KINEBEC - Kinetic simulation of Quantum Boltzmann Equation

License

Authors

KINEBEC is managed by

- Alexandre MOUTON (CNRS Lille - France) as lead developer
- Thomas REY (Université de Lille - France) as lead scientific manager

Description of KINEBEC

KINEBEC is a simulation code developed in C dedicated to classical and quantum Boltzmann equations in 2D and 3D in velocity. This code is based on spectral methods in velocity and can uses MPI, OpenMP or CUDA tools for speeding up its execution and for managing large data. More details about the involved numerical methods can be found in the following papers:

- Mouhot & Pareschi: *Fast algorithms for computing the Boltzmann collision operator*, Math. Comput. **256**-75 (2006), pp. 1833-1852

- Filbet, Hu & Jin: *A numerical scheme for the quantum Boltzmann equation with stiff*, collision terms, Math. Model. Numer. Anal. **46** (2012), pp. 443-463

Requirements

We detail here the libraries and softwares that are required for installing and running KINEBEC on a Linux-type OS. Since the main purpose of KINEBEC is to perform parallelized simulations, we provided some configurations with MPI, OpenMP or CUDA. These configurations are not mandatory but strongly recommended for running 3D quantum simulations.
Note that you need at least one I/O dependency to be satisfied in order to compile and run KINEBEC (HDF5 or Adios).

Required in any case:

- Cmake (version >= 2.8)
- FFTW3 (version >= 3.3.4)
- Python 3.x

Required for MPI parallelization:

- OpenMPI (version >= 1.10.2)
- FFTW3-mpi (version >= 3.3.4) with headers

Required for OpenMP parallelization:

- OpenMP (version >= 3.1)
- FFTW3-omp (version >= 3.3.4) with headers

Required for CUDA parallelization:

- CUDA Toolkit (version >= 9.0)
- CuFFT with headers
- CUDA Runtime with headers

Required for HDF5 I/O:

- HDF5 (version >= 1.8.16) with headers
- Parallel HDF5 (version >= 1.8.16) with headers - recommended if you consider the MPI features

Required for Adios I/O:

- Mxml (version >= 2.9) with headers
- Zlib (version >= 1.2.8) with headers
- Bzip2 (version >= 1.0.6) with headers
- Glib 2.0 (version >= 2.48) with headers
- HDF5 (version >= 1.8.16) with headers
- Parallel HDF5 (version >= 1.8.16) with headers - recommended if you consider the MPI features

- Adios (version >= 1.9.0) with headers

If you are using a Debian-based Linux OS, you can install these dependencies with packages: for example, with Ubuntu, you can type the following command (replace `apt-get` by `apt` with Ubuntu 16.04 or newer):

```
sudo apt-get install cmake openmpi-* libfftw3-mpi-dev \
libfftw3-dev libfftw3-bin libmxml-dev zlib1g-dev libbz2-dev \
libglib2.0-dev libhdf5-openmpi-dev libhdf5-serial-dev libadios-dev
```

Installation

To install KINEBEC, you should proceed as follows:

1. Download and install all dependencies (see Section 4).
2. Move to the sub-directory `(KINEBEC_ROOT_PATH)/CMake_Dependencies` and verify if the paths are valid in each `Find***.cmake` file. If you have some difficulties with this part, do not hesitate to contact us.
3. Open a terminal and move to the directory `(KINEBEC_ROOT_PATH)`.
4. Type the following commands:

   ```
   python build_kinebec.py <IO flags> [build flags] [optional
   flags]
   ```

   You are required to provide at least one of the following IO flags: `--with-hdf5` (for using HDF5) and/or `--with-adios` (for using Adios). In addtion, you should provide one or several build flags which correspond to the versions of Kinebec you aim to compile: `--serial` (for serial version), `--mpi` (for OpenMPI version), `--omp` (for OpenMP version), `--cuda` (for CUDA version).

   Finally, you can also add the flag `--force-vectorization` for compiling specific loops within the code for better benefits of GCC vectorization. Note that this feature is highly memory consuming during the execution.
5. The compilation is successfully finished if the following sub-directories are created and filled:
   – `(KINEBEC_ROOT_PATH)/build/(BUILD_VERSION)/bin`
   – `(KINEBEC_ROOT_PATH)/build/(BUILD_VERSION)/lib`
   – `(KINEBEC_ROOT_PATH)/build/(BUILD_VERSION)/include`
   with `(BUILD_VERSION)` being set to `serial` and/or `mpi` and/or `omp` and/or `cuda`.

Getting started

To generate a setup file, you can type the following command:

```
(KINEBEC_ROOT_PATH)/build/serial/bin/generate_setup_file [setup_file]
```

To run a serial simulation, you can type the following command:

```
(KINEBEC_ROOT_PATH)/build/serial/bin/kinebec <setup_file>
```

To run a MPI simulation, you can type

```
mpirun -np <#CPUs> (KINEBEC_ROOT_PATH)/build/mpi/bin/kinebec <setup_file>
```

To run an OpenMP simulation, you can type

```
export OMP_NUM_THREADS=<#THREADS>
(KINEBEC_ROOT_PATH)/build/omp/bin/kinebec <setup_file>
```

To run a CUDA simulation, you can type the following command

```
(KINEBEC_ROOT_PATH)/build/cuda/bin/kinebec <setup_file> \
-nb <#blocks> -nt <#threads_per_block>
```

Note that a setup file is required for running the application `kinebec`. You can also add the flag `--resume` in order to resume a simulation that has been previously stopped (such option can be useful when you are using a job manager with a limited wall-time).

Visualization of results

Since the goal of KINEBEC is to perform parallelized simulation of Boltzmann-like equations, the I/O part of the code is parallelized with Adios or HDF5 libraries. If the Adios solution has been selected, some `.bp` files will be generated (one per time iteration). If the HDF5 solution is selected, some `.h5` files will be generated and each of them will be supplemented with a `.xdmf` reader file (one pair `.h5`/`.xdmf` per time iteration).

Such file can be read with the VisIt visualization software. More informations about VisIt can be found at the link below:

http://wci.llnl.gov/simulation/computer-codes/visit

For visualizing the contents of a `.bp`, you just need to start VisIt, open the `.bp` file and select a visualization method.

For visualizing the contents of a `.h5`/`.xdmf`, you just need to start VisIt, open the `.xdmf` file and select a visualization method.

Note that if an output result file set of the form {`diags_0.bp`, `diags_1.bp`, `diags_2.bp`, ...} is opened, it is possible to visualize the dynamics of the numerical results.

As a complement, it is also possible to visualize the contents of `.h5`/`.xdmf` files with Paraview:

https://www.paraview.org/

We also provide some Python-Matplotlib routines to visualize the contents of `.h5` files. Please check the file `visualization.py` to get inspired.

Do not hesitate to contact us for highlighting your numerical results on the KINEBEC web gallery by sending us pictures or movies.

Contact

- Alexandre MOUTON (lead developer): alexandre.mouton@univ-lille.fr
- Thomas REY (lead scientific manager): thomas.rey@univ-lille.fr