The LisbOn KInetics Boltzmann solver
User 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 user manual of the LisbOn KInetics Boltzmann (LoKI-B) solver, 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® for LoKI-B, leveraging its matrix-based architecture;
- C++ for LoKI-B++, taking advantage of the higher performance of a compiled language, and linked with Eigen, 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.

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.

Both users and developers are encouraged to consult the *Reference Manual* for practical guidance on obtaining, navigating, and running the code.

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.

---

1 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.

1.1 Information about LoKI-B

1.1.1 How to obtain the code

You can obtain the last 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.

1.1.2 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 section 1.1.3).
      ii. Default configuration files in JSON format `'default_lokib_setup.json'` and `'default_lokib_pulse_setup.json'` (see section 1.1.3).
      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;
         - 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) 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..
   (d) A subfolder `OtherAuxFunctions`, with several `'*.m'` auxiliary functions that calculate some working conditions, help to process or parse data, etc..
   (e) A subfolder `Output` (eventually), where LoKI-B will write the output files resulting from the simulations.

1.1.3 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 1.1.2, 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 1.

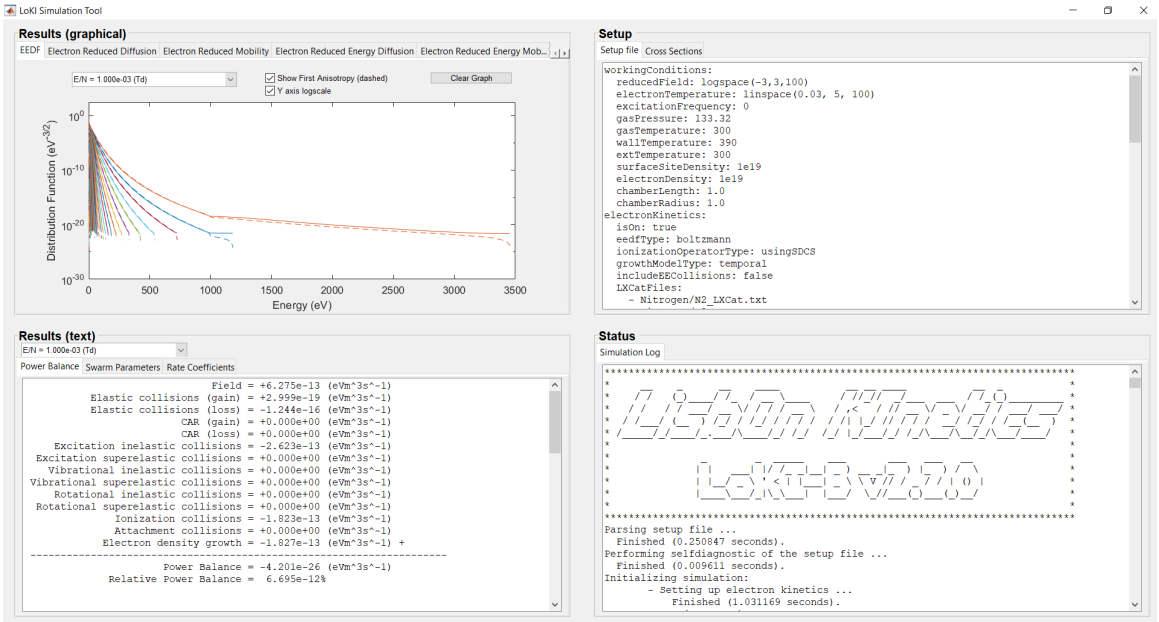

Figure 1: Graphical user interface output, for the default setup-file of LoKI-B (stationary simulations).

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 the *Reference Manual*.

1.2 Information about LoKI-B++

1.2.1 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.

1.2.2 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).

1.2.3 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.

1.3 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**.

1.4 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.