The LisbOn KInetics Monte Carlo solver - user manual

Current version: `LoKI-MC_v1.0.0`

Tiago C Dias, Antonio Tejero-del-Caz, Luís Lemos Alves and Vasco Guerra

email: loki@tecnico.ulisboa.pt

September 28, 2022

The following is the user manual of the LisbOn KInetics Monte Carlo (LoKI-MC) simulation tool [1]. LoKI-MC solves the electron kinetics for non-magnetized low-temperature plasmas excited by homogeneous DC electric fields from different gas mixtures, using Monte Carlo techniques. Following the strategy and data organization of the LisbOn KInetics Boltzmann (LoKI-B) solver [2], the program easily addresses the simulation of the electron kinetics in any complex gas mixture of atomic / molecular species, describing electron collisions with any target state (electronic, vibrational and rotational), characterized by any user-prescribed population. LoKI-MC is written in C++, benefiting from a highly-efficient object-oriented structure. On input, the code requires the working conditions, the gas-mixture composition, the distributions of populations for the atomic/molecular gases considered, and the relevant sets of electron-scattering cross sections obtained from the open-access website LXCat (`http://www.lxcat.net/`). On output, it yields the electron energy distribution function, the electron velocity distribution function, the electron swarm parameters, the collision rate-coefficients, and the electron power absorbed from the electric field and transferred to the different collisional channels. Besides, LoKI-MC offers the possibility of considering the effects of the **gas thermal motion** and the **anisotropic ionization scattering** induced by momentum conservation. Future versions will address time-dependent electric fields and DC magnetic fields.

LoKI-MC was developed by the Portuguese group N-Plasmas Reactive: Modeling and Engineering (N-PRiME) [3], resorting to well-grounded scientific foundations established years ago [4–11]. The code is open-source, licensed under the *GNU general public license*. The simulation tool is freely available at `https://github.com/IST-Lisbon/LoKI-MC`, for users to perform electron kinetics calculations, and for modellers 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 funded by Portuguese FCT - Fundação para a Ciência e a Tecnologia, under Projects UIDB/50010/2020 and UIDP/50010/2020 and Grant PD/BD/150414/2019 (PD-F APPLAuSE).

Contents

1 The LoKI-MC code **3**

2 Quick-start guide **5**

2.1 First LoKI-MC simulation 5

2.2 How to contact us 7

2.3 How to reference the code 7

3 The setup file **8**

3.1 Working conditions 9

3.2 Electron kinetics 9

3.3 Graphical user interface 14

3.4 Output 15

4 Auxiliary input files **18**

4.1 LXCat files 18

4.2 Property files 20

4.3 Species description 21

5 Property functions **23**

A Installation guide **26**

A.1 User 26

A.2 Developer 26

References **29**

1 The LoKI-MC code

LoKI-MC is a Monte Carlo simulation tool that solves the electron kinetics for non-magnetized low-temperature plasmas excited by homogeneous DC electric fields in *any* complex gas mixture. In this section, we summarize briefly the code scope, presenting the input data necessary to run it and the physical information given in the output. The physical grounds of LoKI-MC are not discussed here, since they are detailed in the paper presenting the code [1]. Therefore, in the following, we assume that the user is already familiarized with the concepts presented in the paper.

LoKI-MC solves the electron kinetics given:

- the working conditions (reduced electric field E/N, gas pressure p and gas temperature T_g);
- the gas-mixture composition;
- the distributions of populations of electronic, vibrational and rotational levels (if applicable);
- the sets of cross sections for the different levels of the gases in the system.

The electric field $\mathbf{E}$ is set to be antiparallel to the z axis ($\mathbf{E} = -E\hat{\mathbf{z}}$).

The state levels of of each gas are organized as a system of "Russian puppets." Table 1 exemplifies the system of states for a N_2-N-He gas mixture. Here, each gas is composed by several electronic / vibrational / rotational levels. Electronic levels are parents of vibrational levels and vibrational levels are parents of rotational levels (if they exist). The populations of the levels can be set directly in text input-files or via functions for typical distributions at user-defined temperatures (e.g. Boltzmann or Treanor).

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	**$N(^4S)$**		
		$N(^2D)$		
		$N(^2P)$		
		⋮		
	He	**$He(1^1S)$**		
		$He(2^3S)$		
		$He(2^1S)$		
		⋮		
	⋮			

LoKI-MC uses electron scattering cross sections from the LXCat open-access website [17]. The cross sections can have several types: elastic, excitation, ionization, attachment and *effective*. As explained in section 2.1 of [1], when an effective cross section is used for a given state, the elastic cross section is deduced by subtracting all the inelastic components involving that state. As the state structure of LoKI-MC is richer than the current organization of LXCat, the LXCat files that do not belong to the IST-Lisbon database require some minor modifications, as explained in section 4.1.

The user can choose among different ionization models, concerning the energy sharing between the primary and secondary electrons and the momentum conservation. Besides, the effect of the thermal motion of the background molecules can be considered, allowing to correctly study electron swarms at very low reduced electric fields.

The electron kinetics is solved by following the stochastic trajectory of a representative ensemble of N_e electrons under the influence of an external constant electric field, $\mathbf{E}$. The electrons perform a series of free flights interrupted by elastic, inelastic or superelastic collisions with gas molecules. The collision-free times and the collision dynamics are calculated by generating random numbers sampled from probability distributions based on the underlying physics. The energies of the electron swarm are initialized from a Maxwellian distribution at the gas temperature. The swarm energy increases until a steady-state is reached, after which the energies, positions and velocities of the electrons are sampled in order to calculate distribution functions, transport coefficients and other relevant quantities.

The simulation tool provides as output:

- the electron energy distribution function (EEDF), $f_0(u)$, and the anisotropies, $f_1(u)$ and $f_2(u)$;
- the electron velocity distribution function, $f(v_r, v_z)$;
- bulk and flux swarm parameters together with statistical errors;
- collision rate-coefficients;
- the electron-power distribution through the different collisional channels;
- the spatiotemporal evolution of the electron swarm (cf. figure 2);
- details of the Monte Carlo simulation (e.g. number of electron collisions before and after the steady-state).

If the graphical interface is activated, most of this information can be plotted in the end of the simulation, allowing the user to check immediately the quality of the results.

LoKI-MC 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} Vm^2).

Future versions of the code will allow to study time-dependent electric fields together with DC magnetic fields.

2 Quick-start guide

2.1 First LoKI-MC simulation

- You can obtain the last version of the code from the github URL
  https://github.com/IST-Lisbon/LoKI-MC
  by downloading the `LoKI-MC_v1.0.0` zip-file and extract it to your local repository.
- Install the necessary tools and compile the code, following the instructions given in Appendix A.
- Open the terminal.
- If you are using the Windows Subsystem for Linux (WSL), launch Xming.
- Go to the folder `LoKI-MC_v1.0.0/Code`.
- Run the first LoKI-MC simulation:
  Linux / Mac OS / WSL
  `./lokimc default_setup.in [number of threads]`
  Windows
  `.\lokimc.exe default_setup.in [number of threads]`
  The second argument is the location of the default setup file relatively to the folder `LoKI-MC_v1.0.0/Code/Input`. The third argument is the number of threads used in the parallel calculations. The number should be equal or smaller than the number of CPU cores in the computer. If no number is put, the code takes '1' as default value.

The default setup file concerns the electron kinetics in O_2 plasmas. The simulation can take some minutes, since the calculations are performed for five values of reduced electric field. The current status of LoKI-MC can be followed in the terminal, as exemplified in figure 1. When the calculation for a given E/N reaches to an end, the temporal evolution of the electron swarm and the distribution functions are plotted, as shown in figures 2 and 3. After the calculations are performed for all E/N, the electron power distribution and the swarm parameters are plotted as a function of E/N (figures 4 and 5). The final results are saved in text files at the folder `LoKI-MC_v1.0.0/Code/Output/swarm_O2`.

As you may find in the folder `LoKI-MC_v1.0.0/Input`, there are setup files prepared for several gases. For example, if you want to run CO_2 (N_2), you should run the LoKI-MC command using as second argument `CO2/CO2_swarm_setup.in` (`Nitrogen/N2_swarm_setup.in`).

Now that you have run the code for the first time, you should read carefully the next sections, regarding the structure of the input files and the organization of the code.

```
*********Simulation status*********

E/N: 1 Td

Number of real collisions: 4.29744e+07
Number of null collisions: 2.09016e+08

Current time: 1.40186e-05 s
Steady-state time: 7.87705e-07 s

Number of integration points: 24131

Mean energy [eV]: 0.198642
Relative error: 0.00231725

Flux drift velocity [m/s]: 7.71296 -29.976 12141.5
Relative error:    4.83066   0.926818 0.00653894

Flux diffusion coefficients [m^2 s^-1]: 73.4397 73.8336 32.8937
Relative error: 0.0174391 0.0156816 0.0213585

Power balance relative error: 0.000831014
```

Figure 1: Example of the status of a LoKI-MC simulation.

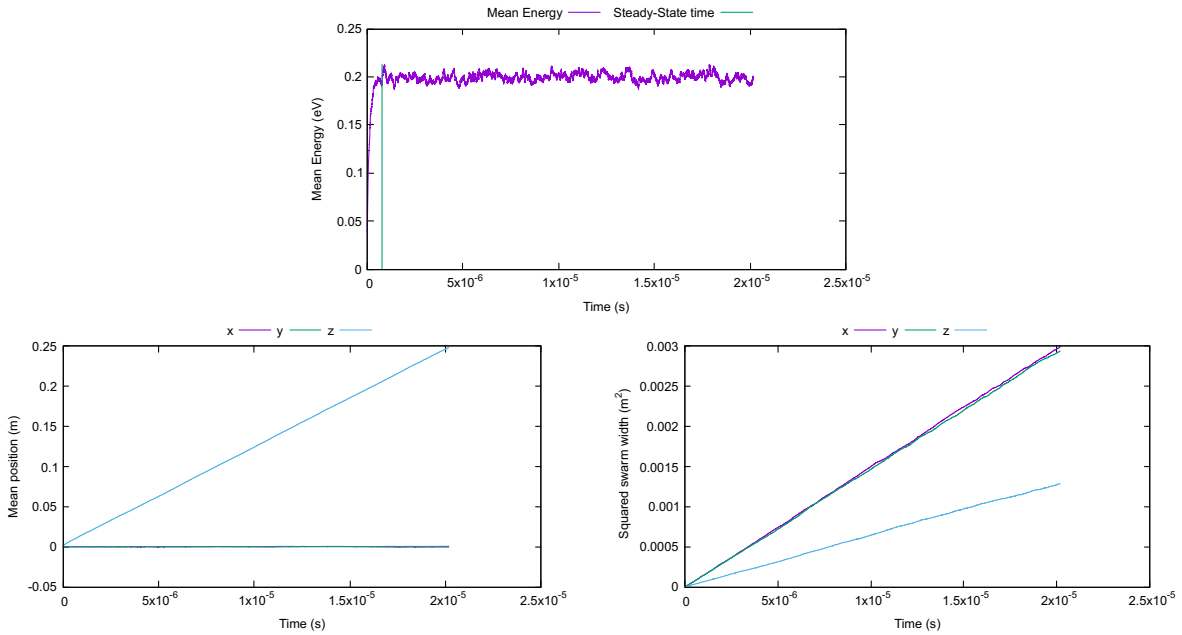


Figure 2: Example of the temporal evolution of the swarm properties in O_2 at $E/N = 1$ Td.

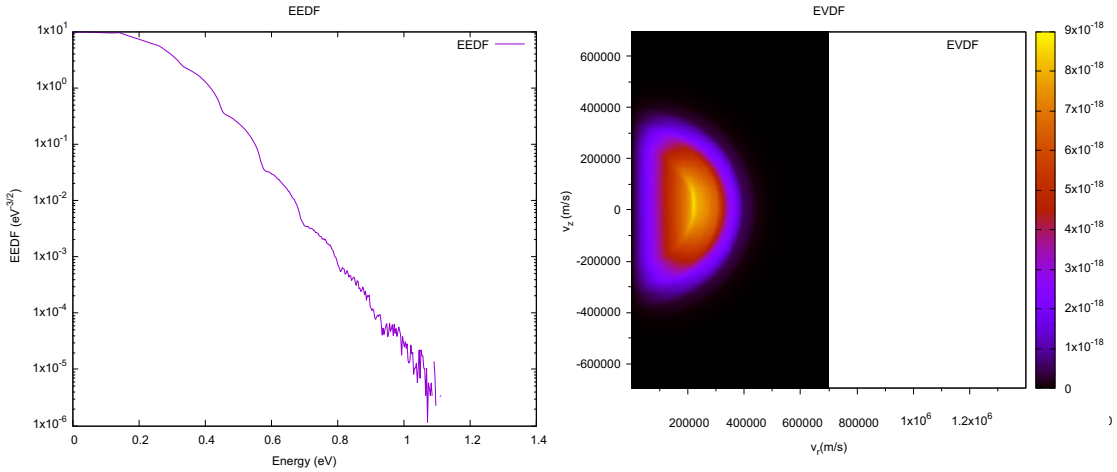


Figure 3: Electron energy distribution function (EEDF) and electron velocity distribution function (EVDF) in O_2 at $E/N = 1$ Td.

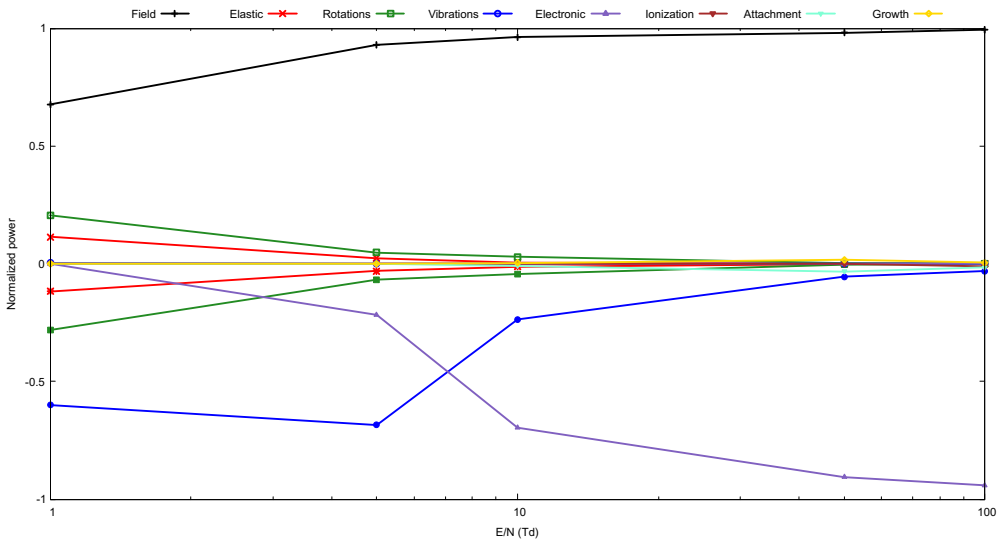


Figure 4: Distribution of electron power through the different collisional channels in O_2.

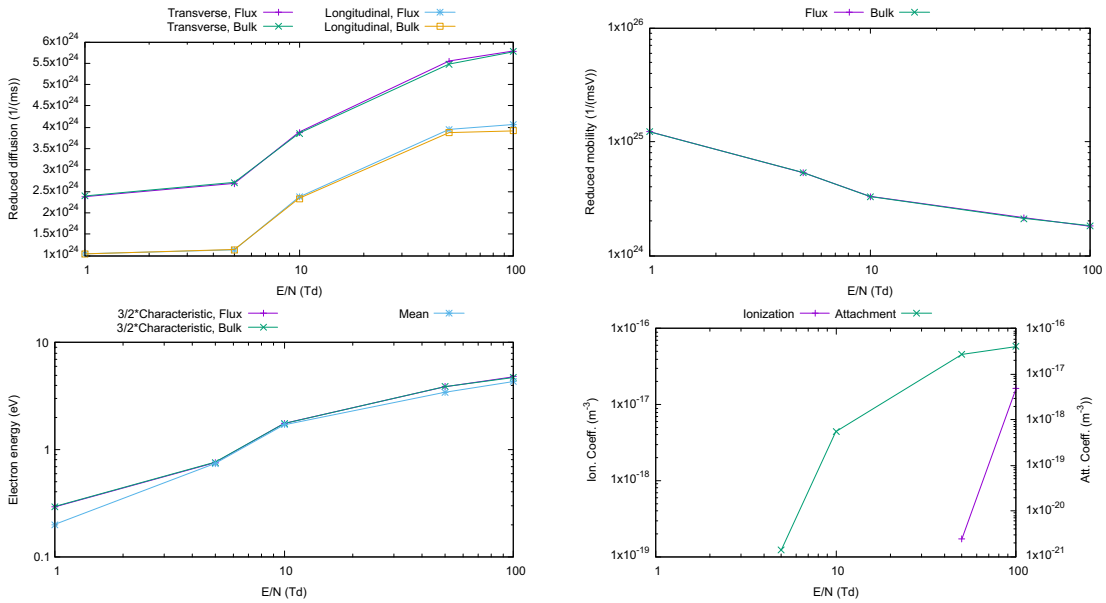


Figure 5: Electron swarm parameters in O_2.

2.2 How to contact us

After downloading LoKI-MC, 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-MC
just giving your **name** and **affiliation**.

2.3 How to reference the code

LoKI-MC is the result of the efforts of the Portuguese group N-PRiME, that decided to share the outcome of its research with the members of the Low-Temperature Plasmas community.

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

Dias T C *et al The LisbOn KInetics Monte Carlo solver* 2022 *Comput. Phys. Commun.* 108554, doi: https://doi.org/10.1016/j.cpc.2022.108554.

Additionally, do not forget to reference properly the LXCat databases used for the cross sections, which can be found in the beginning of each LXCat file.

3 The setup file

The setup file is the main input file of LoKI-MC, as well as the main User Interface (UI) with the code.

> The setup file is an **indentation structured file**, *i.e.* it follows the "Off-side rule" [14] 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.

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 ionization collisions
  - type of scattering in ionization collisions
- about numerical details
  - number of electrons in the ensemble
  - energy grid for LoKI-MC
  - numbers of cells used in the discretization of the distribution functions
  - inclusion of the gas-temperature effect
  - criteria for stopping the simulation

The setup file is organized according to the following syntax rules.

- The setup file can include comments, signaled by the character `%`.
- Some of the fields in the setup 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
      - Oxygen/O2_LXCat.txt
      - Oxygen/O2_rot_LXCat.txt
  ```

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 `LoKI-MC_v1.0.0/Code/Input`, eventually organized in subfolders with self-explanatory names (*e.g.* `Oxygen`, `Databases`). 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 and property functions, see section 4 and 5, respectively.

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

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

The next subsections present these fields in detail.

3.1 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_setup.in`' file that handles this configuration.

```
% --- configuration of the working conditions ---
workingConditions:
  reducedElecField:  [1,5,10,50,100]          % in Td
  gasPressure: 133.32                         % in Pa
  gasTemperature: 300                         % in K
```

Code 1: Configuration of the working conditions of the simulation

- The `reducedElecField` corresponds to E/N, in Td, and it can be composed by multiple values. Besides the format given in the setup file, LoKI-MC also accepts values through the functions `linspace` and `logspace`, using the same parameters as in MATLAB [18]. For example, `linspace(0.1,100,10)` or `logspace(-1,2,10)`.
- The `gasPressure` and the `gasTemperature` are composed by single values, in SI units.

3.2 Electron kinetics

In this field of the setup file the user can define all the details related to the electron kinetics. Code 2 shows the section of the '`default_setup.in`' file that handles this configuration.

```
% --- configuration of the electron kinetics ---
electronKinetics:
  isOn: true               % true or false (to activate of deactivate the electron Kinetics)
  eedfType: boltzmannMC    % only boltzmannMC, for now
  ionizationOperatorType: usingSDCS % oneTakesAll, equalSharing or usingSDCS
  ionizationScattering: isotropic   % isotropic or anisotropic
  LXCatFiles:            % cross section files
    - Oxygen/O2_LXCat.txt
    - Oxygen/O2_rot_LXCat.txt
  gasProperties:        % properties of the gases (S.I. Units)
    mass: Databases/masses.txt
    fraction:
      - O2 = 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:
      - O2(X,v=*) = harmonicOscillatorEnergy
      - O2(X,v=0,J=*) = rigidRotorEnergy
```

```
    statisticalWeight:
      - O2(X) = 3
      - O2(a1Dg) = 2
      - O2(b1Sg+) = 1
      - O2(X,v=*) = 3
      - O(3P) = 9
      - O(1D) = 5
      - O2(X,v=0,J=*) = rotationalDegeneracy
    population:
      - O2(X) = 1.0
      - O2(X,v=*) = boltzmannPopulation@gasTemperature
      - O2(X,v=0,J=*) = boltzmannPopulation@gasTemperature
  numericsMC:
    nElectrons: 1E4                          % number of electrons in the ensemble
    gasTemperatureEffect: smartActivation    % false, true or smartActivation
    nIntegrationPoints: 3E4                  % number of integration points after the steady-state is reached
%    nIntegratedSSTimes: 10                   % number of integrated steady-state times
%    maxCollisionsBeforeSteadyState: 5E4      % maximum number of collisions per electron before the steady-state
%    maxCollisionsAfterSteadyState: 5E4       % maximum number of collisions per electron after the steady-state
%    relError:                                % tolerances for the relative errors of the MC results
%      meanEnergy: 1E-3
%      fluxDriftVelocity: 1E-2
%      fluxDiffusionCoeffs: 1.5E-2
%      powerBalance: 1E-4
%    nInterpPoints: 1E4
%    nEnergyCells: 500
%    nCosAngleCells: 50
%    nAxialVelocityCells: 200
%    nRadialVelocityCells: 200
```

Code 2: Configuration of the electron kinetics of the simulation

- The `isOn` allows the user to activate (`true`) or deactivate (`false`) the simulations of LoKI-MC.
- The `eedfType` allows the user to specify the type of simulation for the EEDF, and it can only be `boltzmannMC`, for now.
- The `ionizationOperatorType` allows the user to choose the type of energy sharing in ionizaton collisions (see section 2.5 of [1] for more details). The possible configurations are:
  - `oneTakesAll`: the new-born secondary electron is introduced at zero kinetic-energy and the scattered electron takes the remaining energy after an ionization event.
  - `equalSharing`: the remaining energy after the event is equally shared between the scattered and the secondary electrons.
  - `usingSDCS`: the energy sharing is determined by 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* [15], 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 ionization cross sections with a very reliable shape [16], which can be normalized from available experimental data for the corresponding integral cross sections. 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

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

    where $\beta \simeq 2$ and w is a parameter close to the ionization threshold $V_{i,\text{ion}}$, which is set for each gas according to the original estimates [15].

    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 ionization potential.

- The `ionizationScattering` allows the user to specify the type of scattering in ionization events, `isotropic` or `anisotropic`. When `isotropic` is chosen, the scattering angles of both electrons are randomly calculated from uniform distributions. When `anisotropic` is chosen, the scattering angles are calculated so as to conserve the linear momentum of the system. Although the second option should be the most correct approach, note that most Boltzmann solvers use `isotropic` scattering in ionization events. Therefore, most cross-section sets are optimized with this assumption and the momentum-conservation scattering may lead to worst comparisons with experimental swarm parameters.
  See sections 2.5 and 4.4 of [1] for more details.

- 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 (`http://www.lxcat.net/`). 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-MC to retrieve the relevant data; see section 4.1 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 Monte Carlo solution of the electron kinetics**, 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 (`http://www.lxcat.net/`), provided they comply with certain format requirements for LoKI-MC to retrieve the relevant data; see section 4.1 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 4.2).
  The user should read carefully section 2.1 of [1] for more details.

  The prescription of populations by the user is strongly discouraged and should be taken with extreme care, since the values of these populations are directly used without any checking or processing. This is an advanced setup option, to be used by proficient users only. 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-MC adopts a Boltzmann distribution at 300 K, 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, the code evaluates an elastic momentum-transfer cross section and uses this **same** cross section for **all** the electronic target-states of the gas, with non-zero population;
  - when the collisional data for a gas includes an electron-impact "elastic" momentum-transfer cross section (*e.g.* for the ground-state of the gas), the user must provide **individual** elastic momentum-transfer cross sections for **each** electronic target-state of the gas, with non-zero population.

- 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 when an harmonic / anharmonic oscillator model is assumed.
- `rotationalConstant` is optional, and is used to evaluate the energy of the different rotational states when a rigid-rotor model is assumed.
- `electricQuadrupoleMoment` is not used for now.
- `OPBParameter` is optional and corresponds to the value of the w parameter to be used in (1), when adopting the `usingSDCS` mode for the ionization 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 should 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 4.2).

  ```
  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.

- 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 [12] 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 and to define prescribed distributions of populations (e.g, see (8) and (11) in section (5)), 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 (5)).

  - 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 values of the populations should add 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 5).

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

When using the "direct input" method or the "property functions" method, properties should 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 4.2).

```
Nitrogen/N2_vibpop.txt
```

- In `numericsMC` the user can specify the numerical details of the simulations.

  There are two **mandatory** parameters.

  – `nElectrons`: the number of electrons in the ensemble. Typically between 10^4 and 10^5.

  – `gasTemperatureEffect`: concerns whether or not the thermal motion of the background molecules is considered. The possible configurations are:

    * `true`: the thermal motion is always considered.
    * `false`: the thermal motion is not considered. This option is faster than `true`, but the low E/N values are not correctly described.
    * `smartActivation`: the thermal motion of the background molecule is considered only in collisions where the energy ϵ of the incident electron is comparable with the gas mean energy: $\epsilon \leq 20 \times \frac{3}{2} k_B T_g$, where k_B is the Boltzmann constant and T_g is the gas temperature. This option improves the computational efficiency relatively to `true`, while assuring a good physical description since, even when the electron and molecule energies are similar, the electron speed is much larger than the molecule speed, due to a much lower mass.

  Additionally, **at least one** stopping criterion must be defined. The simulation will stop when at least one of the following criteria is achieved (for more details, see section 2.7 of [1]).

  – `nIntegrationPoints`: number of integration points used to obtain the average swarm parameters and distribution functions, after the steady-state has been reached. We recommend values between $2{\cdot}10^4$ and 10^5. **This criterion should be used by the non-experienced users, since the others require a deeper knowledge of the method and the right values may vary significantly with the conditions and the target gas.**

  – `nIntegratedSSTimes`: integration time-interval, given as multiples of the steady-state time.

  – `maxCollisionsAfterSteadyState`: maximum number of collisions per electron, after the steady-state has been achieved.

  – `relError`: this field concerns the maximum tolerances for the relative statistical errors of the `meanEnergy`, the `fluxDriftVelocity` and the `fluxDiffusionCoeffs` and the `powerBalance`. For example,

```
relError:
  meanEnergy: 1E-3
  fluxDriftVelocity: 1E-2
  fluxDiffusionCoeffs: 1.5E-2
  powerBalance: 1E-4
```

    The user does not need to define the tolerances for all variables. The criterion `relError` is considered to be achieved when all the defined tolerances are fulfilled.

  In this field, it is also possible to give details on how the steady-state is considered. **By default**, the steady-state time t_{ss} is considered automatically when two conditions are fulfilled (for more details, see section 2.6 of [1]):

  – the swarm is constant apart from statistical fluctuations, which is verified if the temporal average of the swarm energy in the interval $[0.5t, 0.75t]$ is larger than the one in the interval $[0.75t, t]$;

– the relative standard deviation of the temporal average in the latter interval is smaller than 1%.

Moreover, for the proficient user, the steady-state can be forced by fixing the maximum number of collisions per electron before the steady state, using the parameter `maxCollisionsBeforeSteadyState`. However, **this is not recommended**, since in general the automatic steady-state detection works very well.

LoKI-MC interpolates the cross-sections in an energy grid with a certain number of points. This number is, **by default**, 10^4. However, the user can change it by setting `nInterpPoints` to a different value. Note that the maximum energy is dynamically defined during the simulation, depending on the energy of the electrons in the ensemble.

The numbers of cells used in the discretization of the distribution functions can also be changed in the setup file (see section 2.6 of [1]).

– `nEnergyCells`: number of energy cells used to discretize the EEDF and the anisotropies. Default: 500.

– `nCosAngleCells`: number of 'cosine-angle' cells used to discretize the electron angular distribution function, so as to obtain the anisotropies. Default: 50.

– `nRadialVelocityCells`: number of radial velocity cells used to discretize the EVDF. Default: 200.

– `nAxialVelocityCells`: number of axial velocity cells used to discretize the EVDF. Default: 200.

3.3 Graphical user interface

In this field of the setup file the user can set the configuration of the Graphical User Interface (GUI). Code 3 shows the section of the '`default_setup.in`' file that handles this configuration.

```
% --- configuration of the grafical user interface ---
gui:
  isOn: true
  fontSize: 11
  plotOptions:
    - MCTemporalInfo
    - distributionFunctions
    - powerBalance
    - swarmParameters
  terminalDisp:
    - setup
    - MCStatus
```

Code 3: Configuration of the GUI of the simulation

- The `isOn` parameter allows the user to activate (`true`) or deactivate (`false`) the GUI.
- The `fontSize` parameter allows the user to choose the size of the text font used in the plots (11, by default). If it is too large, the plots will be badly formatted.
- In `plotOptions`, the user can choose the plots to be shown:
  – `MCTemporalInfo`: temporal evolution of the swarm properties (energy, position and squared width) for each E/N (see figure 2);
  – `distributionFunctions`: EEDF and EVDF for each E/N (see figure 3);
  – `powerBalance`: distribution of electron power through the different collision channels as a function of E/N (see figure 4);
  – `swarmParameters`: bulk and flux swarm parameters as a function of E/N (see figure 5);
- In `terminalDisp`, the user can choose what is printed in the terminal:
  – `setup`: setup file;
  – `MCStatus`: current status of the Monte Carlo simulation (see figure 1).

3.4 Output

In this field of the setup file the user can define the details of the simulation output. Code 4 shows the section of the '`default_setup.in`' file that handles this configuration.

```
output:
  isOn: true
  folder: swarm_O2
  dataFiles:
    - eedf
    - evdf
    - swarmParameters
    - rateCoefficients
    - powerBalance
    - MCSimDetails
%    - MCTemporalInfo
    - lookUpTable
```

Code 4: Configuration of the output of the simulation

- The `isOn` parameter allows the user to activate (`true`) or deactivate (`false`) the output.
- In `folder` the user can specify the folder where the datafiles 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 `LoKI-MC_v1.0.0/Code/Output/`.
- When `isOn` is `true`, `dataFiles` is a **mandatory** parameter, where the user can specify the information to be saved after each simulation. The available options for `dataFiles` are:
  - `eedf`
    It writes, for each E/N simulation, the datafile `eedf.txt`, with four columns displaying the kinetic energy u, the EEDF f_0 and the anisotropies f_1 and f_2, respectively. For details on the calculations, check section 2.6.1 of [1].
  - `evdf`
    It writes, for each E/N simulation, the datafile `evdf.txt`, with three columns displaying the radial velocity v_r, the axial velocity v_z and the EVDF $f(v_r, v_z)$, respectively. For details on the calculations, check section 2.6.1 of [1].
  - `swarmParameters`
    It writes, for each E/N, the datafile `swarmParameters.txt`. Please check sections 2.6.1 and 2.6.3 of [1]. The following electron parameters are present in the file, for both bulk and flux components (if applicable):
    * Reduced diffusion matrix, ND_{ij}, where N is the gas density (since there is no magnetic field and the electric field is oriented along $-z$, the off-diagonal components should be zero apart from statistical error);
    * Reduced transverse diffusion coefficient, $D_\perp N = \frac{D_{xx}+D_{yy}}{2}N$;
    * Reduced longitudinal diffusion coefficient, $D_{||}N = D_{zz}N$;
    * Velocity vector, v_i (first two components should be null apart from statistical error);
    * Drift velocity, $v_{d,z}$;
    * Reduced mobility coefficient, $\mu N = v_{d,z}/E$;
    * Reduced Townsend coefficient, $\alpha_{\text{ion}}/N = k_{\text{ion}}/v_{d,z}$, where k_{ion} is the total ionization rate coefficient;
    * Reduced attachment coefficient, $\alpha_{\text{att}}/N = k_{\text{att}}/v_{d,z}$, where k_{att} is the total attachment rate coefficient;
    * Characteristic energy, $\varepsilon_{\text{char}} = D_\perp/\mu$;
    * Mean energy, $\langle\varepsilon\rangle$;
    * Electron temperature, $T_e = \frac{2}{3}\langle\varepsilon\rangle$.

    Finally, the file contains parameters that are deduced from the EEDF:

* total ionization coefficient, k_{ion};
* total attachment coefficient, k_{att};
* reduced energy diffusion coefficient,
  $D_{\varepsilon}^{eedf} N = \frac{1}{3}\sqrt{\frac{2e}{m_e}} \int_0^{\infty} \frac{\epsilon'^2}{\Omega(\epsilon')} f_0(\epsilon') d\epsilon'$,
  where $\Omega(\epsilon') = \sigma_{t,c}(\epsilon') + \sqrt{m_e/(2e\epsilon')} k_{eff}$, being $k_{eff} = k_{ion} - k_{att}$ the effective ionization coefficient and $\sigma_{t,c}(\epsilon')$ the total cross-section for momentum transfer;
* reduced energy mobility coefficient,
  $\mu_{\varepsilon}^{eedf} N = -\frac{1}{3}\sqrt{\frac{2e}{m_e}} \int_0^{\infty} \frac{\epsilon'^2}{\Omega(\epsilon')} \frac{df_0(\epsilon')}{d\epsilon'} d\epsilon'$;
* reduced diffusion coefficient,
  $D^{eedf} N = \frac{1}{3}\sqrt{\frac{2e}{m_e}} \int_0^{\infty} \frac{\epsilon'}{\Omega(\epsilon')} f_0(\epsilon') d\epsilon'$;
* reduced mobility,
  $\mu^{eedf} N = -\frac{1}{3}\sqrt{\frac{2e}{m_e}} \int_0^{\infty} \frac{\epsilon'}{\Omega(\epsilon')} \frac{df_0(\epsilon')}{d\epsilon'} d\epsilon'$;
* characteristic energy, $\epsilon_{char}^{eedf} = D^{eedf}/\mu^{eedf}$.

Pay attention that the last five parameters are calculated under the **two-term** assumption [20,21]. Hence, they should be interpreted with care.

– `rateCoefficients`
It writes, for each E/N, the datafile `rateCoefficients.txt`, displaying 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 (including any additional extra cross sections), with the appropriate integration over the eedf. Besides, it writes the datafile `rateCoefficientsMC.txt`, containing the coefficients directly obtained from the Monte Carlo simulation (see section 2.6.2 of [1]).

– `powerBalance`
It writes, for each E/N, the datafile `powerBalance.txt`, displaying the values of the following electron power-density gains/losses (per electron at unit gas density; check section 2.6.4 of [1]):

* P_E/N
* $P_{\text{el}}^{\text{gain}}/N$, $P_{\text{el}}^{\text{loss}}/N$, $P_{\text{el}}^{\text{net}}/N \equiv P_{\text{el}}^{\text{gain}}/N + P_{\text{el}}^{\text{loss}}/N$
* $P_{\text{sup}}^{\text{gain}}/N$, $P_{\text{inel}}^{\text{loss}}/N$, $P_{\text{ele,vib,rot}}^{\text{net}}/N \equiv P_{\text{sup}}^{\text{gain}}/N + P_{\text{inel}}^{\text{loss}}/N$ (for electronic, vibrational, rotational mechanisms)
* P_{ion}/N
* P_{att}/N
* P_{growth}/N.

It also writes

* the power balance $P_{\text{growth}}/N + P_E/N + P_{\text{coll}}^{\text{gain}}/N + P_{\text{coll}}^{\text{loss}}/N$ (with "coll" referring to all collision types);
* the relative power balance (in %) $\left[P_{\text{growth}}/N + P_E/N + P_{\text{coll}}^{\text{gain}}/N + P_{\text{coll}}^{\text{loss}}/N\right] / \left[P_{\text{ref}}/N\right] \times 100$;

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.

– `MCSimDetails`
It writes, for each E/N, a file with a summary of the Monte Carlo simulation, containing:

* number of electrons;
* final simulation time;
* steady-state time;
* number of integration points;
* number of collisions before and after the steady-state;
* relative errors of the mean energy, flux drift velocity, flux diffusion coefficients and power balance;
* elapsed time during the calculation.

– `MCTemporalInfo`
It writes, for each E/N, a file with the spatiotemporal evolution of the electron swarm. It contains, the time, energy, position, squared width and velocity of the electron swarm.

– `lookUpTable`
  It writes the following datafiles:
  * `lookUpTableSwarm.txt`, containing a table with the data in the files `swarmParameters.txt`, as a function of E/N;
  * `lookUpTableRateCoeff.txt`, containing a table with the data in the files `rateCoefficients.txt`, as a function of E/N;
  * `lookUpTablePower.txt`, containing a table with the data in the files `powerBalance.txt`, as a function of E/N;
  * `lookUpTableEGrid.txt`, containing a table with all energy grids used to discretize the EEDFs. Each line contains the grid of one E/N, in the same order as the order files;
  * `lookUpTableEedf.txt`, containing a table with all EEDFs, f_0. Each line contains one EEDF, in the same order as the order files;

4 Auxiliary input files

This section gives detailed information on the configuration of the different auxiliary input files used by LoKI-MC, 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 `LoKI-MC_v1.0.0/Code/Input/` or in any subfolder within it. They are plain text files, although some can display extensions different than `*.txt`, for the sake of folder organization.

4.1 LXCat files

By default, LoKI-MC uses electron-scattering cross sections obtained from the LXCat open-access website [17], in the solution to the electron kinetics 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-MC**.

LXCat is organized to provide "data required for modeling low temperature plasmas" [17] 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 [19], yielding electron swarm parameters in agreement with available measured data (within experimental uncertainties), when used in a two-term Boltzmann solver [13, 19]. 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 ionization 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 organization 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.

This problem is to be solved by the developers of the platform, but meanwhile users can modify any cross section datafile downloaded from LXCat, to make it compliant with the LoKI-MC parser. For this, it is important to understand how LXCat structures the metadata associated with each cross section. In an LXCat file, each cross-section-table comes with a header including the following fields:

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

In principle, LoKI-MC 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-MC 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>]`
`...`

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`.
- `target state:` can detail the electronic/vibrational/rotational state of the target species.
- `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.
- `collision type:` 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.

See section 4.3 for more information about the description of the species involved in electron-scattering mechanisms.

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

- the LXCat files distributed with the code, and located in `LoKI-MC_v1.0.0/Code/ Input/`.

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

- cross section files, **directly downloadable only from the IST-Lisbon database with the website LXCat, and to date only for the gases indicated below**:
  - 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> 0); `N2-rot` for electron excitations from rotational states N_2(X,v=0,J);
  - 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);
  - 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); `O2-rot` for electron excitations from rotational states O_2(X,v=0,J); `O2-elec` for electron excitations from excited electronic states of O_2, namely $O_2(a^1\Delta_g, b^1\Sigma_g^+)$;
  - 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^-;
  - Ozone: `O3`, for electron excitations from the ground-state O_3(X);
  - 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).

Code 5 shows an example of a modified header, for a cross-section-table extracted from the `N2_LXCat.txt` file distributed with LoKI-MC.

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

Code 5: Extract from the `N2_LXCat.txt` file distributed with LoKI-MC (cf the modified `COMMENT` field)

4.2 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-MC, but in general these files can be prepared by the user, considering the following rules:

- for organization sake, all "property files" specified in the setup file should be located inside the input folder (`LoKI-MC_v1.0.0/Code/Input`), eventually organised in subfolders with self-explanatory names (*e.g.* `Nitrogen`, `Databases`);
- 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 4.3 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 6 as an example);
- property files can include the properties of gases not used in the simulations. **In the case of states, an error will be thrown if the user is trying to set a property of a non-existent state.**

Codes 6 and 7 show two examples of property files distributed with LoKI-MC.

```
% 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 6: 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
```

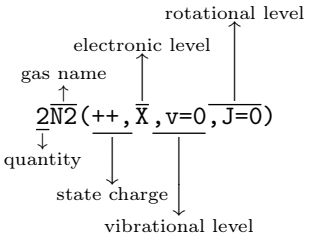
Code 7: Property file with a vibrational distribution of N_2 (`Nitrogen/N2_vibpop.txt`)

4.3 Species description

This section describes the ontology, including syntax rules, for the representation of the different species used in the simulations of LoKI-MC.

LoKI-MC 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.

5 Property functions

Property functions are functions at `LoKI-MC_v1.0.0/Code/PropertyFunctions/GasPropertyFunctions.h` and `LoKI-MC_v1.0.0/Code/PropertyFunctions/StatePropertyFunctions.h` that can be used to specify the properties of gases and states used in the simulations, respectively. Some property functions are distributed with LoKI-MC, 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, and add it also in the function `evaluateGasPropertyFunction` (for gases) or `evaluateStatePropertyFunction` (for states) located in the same respective file. **Note that the code must be recompiled, in order to use the new property functions (see appendix A).**

Property functions are called from the setup file (see section 3) using the following structure:

$$\texttt{<function name>}\underbrace{\texttt{@[list of parameters]}}_{\text{Optional}}$$

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 2 for examples, and therein the implementation of property function `boltzmannPopulation @gasTemperature`);

Property functions **SHOULD** be set only for those states used in the simulations, *i.e.* those listed in the specified LXCat files.

Table 2 lists the property functions distributed with `LoKI-MC_v1.0.0`, along with the required parameters and dependencies.

Table 2: List of property functions distributed with `LoKI-MC_v1.0.0`

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
	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

The property functions in table 2 correspond to the following physical models.

`harmonicOscillatorEnergy`

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

model

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

where $\omega_{\rm vib}$ 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

$$E_j = \frac{\hbar\omega_{vib}}{e}\left(j+\frac{1}{2}\right) - \frac{\hbar\omega_{\rm vib}\,\chi_e}{e}\left(j+\frac{1}{2}\right)^2 \quad , \tag{3}$$

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

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

$$E_J = BJ(J+1) \quad , \tag{4}$$

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 [22]

$$g_J = 2J+1 \quad . \tag{5}$$

`rotationalDegeneracy_N2`
It calculates the statistical weight of the *intrinsic* partition function of a rotational state J of the N_2 molecule [22]

$$g_J = 3\left[1+\frac{1}{2}\left(1+(-1)^J\right)\right](2J+1) \quad . \tag{6}$$

`rotationalDegeneracy_H2`
It calculates the statistical weight of the *intrinsic* partition function of a rotational state J of the H_2 molecule [22]

$$g_J = \left[2-(-1)^J\right](2J+1) \quad . \tag{7}$$

These property functions should be called within the `statisticalWeight` property of the setup file, for the corresponding rotational states (see also section 3.2).
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

$$\xi_j = \frac{g_j\,\exp\left(-\frac{E_j}{k_B T}\right)}{\sum_{i\in\text{siblings}} g_i\,\exp\left(-\frac{E_i}{k_B T}\right)} \quad , \tag{8}$$

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}

$$\xi_J = \frac{g_J \exp\left(-\frac{E_J}{k_B T}\right)}{\sum_{J'=0}^{J_{\mathrm{MAX}}} g_{J'} \exp\left(-\frac{E_{J'}}{k_B T}\right)} \quad , \tag{9}$$

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}

$$\xi_v = \frac{g_v \exp\left(-\frac{E_v}{k_B T}\right)}{\sum_{v'=0}^{v_{\mathrm{MAX}}} g_{v'} \exp\left(-\frac{E_{v'}}{k_B T}\right)} \quad , \tag{10}$$

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 [23, 24]

$$\xi_j = \frac{g_j \exp\left(\frac{j(E_1-E_0)-(E_j-E_0)}{k_B T_0}\right) \exp\left(-\frac{j(E_1-E_0)}{k_B T_1}\right)}{\sum_{i\in\mathrm{siblings}} g_i \exp\left(\frac{i(E_1-E_0)-(E_i-E_0)}{k_B T_0}\right) \exp\left(-\frac{i(E_1-E_0)}{k_B T_1}\right)} \quad , \tag{11}$$

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 [23, 24]

$$\xi_j = \begin{cases} \frac{g_j \exp\left(\frac{j(E_1-E_0)-(E_j-E_0)}{k_B T_0}\right) \exp\left(-\frac{j(E_1-E_0)}{k_B T_1}\right)}{\sum_{i\in\mathrm{siblings}} \xi_i} & , j \le j^\star \\ \frac{\xi_{j^\star} \frac{j^\star}{j}}{\sum_{i\in\mathrm{siblings}} \xi_i} & , j > j^\star \end{cases} \quad , \tag{12}$$

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

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

is the vibrational level corresponding to the minimum of the Treanor distribution (with $\omega_{\mathrm{vib}}\chi_e$ 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_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.

A Installation guide

This section describes the installation steps of LoKI-MC, depending on the purpose (user or developer) and on the operating system. Here, we call 'User' to someone that wants to utilize the code without changing any function and 'Developer' to someone that aims to change / improve the code built-in functions. In the case of Mac OS, only the 'Developer' option is available. At this point, you should have already downloaded the code and extracted it into your local repository.

A.1 User

Linux

- Open the command terminal.
- Type the following commands to install gnuplot:
  `sudo apt-get update`
  `sudo apt-get install gnuplot`
- Copy the executable `lokimc` from `LoKI-MC_v1.0.0/CompiledExecutables/linux_x64` to `LoKI-MC_v1.0.0/Code`.
- Inside the `Code` folder, change the permissions of the executable `lokimc` using the command:
  `chmod a+rwx lokimc`
- Done! Now you should come back to section 2.

Windows

- Install the latest version of Gnuplot using an executable available at https://sourceforge.net/projects/gnuplot/files/gnuplot/. For example, `5.4.2/gp542-win64-mingw.exe`. **During the installation you must select the gnuplot to be added to the PATH. This will allow you to run commands from anywhere on the command line. If you choose not do so, LoKI-MC will not work properly.**
- Assure that Gnuplot was successfully installed (and added to the PATH) by typing in the command line:
  `gnuplot -p -e 'plot sin(x)'`
  Verify that this command generates a window with the graph.
- Install the Windows Terminal, available at the Windows Store. This is important since the default terminal in Windows does not recognize some ANSI sequences which are used in the code.
- Copy the files `lokimc.exe`, `gsl.dll`, `gslcblas.dll`, `vcomp140.dll` and `vcomp140_1.dll` from `LoKI-MC_v1.0.0/CompiledExecutables/win_x64` to `LoKI-MC_v1.0.0/Code`. If you have a computer with a `x86` architecture, then you should choose the ones in the `win_x86` folder.
- Done! Now you should come back to section 2.

A.2 Developer

Linux

- Open the terminal and go to the folder `LoKI-MC_v1.0.0/InstallationCommands` using `cd`.
- Run: `sudo bash install_linux.sh all`
  Accept the steps whenever asked.
  We advise you to check previously the commands inside the bash script.
- Whenever you change the headers or the source files of the code, you should run the following command regarding the code compilation:
  `sudo bash install_linux.sh onlyCode`
- Done! Now you should come back to section 2.

Mac OS

- Open the Apple Store and install the `Xcode` program. The installation may take a long time.
- Open the terminal and go the folder `LoKI-MC_v1.0.0/InstallationCommands` using `cd`.
- Accept the Xcode license using the command:
  `sudo xcodebuild -license`
  Follow the instructions on the terminal to accept the license.
- To install the necessary tools to run the code, you can use either Homebrew or Macports.
  We advise you to check previously the commands inside the LoKI-MC bash scripts.

  Using Homebrew.

  Insert the following command on the terminal:
  `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"`
  Then, follow the two instructions in the end to add Homebrew to the path.

  Run the installation script of LoKI-MC:
  `bash install_macos.sh brewAll`

  Using Macports.

  Install the package manager Macports. The easiest way is to download the 'pkg' installer for your version of Mac OS.
  Run the installation script of LoKI-MC:
  `bash install_macos.sh portAll`
- Whenever you change the headers or the source files of the code, you should run the following command regarding the code compilation:
  `bash install_macos.sh brewOnlyCode`
  Or, if you have used Macports in the installation, the same command but with the option `portOnlyCode`.
- Done! Now you should come back to section 2.

Windows - using Windows Subsystem for Linux (WSL)
For C++ development in Windows 10, we recommend the usage of the Windows Subsystem for Linux (WSL), which enables running a Linux system under Windows 10. If you prefer to do it in Microsoft Visual Studio, we also explain it in the next section.

- Enable Windows Subsystem for Linux (WSL)
  - Open 'Settings'.
  - Click on 'Apps'.
  - Under the 'Related settings' section, click the 'Programs and Features' option.
  - Check the 'Windows Subsystem for Linux' option.
  - Click the 'OK' button.
  - Click the 'Restart' now button.
- Install Linux distribution using Microsoft Store
  - Open 'Microsoft Store'.
  - Install 'Ubuntu'.
  - Launch 'Ubuntu'.
  - Create a username for the Linux distribution and press 'Enter'.
  - Specify a password and press 'Enter'.
  - Repeat the password and press 'Enter' to confirm.
- Enable graphical Linux desktop applications

  - Install an X server to enable a graphical interface. We recommend Xming.
  - You can just accept the default settings.
  - Launch Ubuntu and insert the following command:
    `echo "export DISPLAY=localhost:0.0" >> ~/.bashrc`
  - Restart Ubuntu.

- After you complete the previous steps, you can start using the distribution as any other Linux system.
- Go to `LoKI-MC_v1.0.0/InstallationCommands`
  - Find the Windows directory of the LoKI-MC folder in the Windows explorer. To do that, go to the folder and click on the location near the top of the explorer, to get something similar to 'C:\Users\userName\Example folder\LoKI-MC_v1.0.0'.
  - Launch Ubuntu and Xming, and change the directory to the code folder using the 'cd' command. Using the previous example:
    `cd /mnt/c/Users/userName/Example\ folder/LoKI-MC_v1.0.0/InstallationCommands`
    Note 1: the characters '\' separating folders in Windows are replaced by '/' in Linux.
    Note 2: the Windows folder 'C:' is accessed using '/mnt/c'.
    Note 3: the spaces in the folder names must be preceded by the character '\'.
- Run: `sudo bash install_linux.sh all`
  Accept the steps whenever asked.
  We advise you to check previously the commands inside the bash script.
- Whenever you change the headers or the source files of the code, you should run the following command regarding the code compilation:
  `sudo bash install_linux.sh onlyCode`
- Done! Now you should come back to section 2.

Windows - using Microsoft Visual Studio

- Install the latest version of Gnuplot using an executable available at https://sourceforge.net/projects/gnuplot/files/gnuplot/. For example, `5.4.2/gp542-win64-mingw.exe`. **During the installation you must select the gnuplot to be added to the PATH. This will allow you to run commands from anywhere on the command line. If you choose not do so, LoKI-MC will not work properly.**
- Assure that Gnuplot was successfully installed (and added to the PATH) by typing in the command line:
  `gnuplot -p -e 'plot sin(x)'`
  Verify that this command generates a window with the graph.
- Install Visual Studio with the option 'Desktop development with C++' activated.
- Open Visual Studio. Select 'Continue without code'. Go to 'Tools → Command Line → Developer Command Prompt'.
- Go the folder `LoKI-MC_v1.0.0/InstallationCommands` using `cd`.
- Insert the following in the command prompt:
  `install_win_x64.bat all`
  We advise you to check previously the commands inside this batch script. If you have a computer with a `x86` architecture, you should change the command from `x64` to `x86`.
- The code is already installed. However, we advise you to install also the Windows Terminal, available at the Windows Store, since the default terminal in Windows does not recognize some ANSI sequences which are used in the code.
- Whenever you change the headers or the source files of the code, you should run the following command regarding the code compilation:
  `install_win_x64.bat onlyCode`
- Done! Now you should come back to section 2.

References

[1] Dias T C, Tejero-del-Caz A, Alves L L, Guerra V 2022 *Comput. Phys. Commun.* 108554, doi: https://doi.org/10.1016/j.cpc.2022.108554

[2] 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

[3] https://www.ipfn.tecnico.ulisboa.pt/nprime/ (last access September 28, 2022)

[4] Skullerud H R 1968 *J. Phys. D Appl. Phys* **1** 1567-1568

[5] Reid I 1979 *Aust. J. Phys.* **32** 231

[6] Boeuf J P and Marode E 1982 *J. Phys. D Appl. Phys* **15** 2169-2187

[7] Yousfi M, Hennad A and Alkaa A 1994 *Phys. Rev. E* **49** 3264-3273

[8] Vahedi V and Surendra M 1995 *Comput. Phys. Commun.* **87** 179-198

[9] Longo S 2000 *Plasma Sources Sci. Technol.* **9** 468-476

[10] Dujko S, White R D, Petrović and Robson R E 2010 *Phys. Rev. E* **81** 046403

[11] Rabie M and Frack C M 2016 *Comput. Phys. Commun.* **203** 268-277

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

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

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

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

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

[17] www.lxcat.net (last access September 28, 2022)

[18] https://www.mathworks.com/products/matlab.html (last access September 28, 2022)

[19] 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

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

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

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

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

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