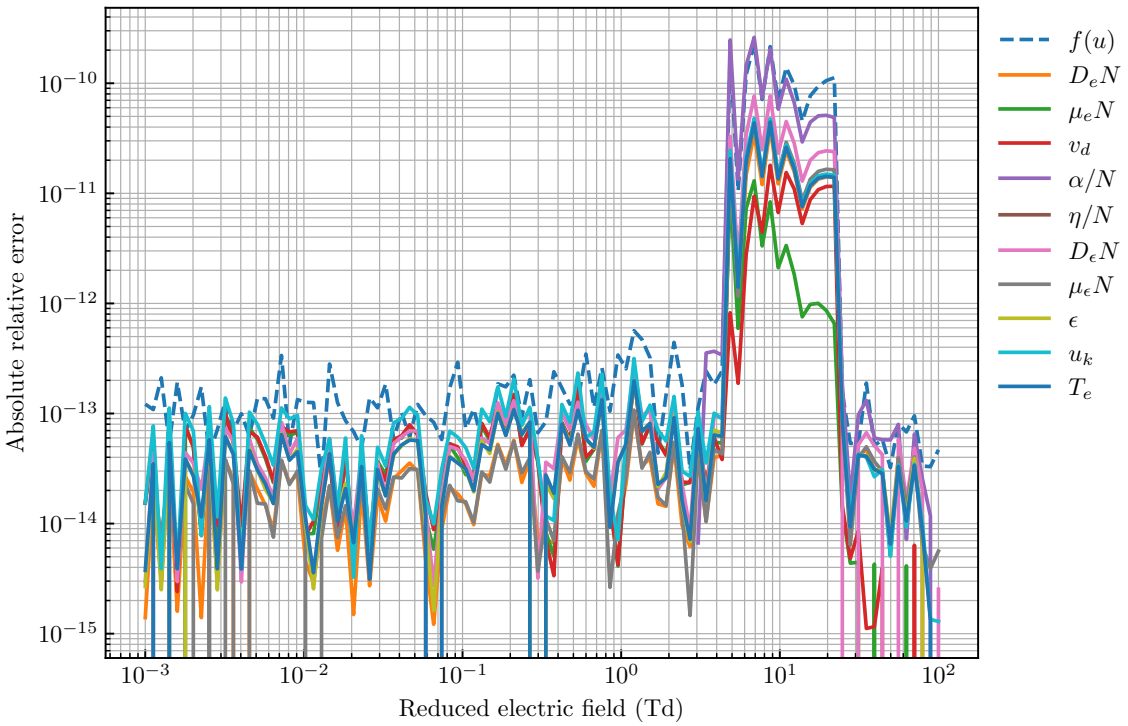


Figure 8: A comparison of He output parameters between LoKI-B MATLAB and LoKI-B++ for the `He_swarm_setup.in` input file.

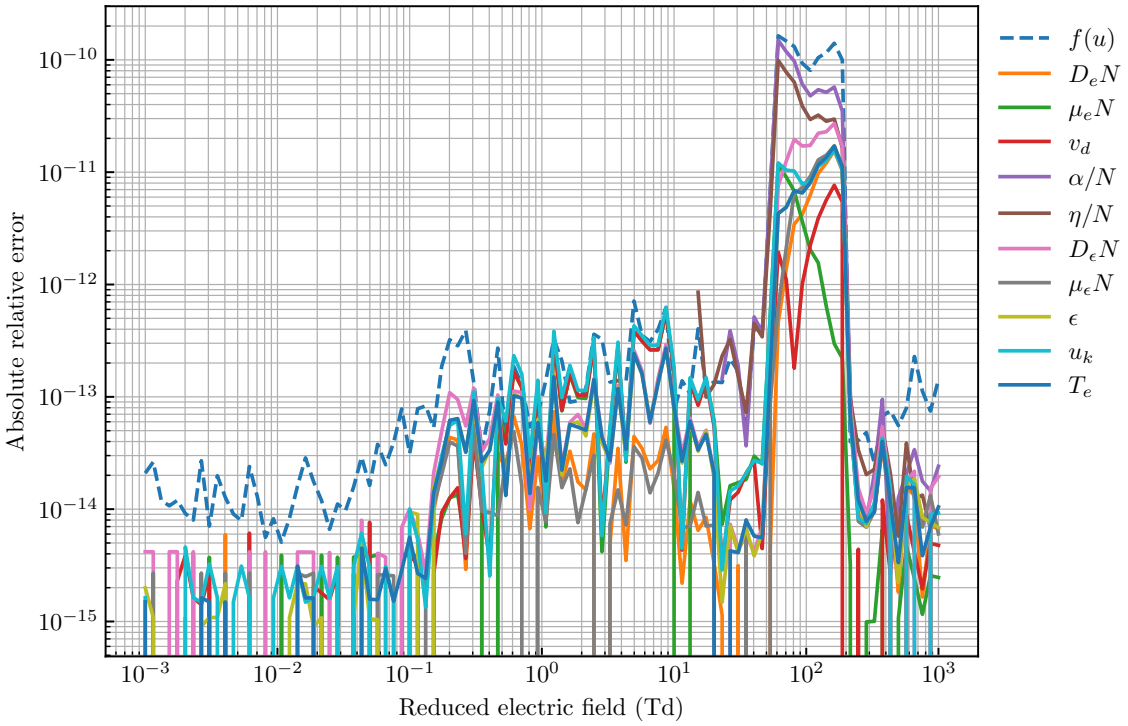


Figure 9: A comparison of CO output parameters between LoKI-B MATLAB and LoKI-B++ for the `CO_swarm_setup.in` input file.

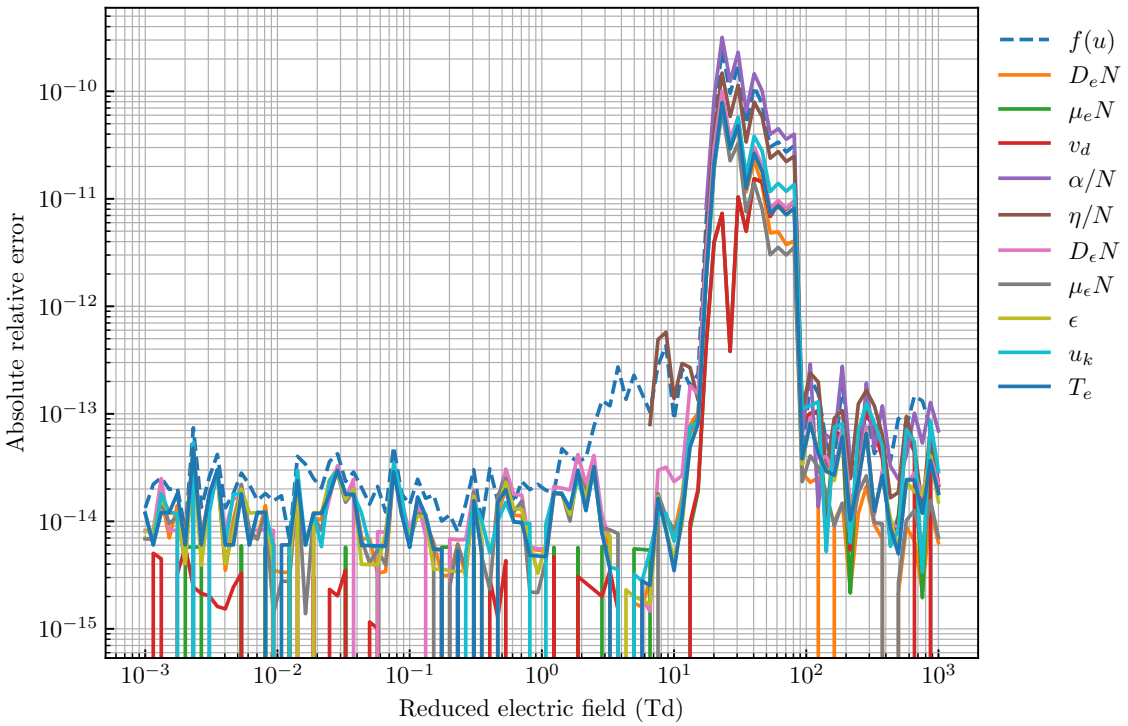


Figure 10: A comparison of CO2 output parameters between LoKI-B MATLAB and LoKI-B++ for the `CO2_swarm_setup.in` input file.

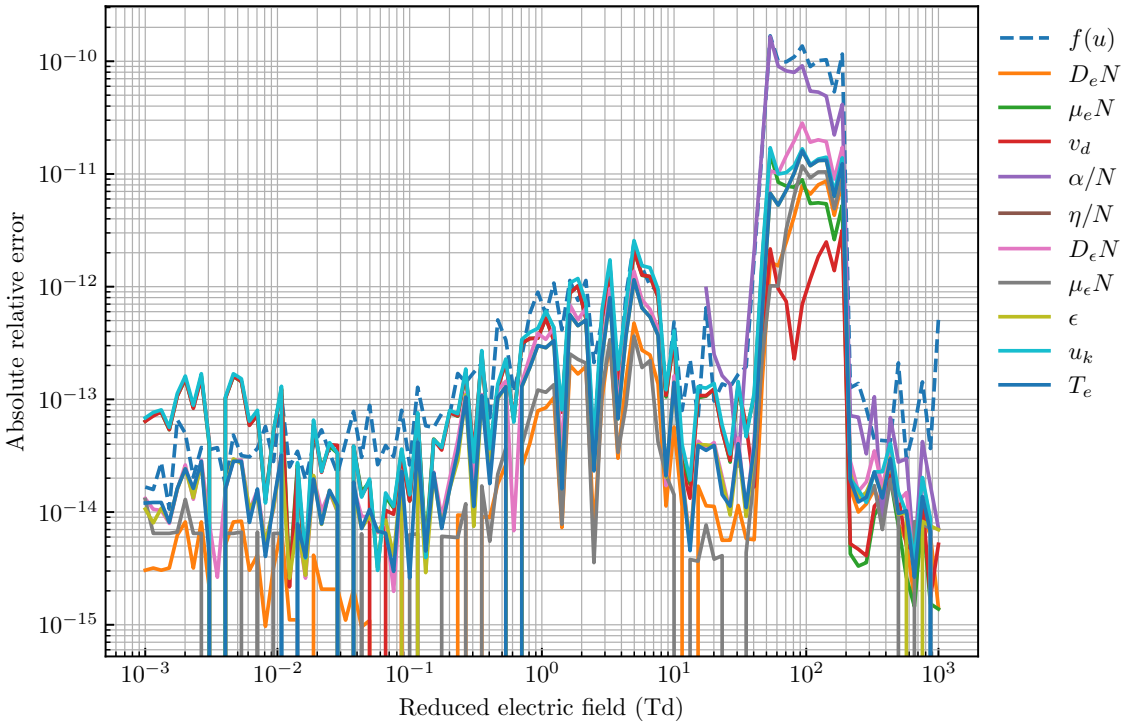


Figure 11: A comparison of N2 output parameters between LoKI-B MATLAB and LoKI-B++ for the `N2_swarm_setup.in` input file.

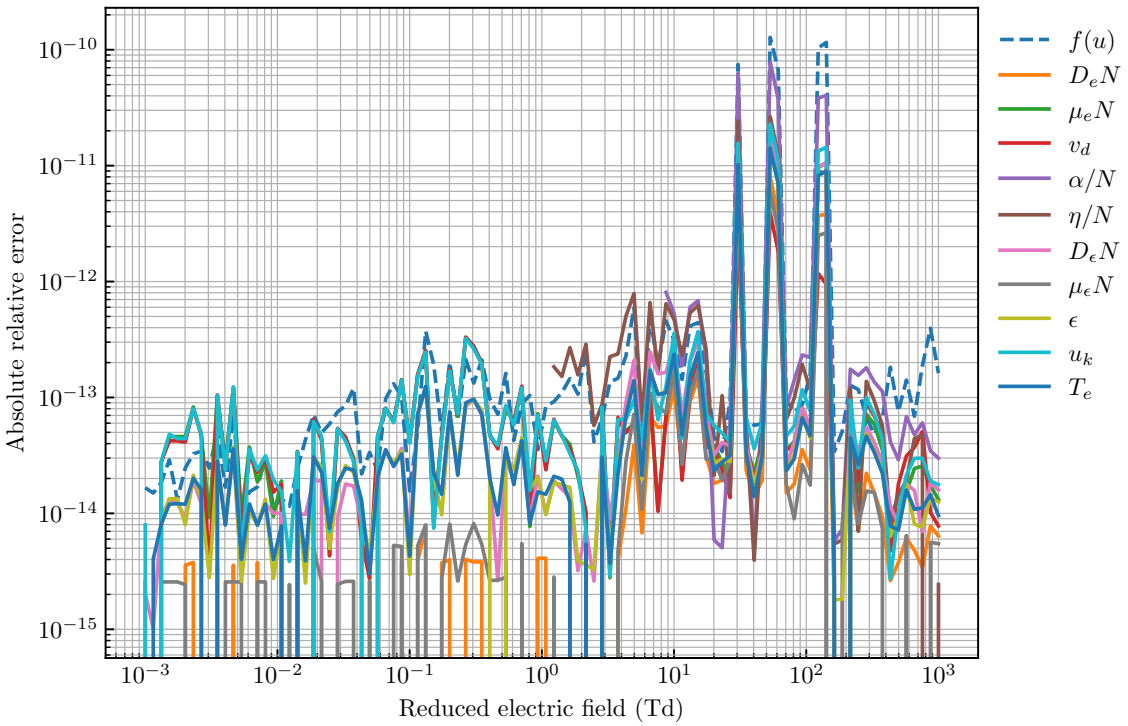


Figure 12: A comparison of O2 output parameters between LoKI-B MATLAB and LoKI-B++ for the `O2_swarm_setup.in` input file.

10 Changelog

This is probably not very interesting for the ordinary user. But since you have made it this far, please take a moment to read a brief history of the code since its first release.

- `LoKI-B_v25.10.1` (2025/09/25):
  - IMPORTANT NEW FEATURE: initial support in JSON format, for auxiliary LXCat files and input files.
  - IMPORTANT NEW FEATURE: definition of elastic collisions extended to electronic/vibrational/rotational states, without any restriction other than avoid duplicated or missing information.
  - IMPORTANT NEW FEATURE: added support for output in HDF5 format, with changes in some fields of the input file. For more information see section 4.4.
  - NEW FEATURE: generalizing the writing of thresholds in cross section files using scientific notation with uppercase (E) descriptors (before, only lowercase (e) descriptors were supported).
  - NEW FEATURE: automatic indexing to maximum power-balance error in the convergence of e-e collisions.
  - NEW FEATURE: added property function to calculate the Boltzmann population of orto/para rotational states of H_2.
  - BUG FIX: throw an error when effective cross sections are provided for several electronic states' siblings of the same gas.
  - BUG FIX: correction of function `Setup.m` to meet changes in MATLAB's function `ind2sub`, which no longer supports scalar input for array size (after R2024a).
  - BUG FIX: multiplication of the `prescribedEedf` expression by the "shape" parameter was missing.
  - BUG FIX: correction of (i) the discretization of the electron density spatial-growth operator; (ii) evaluation of the electron mobility in the case of spatial growth, consistent with (i) and with its evaluation in the `evaluateSwarmParameters` function; (iii) evaluation of the power lost due to the electron density spatial growth, consistent with (i) and with the calculation of the power using swarm parameters, as evaluated in the `evaluateSwarmParameters` function.
  - BUG FIX: solved error in the function that solves e-e collisions. When evaluating the A matrix, both old and new values of the matrix elements were used to impose detailed balance.
  - BUG FIX: corrections in the GUI related to `prescribedEedf`.
  - BUG FIX: solved error preventing LXCat extra files to be saved as output.
  - BUG FIX: solved error in the checking of the availability of elastic cross sections.
  - UPDATE: in cross section files for N-atoms, including superelastic mechanisms from N(2P,2D).
  - A few minor changes to improve stability and performance of the code.
- `LoKI-B_v2.2.0` (2022/11/04):
  - NEW FEATURE: general implementation of anisotropic e-impact collisions.
  - NEW FEATURE: proper implementation of hf simulations for prescribed eedf .
  - BUG FIX: added checking to avoid e-impact collisions types non-supported by LXCat.
  - BUG FIX: improve the evaluation of power in electron-electron collisions.
  - BUG FIX: solved error in the temporal growth operator, when using the time-integration algorithm.
  - A few minor changes to improve stability and performance of the code.
- `LoKI-B_v2.1.0` (2022/03/04):
  - NEW FEATURE: the DC reduced mobility is always evaluated and the (complex) HF mobility is only provided in the case of oscilating fields.
  - NEW FEATURE: added the possibility of selecting lin/log scale for both axis on the graphs of the different swarm parameters. For the power graph only the xaxis can be toggled between lin/log scale.

- BUG FIX: solved important error in the e-e collision (`mixingDirectSolutions` scheme only) operator related with the units of the reduced field
- BUG FIX: now attachment collisions are not considered for the evaluation of a reduced field from power balance when using `PrescribedEedf`.
- BUG FIX: activated the renormalisation of the eedf for the calculation of the eedf derivative.
- BUG FIX: solved an inconsistency in the evaluation of the reference power in `PrescribedEedf` with respect to `Boltzmann` class.
- Added the possibility of providing thresholds with scientific notation in LXCat files.
- A few minor changes to improve stability and performance of the code.

- `LoKI-B_v2.0.1` (2021/09/20):
  - BUG FIX: solved error in the clipping of negative values of an elastic cross section (when evaluated from an effective one) to zero.

- `LoKI-B_v2.0.0` (2021/02/14):
  - NEW FEATURE: Added the option to perform pulsed simulations (see sections 1.3 and 4).
  - NEW FEATURE: Added two new parameters calculated on output, the energy diffusion coefficient and the energy mobility (see section 2).
  - IMPORTANT CHANGE: Structure of property functions have changed from the previous version to accommodate new features. For more information see section 6.1.
  - BUG FIX: Solved small error in the implementation of the CC-CAR operator (see section 1.2).
  - BUG FIX: Solved an error in the evaluation of both the Townsend coefficient and the attachment coefficient (the code now accounts for more than one of such collisions per gas).
  - BUG FIX: Solved an inconsistency between the code and the documentation related with the evaluation of the reduced mobility in HF simulations The code now evaluates the real part of the HF mobility in these simulations (see section 2).
  - BUG FIX: Solved some inconsistencies in the evaluation of the power from the field when non linear operators were activated resulting in a wrong evaluation of the power balance in some cases.
  - Removed deprecated class `Maxwellian`.
  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v1.0.0` (2019/03/20):
  - NEW FEATURE: The user can now use a time integration algorithm to solve the non-linear operators. For more information see section 4.2.
  - NEW FEATURE: Added attachments collisions as non-conservative mechanisms.
  - NEW FEATURE: Added power transfer plot to the GUI.
  - NEW FEATURE: Added evaluation and output of the elastic/effective collision rate coefficients.
  - NEW FEATURE: Added attachment collisions as non-conservative mechanism (when they are present).
  - NEW FEATURE: Added the possibility for the user to change the scale of the EEDF graph in the GUI (linear/log).
  - NEW FEATURE: Added evaluation of the first anisotropy
  - IMPORTANT CHANGE: The reference power is now considered as the total gained power from all possible sources.
  - IMPORTANT CHANGE: The configuration of the electron kinetics on the setup file has been slightly modified in order to accommodate new features. For more information see section 4.2.
  - IMPORTANT CHANGE: Changed the "Maxwellian" EEDF type by "prescribedEedf" (generalised expression with Maxwellian/Druyvesteyn as particular cases). For more information see section 4.2.
  - BUG FIX: fixed a problem that caused the results of the first job of a simulation to be saved outside its subfolder.
  - BUG FIX: fixed small error in the morse oscillator energy formula.

- A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.6.3` (2018/03/26):
  - BUG FIXED: corrected a problem with the Klein-Rosseland expression in the evaluation of superelastic cross sections. The bug only affected the evaluation of an elastic cross section from an effective one, and not the actual Boltzmann calculations. However, results obtained with previous versions of the code may be greatly affected (when using "effective" cross sections and superelastic mechanisms) hence it is recommended to repeat calculations with this new version.
  - BUG FIXED: fixed a problem with the detailed balance in the e-e collisions. This bug only introduced a small error that affected convergence of the non-linear algorithm.

- `LoKI-B_v0.6.2` (2018/01/18):
  - BUG FIXED: corrected some expressions used in the evaluation of the swarm parameters, for the cases when the electron density growth is activated.
  - BUG FIXED: fixed a problem with the Maxwellian type of EEDF, for which the code used to crash when trying to output (GUI or file) the power losses due to the electron density growth.

- `LoKI-B_v0.6.1` (2018/01/16):
  - BUG FIXED: previously the Spitzer integrals (e-e collision operator) were updated inside the routines with the operators for the electron-density growth. Now the update is done outside these routines, and convergence on both operators is completely isolated.
  - BUG FIXED: previously, different EEDFs were being used to evaluate different parts of the e-e collision operator. Now, the convergence of the algorithm has been improved.
  - The global cycle for the non-linear operators of the Boltzmann equation is now testing the changes in the EEDF only as convergence criterion.
  - Slight improvement in the warning messages, in the cases where the non-linear operators of the Boltzmann equation do not converge.
  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.6.0` (2018/01/13):
  - NEW FEATURE: The user can now select non-conservative ionisation operators ("one takes all", "equal sharing" or "using single differential cross section") for the Boltzmann equation as well as two types of growth models ("spatial" or "temporal") for the electron number density. For more information, see sections 1 and 4.2.
  - NEW FEATURE: The user can now include the electron-electron collision operator in the Boltzmann calculations. For more information, see sections 1 and 4.2.
  - IMPORTANT CHANGE: Changes in the way the power balance is evaluated. The evaluation of the "reference power" is now fairer. The power transferred to the elastic and the CAR channels is now separated into gains and losses, and the reference power is obtained as the channel from which the electrons gain more energy.
  - BUG FIXED: fixed bug in the State.find static method, that caused errors when finding all the electronic states of a certain gas.
  - BUG FIXED: fixed bug introduced in commit 813e38f (Nov 9, 2017), that prevented the user to deactivate the output of the code.
  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.5.0` (2017/12/14):
  - NEW FEATURE: The electron drift velocity is now evaluated by the code. This information can be seen in the "Swarm parameters" and "Look-up-table" tabs of the GUI, as well as in the corresponding output files. (User Requested)
  - NEW FEATURE: "Extra" electron-scattering cross sections, *i.e.* cross sections that are not taken into account when solving the electron Boltzmann equation, can now be added. The corresponding rate coefficients are obtained by integrating these cross sections over the calculated EEDF. For more information, see section 4.2.

  - NEW FEATURE: The power balance is now detailed in a per-gas basis. (User Requested)
  - BUG FIXED: The output of the electron "Ionization" and "Attachment" was not being done properly.
  - IMPORTANT CHANGE: the only comment character allowed in the input files is now `%`, for performance reasons. The `#` character is NO LONGER RECOGNISED.
  - IMPORTANT CHANGE: A few changes in the way the properties of the setup file are organised. For more information, see section 3.1.3.
  - A few changes where introduced to improve the performance of the code:
    * Improved performance of the Collision.find static method, by avoiding implicit values in the loop.
    * Improved performance of the output.saveEedf method, by vectorizing a loop (almost 13x speedup)
  - A lot of LoKI-C code has been introduced and several changes in the structure of the code has been made in order to accommodate and integrate the chemistry module. However, LoKI-C is still not available (Coming soon!)
  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.4.0` (2017/10/17):
  - NEW FEATURE: The user can now prescribe a Maxwellian EEDF. For more information, see section 4.2.
  - NEW FEATURE: The user can now specify the "refresh frequency" of the GUI, in order to speedup simulations. For more information, see section 4.3.
  - Structural changes in the calculations and aesthetical changes in the GUI, in order to speedup calculations. A huge difference is observed, when compared with previous versions.
  - BUG FIXED: the electron temperature was being calculated with the characteristic energy instead of the mean energy.
  - BUG FIXED: solved bug causing the GUI to display the same rate coefficient for direct and inverse processes.
  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.3.0` (2017/09/08):
  - NEW FEATURE: The user can now specify the populations used for the evaluation of the elastic momentum-transfer cross section, from the effective. This feature must be used with care. For more information, see section 4.2.
  - Added graphs in the GUI, for the reduced first Townsend and attachment coefficients vs. E/N.
  - Included the character "-" in the names of electronic states (important for some N2 states, and not supported before).
  - BUG FIXED: fixed error in the evaluation of the elastic momentum-transfer cross section from the effective. The bug affected molecular gases only.
  - BUG FIXED: fixed a completely harmless error related with the parsing of collisions. Electrons where not properly accounted for.
  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.2.0` (2017/06/09):
  - NEW FEATURE: added support for the continuous approximation for rotations (CAR). For more information, see sections 1 and 4.2.
  - BUG FIXED: the evaluation of the elastic momentum-transfer cross section from the effective is now properly done. Previously, there were some errors in the evaluation of the populations of rotational states, as well as with the inclusion of superelastic mechanisms. The problem was not affecting the simulations in which an elastic cross section was provided by the user.
  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.1.1` (2017/04/10):
  - BUG FIXED: solved error in the calculation of the reduced first Townsend coefficient. The reduced field was being used in Townsend instead of SI units, in the evaluation of the coefficient.

  - A few minor changes to improve stability and performance of the code.

- `LoKI-B_v0.1.0` (2017/04/05):
  - First release of the code (beta).

References

[1] Tejero A, Guerra V, Gonçalves D, Lino da Silva M, Marques L, Pinhão N, Pintassilgo C D and Alves L L 2019 *Plasma Sources Sci. Technol.* **28** 043001

[2] Tejero A, Guerra V, Pinhão N, Pintassilgo C D and Alves L L 2021 *Plasma Sources Sci. Technol.* **30** 065008

[3] https://www.ipfn.tecnico.ulisboa.pt/nprime/ (last access September 29, 2025)

[4] Guerra V and Loureiro J 1997 *Plasma Sources Sci. Technol.* **6** 373-385

[5] Alves L L 2007 *Plasma Sources Sci. Technol.* **16** 557-569

[6] https://www.tue.nl/en/research/research-groups/elementary-processes-in-gas-discharges (last access September 29, 2025)

[7] https://www.mathworks.com/products/matlab.html (last access September 29, 2025)

[8] https://isocpp.org/ (last access September 29, 2025)

[9] https://eigen.tuxfamily.org/ (last access September 29, 2025)

[10] Lorentz H A 1905 The motion of electrons in metallic bodies I *Proc. Royal Netherlands Academy of Arts and Sciences (KNAW)* vol 7 (Amsterdam: KNAW) pp 438–53

[11] Holstein T 1946 *Phys. Rev.* **70** 367–84

[12] Allis W P 1956 Motions of ions and electrons *Handbuck der Physik ed S Flügge* vol 21 (Berlin: Springer) pp 383–444

[13] Delcroix J-L 1963/1966 *Physique des Plasmas* (*Monographies Dunod* vol 1 et 2) (Paris: Dunod) (in French)

[14] Alves L L, Bogaerts A, Guerra V and Turner M M 2018 *Plasma Sources Sci. Technol.* **27** 023002

[15] Tagashira H, Sakai Y and Sakamoto S 1977 *J. Phys. D: Appl. Phys.* **10** 1051-1064

[16] Thomas W R L 1969 *J. Phys. B: At. Mol. Phys.* **2** 551-561

[17] Ridenti M A, Alves L L, Guerra V and Amorim J 2015 *Plasma Sources Sci. Technol.* **24** 035002

[18] Chapman S and Cowling T G 1939 *The Mathematical Theory of Non-uniform Gases* (Cambridge: Cambridge University Press)

[19] Shkarofsky I, Johnston T and Bachynski M 1966 *The Particle Kinetics of Plasmas* (Addison-Wesley Publishing Company)

[20] Dyatko N A, Kochetov I V and Napartovich A P 1993 *J. Phys. D: Appl. Phys.* **26** 418-423

[21] Ferreira C M and Loureiro J 2000 *Plasma Sources Sci. Technol.* **9** 528-540

[22] Klein O and Rosseland S 1921 *Z. Phys.* **4** 46-51

[23] Amemiya H 1997 *J. Phys. Soc. Jpn.* **66** 1335-1338

[24] Nam S K and Verboncoeur J P 2009 *Computer Physics Communications* **180** 628–635

[25] Toneli D A, Pessoa R S, Roberto M and Gudmundsson J T 2015 *J. Phys. D: Appl. Phys.* **48** 495203

[26] https://lxcat.net/ (last access September 29, 2025)

[27] Alves L L 2014 *J. Phys. Conf. Series* **565** 012007

[28] Petrović Z Lj, Dujko S, Marić D, Malović G, Nikitović Ž, Šašić O, Jovanović J, Stojanović V and Radmilović-Radenović M 2009 *J. Phys. D: Appl. Phys.* **42** 194002

[29] Alves L L, Marques L, Pintassilgo C D, Wattieaux G, Es-sebbar Et, Berndt J, Kovačević E, Carrasco N, Boufendi L and Cernogora G 2012 *Plasma Sources Sci. Technol.* **21** 045008

[30] Turner M M 2015 *Plasma Sources Sci. Technol.* **24** 035027

[31] Salabas A, Gousset G and Alves L L 2002 *Plasma Sources Sci. Technol.* **11** 448-465

[32] Blevin H A and Fletcher J 1984 *Aust. J. Phys.* **37** 593-600

[33] https://demo.lxcat.net/ (last access September 29, 2025)

[34] https://en.wikipedia.org/wiki/Off-side_rule

[35] Opal C, Peterson W and Beaty E 1971 *J. Chem. Phys.* **55** 4100–4106

[36] Kim Y-K 1995 *Atomic and Molecular Processes in Fusion Edge Plasmas* (Springer Science, Business Media, New York)

[37] Vialetto L, Ben Moussa A, van Dijk J, Longo S, Diomede P, Guerra V and Alves L L 2021 *Plasma Sources Sci. Technol.* **30** 075001

[38] www.mathworks.com/help/matlab/ref/odeset.html (last access September 29, 2025)

[39] https://www.hdfgroup.org/solutions/hdf5/

[40] https://myhdf5.hdfgroup.org/
http://www.h5py.org/, https://docs.h5py.org

[41] Pitchford L C, Alves L L, Bartschat K, Biagi S F, Bordage M C, Phelps A V, Ferreira C M, Hagelaar G J M, Morgan W L, Pancheshnyi S, Puech V, Stauffer A and Zatsarinny O 2013 *J. Phys. D: Appl. Phys.* **46** 334001

[42] Treanor C E, Rich J W and Rehm R G 1968 *J. Chem. Phys.* **48** 1798

[43] Gordiets B, Mamedov S and Shelepin L 1975 *Sov. Phys. - JETP* **42** 237

[44] Rockwood S D 1973 *Phys. Rev.* A **8** 2348-2358

[45] https://www.mathworks.com/help/matlab/ref/ode15s.html (last access September 29, 2025)