CoarseAIR

An Object-Oriented Code for Coarse-Grained Quasi-Classical Trajectories: Version 1.0, User's Manual

S. Venturi & B. Lopez

University of Illinois at Urbana-Champaign

Code documented by Daniel Steinberg

Table of Contents

1 Introduction **3**

1.1 User's Manual Organization 3

1.2 Notation 4

2 Tutorial **5**

2.1 Quickstart 5

2.1.1 Acquiring and Installing CoarseAIR (first time) 5

2.1.2 Installing CoarseAIR (later times) 6

2.2 Examples 7

2.2.1 An Ab Initio Study of the O_2+O Kinetics 7

2.2.2 Computing Dissociation Thermal Rate Coefficients for N_2+N_2 7

2.3 General Overview 9

2.3.1 CoarseAIR.sh, the CoarseAIR Launching Script 9

2.3.2 CoarseAIR Input Files 10

2.3.3 Tracking CoarseAIR Operations 12

3 Diatomic Potentials and Computation of Ro-Vibrational Levels **13**

3.1 Diatomic Potentials 13

3.1.1 Choosing Chemical System, Molecules, Atoms, and their Properties 13

3.1.2 Plotting Diatomic Potential 13

3.2 Computing Ro-vibrational Levels 13

3.2.1 Executing the Code 13

3.2.2 Format of the Output File 14

3.2.3 Visualization of the Level Properties 15

3.2.4 MATLAB Program for Ro-Vibrational Levels Calculation . . 15

3.3 References 15

4 Potential Energy Surfaces **16**

4.1 Potential Energy Surfaces (PES) 16

4.1.1 Input File 16

4.2 Computing the Potential Energy at Specific Atomic Configurations . . 17

4.2.1 Executing the Code 17

4.2.2 Plotting the Potential Energies 17

4.3 References 17

5 Preprocessing **18**

5.1 Splitting the Molecules' Level Lists 18

5.1.1 Input File 18

5.1.2 Executing the Code 19
5.1.3 Output File Formats 19

6 Simulating Trajectories **20**
6.1 Executing the Running Trajectories Program 20
6.2 Input File 20
6.3 Output File Formats 22

7 Post-processing the Trajectories: Cross Sections and Rate Coefficients **24**
7.1 Post-processing Trajectories 24
7.1.1 Executing the Code 24
7.1.2 Input File 24
7.2 Output File Formats 25
7.2.1 Cross Sections 25
7.2.2 Rate Coefficients 26
7.2.3 Processes Indexing 28

8 PyCoarseAIR: A Python Tools For Postprocessing **29**
8.1 Quickstart 29
8.1.1 Requirements 29
8.1.2 Executing PyCoarseAIR 30
8.1.3 PyCoarseAIR Output Files 30
8.2 A Brief List of PyCoarseAIR's Fundamental Input Variables 30
8.3 References 31

SECTION 1

Introduction

1.1 User's Manual Organization

This manual uses different colors for differentiating between commands, paths, inputs, variables, etc.:

- In RED, commands to be executed from the terminal.
- In GREEN, paths and file names.
- In BLUE, input strings for the input file.

1.2 Notation

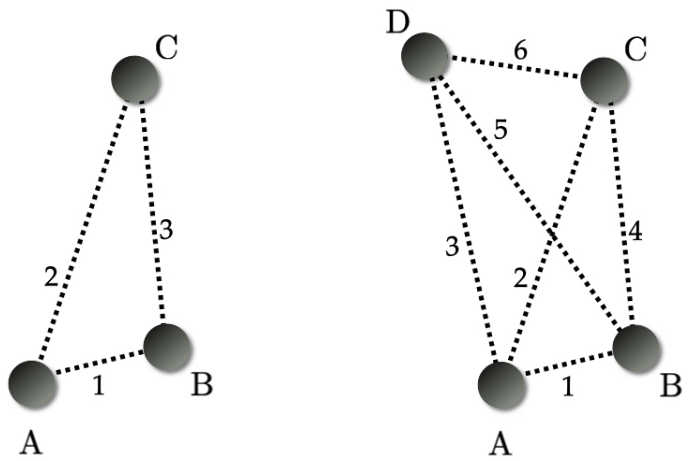

Figure 1: 3-atoms (a) and 4-atoms (b) chemical systems with the correspondent atomic pairs enumerated in the order followed by CoarseAIR.

- $\mathcal{N}_i^L$ indicates the number of levels contained by the molecule corresponding to the i-th pair;
- $\mathcal{N}_i^G$ indicates the number of levels/groups contained by the molecule corresponding to the i-th pair, if grouped. If not, $\mathcal{N}_i^G = \mathcal{N}_i^L$.

SECTION 2

Tutorial

2.1 Quickstart

This section entails everything necessary to install CoarseAIR and it briefly outlines the main steps for running the code. Throughout this manual, the instructions will be accompanied by examples for 3-atoms and 4-atoms systems.

2.1.1 Acquiring and Installing CoarseAIR (first time)

1. In your ∼/WORKSPACE, create a directory named CoarseAIR. For the remaining part of this manual, /CoarseAIR/ will indicate the ∼/WORKSPACE/CoarseAIR/ folder.

2. Once in /CoarseAIR, clone the CoarseAIR code from `https://github.com/simoneventuri/CoarseAIR.git` (i.e., type:
   git clone https://github.com/simoneventuri/CoarseAIR.git).
   Rename the downloaded folder as /CoarseAIR/coarseair.

3. Ensure the following programs and libraries have been properly installed in your machine:

   - A Fortran compiler (GCC or IFORT). The environmental variable `$FC` needs to be declared;
   - The OpenBLAS library (`https://www.openblas.net/`). The environmental variables `$OBLAS_INC, $OBLAS_LIB` need to be declared;

- The Parallel program (https://www.gnu.org/software/parallel/). The `parallel` command needs to point to the program executable;
- A Python3 compiler, and the python libraries numpy and pandas (for installing the libraries, type: pip3 install numpy; pip3 install pandas).

Note: For installing CoarseAIR in NASA Pleiades cluster, we suggest to add the lines of:
/CoarseAIR/coarseair/README/LinesTo_Add_ToBashrc/AddToBashrc_Pleiades.sh
to the local ∼/.bashrc.

4. Add the following lines to the ∼/.bashrc:

```
export WORKSPACE_PATH='path_to_workspace'
###################### CoarseAIR #######################
function COARSEAIR_UPDATE {
cd \$WORKSPACE_PATH/CoarseAIR/coarseair/
source ./scripts/building/BuildingInstalling.sh
}
#########################################################
```

`'path_to_workspace'` should be substituted with the absolute path to ∼/WORKSPACE/.

5. Type COARSEAIR_UPDATE. The command updates some CoarseAIR's environmental variables from the bash script:
/CoarseAIR/coarseair/scripts/building/BuildingInstalling.sh.

6. Type COARSEAIR_'CompFlg', with `'CompFlg'` equal to either `release` or `debug`. The command updates further CoarseAIR's environmental variables, and it generates the desired installation flags.

7. Finally, type COARSEAIR_INSTALL for building and installing each of the programs composing the CoarseAIR code. The installed programs can be found at:
/CoarseAIR/install/coarseair-1.1-'CompFlg'-'Compiler'/bin/.

2.1.2 Installing CoarseAIR (later times)

1. Type COARSEAIR_UPDATE.

2. Type COARSEAIR_'version', with `'version'` equal to either `release` or `debug`.

3. Type COARSEAIR_INSTALL.

Run this block of code whenever a change is made to the source code.

2.2 Examples

2.2.1 An Ab Initio Study of the O_2+O Kinetics

2.2.2 Computing Dissociation Thermal Rate Coefficients for N_2+N_2

The following lines describe the steps required for computing the trajectories:

1. Create the run folder /CoarseAIR/N4_NASA_Th.

2. Copy CoarseAIR.sh, the CoarseAIR launching script, from:
   /CoarseAIR/coarseair/scripts/launching/CoarseAIR.sh
   to the run folder.

3. Copy the example input folder from:
   /CoarseAIR/coarseair/input/N4/NASA/input/
   to the run folder.

4. Copy the main file from
   /CoarseAIR/N4_NASA_Th/input/CoarseAIR_Thermal_NASA.inp
   to /CoarseAIR/N4_NASA_Th/input/CoarseAIR.inp.

5. Modify /CoarseAIR/N4_NASA_Th/input/CoarseAIR.inp:

   - Translational Temperature [K] 1 = should correspond to the desired translational temperature.
   - Database Path = should correspond to the absolute path to /CoarseAIR/coarseair/dtb/.
   - Nb of Trajectories per Level / Bin = should correspond to the desired number of overall trajectories.

6. Upload the environmental variables by typing
   COARSEAIR_UPDATE; COARSEAIR_release.

7. From the run folder, execute the bash script by typing:
   bash ./CoarseAIR.sh.

Note: If the ./CoarseAIR.sh file is not modified, the code will be executed on one single processor. In order to parallelize CoarseAIR, the instructions outlined in Sec. 2.3.1 should be followed.

Finally, for obtaining the rate coefficients:

1. Modify /CoarseAIR/N4_NASA_Th/input/CoarseAIR.inp:

   - Preprocessing Energy Levels List? = no
   - Running Trajectories? = no
   - Postprocessing Trajectories? = yes

2. Upload the environmental variables by typing
   COARSEAIR_UPDATE; COARSEAIR_release.

3. From the run folder, execute the bash script by typing:
   bash ./CoarseAIR.sh.

2.3 General Overview

CoarseAIR is a collection of standalone Fortran 2008 programs for the computation of state- and group- specific rate coefficients starting from ab initio calculations. Each of the programs has a main file, located at /CoarseAIR/coarseair/app/, and it shares functions and subroutines (/CoarseAIR/coarseair/src/) with the others. The programs form a pipeline, which is controlled by the bash scripts in (/CoarseAIR/coarseair/scripts/executing/) and can be initiated by the CoarseAIR.sh file presented in the next subsection.

2.3.1 CoarseAIR.sh, the CoarseAIR Launching Script

CoarseAIR.sh file is the bash script that launches CoarseAIR.
It can be found at: /CoarseAIR/coarseair/scripts/launching/

In order to run the script:

1. The bash script must be first copied in a run-folder.
2. CoarseAIR's environmental variables must be uploaded through the terminal commands: COARSEAIR_UPDATE; COARSEAIR_release.
3. The parallelization variables can be changed in ./CoarseAIR.sh.
4. The bash script can be executed as: bash ./CoarseAIR.sh.

Through the following variables, CoarseAIR.sh control CoareAIR's parallelization:

- NProc = Number of processors.
- ParNodes:
  - ParNodes = 0. 1 node selected. Running the code via bash script (for local machines, e.g.: mimmo, entropy, etc.).
  - ParNodes = 1. 1 node selected. Running the code by submitting a .pbs file (for clusters, e.g.: NASA Pleiades).
  - ParNodes = N. N nodes selected. Running the code by submitting N .pbs files (for clusters, e.g.: NASA Pleiades).
- ProcType = Processor Types (Note: It applies only if ParNodes ≥ 1)
  - For NASA Pleiades: ProcType = 'san' or 'ivy' (Sandy Bridge/Ivy Bridge).
  - For UIUC Cluster: not required.
- Queue = Selecting the queue (Note: It applies only if ParNodes ≥ 1)
  - For NASA Pleiades: Queue = 'devel' or Queue = 'debug' or Queue = 'low' or Queue = 'medium' or Queue = 'long'.
  - For UIUC Cluster: not required.

- WallTime = Selecting the wall time, in hour (Note: It applies only if ParNodes ≥ 1)
  - For NASA Pleiades: e.g.: WallTime = 120
  - For UIUC Cluster: not required.

Other flags that may be useful to change are:

- SlncFlg = if 1, no ouptut at screen.
- MergeAllFlg = if 1, at the end of the trajectory calculations, CoarseAIR merges all the trajectory files in one.
- RmTrajFlg = if 1, at the end of the trajectory calculations, CoarseAIR deletes all the folders for the nodes.
- BinaryTrajFlg = if 1, the Statistics program looks for trajectories output written in a binary format.

2.3.2 CoarseAIR Input Files

CoarseAIR accepts the following files as input to the programs:

- CoarseAir.inp: The main input file, containing the main information about the chemical system, molecules, atoms, trajectory initial conditions, etc. Some of its lines, in particular, are common to all the programs:
  - System = The chemical system being analyzed (e.g., System = O3).
  - Nb of Species = Number of targets and projectiles (for 3-atoms and 4-atoms systems, Nb of Species = 2).
  - Species 1 = Chemical formula of the target (e.g., Species 2 = O2).
  - Species 2 = Chemical formula of the projectile (e.g., Species 2 = O).
  - Nb of Atoms = Number of atoms in the system.
  - Atom 1 = Chemical symbol of atom 1.
  - Mass [a.u.] of Atom 1 = Mass of atom 1, in atomic units.
  - Nb of Molecules = Number of distinct molecules in the system (e.g., Nb of Molecules = 1 for O3, and Nb of Molecules = 3 for CHN).
  - Molecule 1 = Chemical formula of molecule 1.
  - Nb of Atoms, Molecule 1 = Number of atoms in Molecule 1.
  - Database Path = Absolute path to the /CoarseAIR/coarseair/dtb folder.
  - Nb of Initial Molecules =Number of molecules from which the collision starts (Nb of Initial Molecules = 1 for 3-atoms systems and Nb of Initial Molecules = 2 for 4-atoms systems).

- GenerateLevels.inp: This file lists specifications for the molecular level generation process through the solution of the Schrödinger equation. The file dictates for which molecules levels should be generated, the resolution of discretized and quantized energy states, and the intervals of interest for rotational and vibrational quantum numbers.

- PlotPES.inp: This file lists specifications for plotting the potential energy surfaces (PESs) of the system of interest.

Examples of input files for the chemical systems implemented so far can be found at: /CoarseAIR/coarseair/input/).

Important notes for the format of input files:

- The order of the lines does not matter, if the line containing the number of objects to allocate comes before the description of the objects;

- The indentation of each line does not matter. However, the white spaces between the characters (including before and after the equal sign) do matter;

- As an example:

  - Case 1:

```
System = O3
Nb of Species = 2
    Species 1 = O2
    Species 2 = O
```

  - Case 2:

```
    Nb of Species = 2
Species 1 = O2
        Species 2 = O
System = O3
```

  - Case 3:

```
System = O3
Nb of Species  = 2
    Species  1 = O2
    Species 2 =  O
```

- Case 4:

```
System = O3
    Species 1 = O2
Nb of Species = 2
    Species 2 = O
```

Case 1 and 2 are equivalent, but they are not the same as Cases 3 and 4.

2.3.3 Tracking CoarseAIR Operations

The main files of CoarseAIR programs are located in the /CoarseAIR/coarseair/app/ directory. Within the first few lines of these files, there are `i_Debug` boolean flags. If set to `.True.`, the code will take more time to run, but it will output a .log file, useful for understanding the operations and/or debugging. (Note: In order for the flags to be activated, CoarseAIR must be recompiled).

SECTION 3

Diatomic Potentials and Computation of Ro-Vibrational Levels

Blurb

3.1 Diatomic Potentials

3.1.1 Choosing Chemical System, Molecules, Atoms, and their Properties

The diatomic potentials are controlled through the `CoarseAIR.inp`:

- `Diatomic Potential Model, Molecule 1 =` The desired diatomic potential of the primary molecule of study.

For verifying what diatomic potentials have been already implemented, please check `.../coarseair/src/D`
Within that file, find the string for the desired molecule (e.g. `'CO'`) and enter one of the diatomic potential strings into the above line (e.g. `'NASA'`).

3.1.2 Plotting Diatomic Potential

It is possible to plot the Diatomic Potential using the `PlotPES` script (see Section 4). If this is desired, modify the `PlotPES.inp` input file so that it says `PES or Diatomic = PES` in the appropriate line, and then execute the `PlotPES` code via the procedure in Section 4.2.1.

3.2 Computing Ro-vibrational Levels

3.2.1 Executing the Code

1. In ~`/WORKSPACE/CoarseAIR/` directory, create a run directory.

2. Once in the run directory, copy an example of input folder (e.g., `scp -r ../coarseair/input/'system'/input/ ./`)

3. Modify `./input/CoarseAIR.inp`:

   - Select:
     `Generating Energy Levels?  = yes`
   - Change the lines based on Sec. 2.3.2 and Sec. 3.1.1.

4. Modify `GenerateLevels.inp` following the instructions in the file.

5. Copy the bash file into the directory via
   `scp -r ../coarseair/scripts/launching/CoarseAIR.sh ./`

6. If needed, change the number of nodes and/or processors running the code within the bash file.

7. Upload the environmental variables:
   `COARSEAIR_UPDATE`
   `COARSEAIR_release`.

8. Once back in the run directory, execute the bash script by typing:
   `bash CoarseAIR.sh`.

3.2.2 Format of the Output File

The output files containing the list of levels just computed are located in:
`'rundir'/Test/'system'/'molecule'/'filename'`,
where `'filename'` has been specified in the `GenerateLevels.inp` input file. The columns in the data-file represent, from left to right:

1. Vibrational quantum number, v;

2. Rotational quantum number, J;

3. Internal energy, E (in Eh);

4. Life-time Parameter for quasi-bound levels, Γ;

5. Location of the minimum potential, $r_{V^{Min}(J)}$ (in a.u.);

6. Location of the maximum potential, $r_{V^{Max}(J)}$ (in a.u.);

7. Minimum of the J-dependent diatomic potential, $V^{Min}(J)$ (in Eh);

8. Maximum of the J-dependent diatomic potential, $V^{Max}(J)$ (in Eh);

9. Vibrational time constant, τ_v (in a.u.);

10. Location of the inner turning points, r_{In} (in a_0);

11. Location of the outer turning points, r_{Out} (in a_0).

3.2.3 Visualization of the Level Properties

In order to visualize the level properties, we suggest to post-process the output file using Paraview (https://www.paraview.org/), following these instructions:

1. Ensure the data file is in the .csv format. The top 3 rows will contain filler formatting; remove this before plotting.
2. Open the .csv file in Paraview.
3. Run the TableToPoints filter on the .csv file, with `rIn[a0]` on the x-axis, `jqn` on the y-axis and `E[Eh]` on the z-axis.
4. Color the points based on `vqn`, and either
   (a) increase the point size to 10, or
   (b) plot in Point-Gaussian.
5. Downscale the y-axis by a factor of 10, and upscale the z-axis by the conversion factor from hartree to electron volts ($1\,E_h = 27.211386246\,eV$).
6. Repeat steps 3-5, except specify instead `rOut[a0]` on the x-axis.

3.2.4 MATLAB Program for Ro-Vibrational Levels Calculation

A version of the code is available in `coarseair/scripts/postprocessing/MCoarseAIR/` as separate MATLAB scripts. In particular, the `TestNumerov.m` script under `Stand-Alone` can be helpful as a top-down overview of the structure of the CoarseAIR code used to compute rovibrational levels. To modify and run the code, ensure that the path to the `MCoraseAIR` folder is added to the MATLAB interface (e.g. using Set Path).

3.3 References

[1] Sandvik, A., ”Numerical Solutions of the Schrödinger Equation,” Boston University, 2016.

[2] Johnson, B.R., ”The renormalized Numerov method applied to calculating bound states of the coupled-channel Schroedinger equation,” *The Journal of Chemical Physics*, 16 Jun 1978.

SECTION 4

Potential Energy Surfaces

Blurb

4.1 Potential Energy Surfaces (PES)

4.1.1 Input File

The PESs are controlled through the `CoarseAIR.inp`. In particular:

- `Diatomic Potential Model, Molecule 1 =` The desired diatomic potential of the primary molecule of study.
- `Nb of PESs =` number of PES models being analyzed
- `Model, PES 1 =` The desired PES model for the first surface
- Any of the other inputs mentioned in Sections 2 and 3.

For verifying what PESs have been already implemented, please check `.../coarseair/src/PESs/PES_Factory_Class.F90`. Similarly to the Diatomic Potential methodology, find the string of the desired molecule or system, and enter one of the corresponding strings into the lines above.

4.2 Computing the Potential Energy at Specific Atomic Configurations

4.2.1 Executing the Code

1. In ∼/WORKSPACE/CoarseAIR/ directory, create a run directory.
2. Once in the run directory, copy an example of input folder (e.g., `scp -r ../coarseair/input/'system'/input/ ./`)
3. Upload the environmental variables:
   `COARSEAIR_UPDATE`
   `COARSEAIR_release`
4. Source the PlotPES bash script by typing:
   `source ∼/WORKSPACE/CoarseAIR/coarseair/scripts/executing/PlotPES.sh`
5. Modify `./input/CoarseAIR.inp` based on Sec. 2.3.2 and Sec. 4.1.1.
6. In `./input/PlotPES.inp`, change the lines as preferred, following the instructions in the file.
7. From the run directory, type: `PlotPES`.

4.2.2 Plotting the Potential Energies

1. In paraview, load the `'rundir'/Test/PlotPES/'pesnb'/PESFromGrid.csv` file.
2. Run TableToPoints filter on the .csv, plotting the desired distances on the x- and y-axes and `E` on the z-axis.
3. Run a Delaunay-2D filter on these points for fitting them and creating the surface.

4.3 References

[1] Venturi, S., Jaffe, R., and Panesi, M., "A Bayesian Machine Learning Approach to the Quantification of Uncertainties on Ab Initio Potential Energy Surfaces," 15 Feb 2000.

[2] Jaffe, R., Schwenke, D. L., and Panesi, M., "First Principles calculation of heavy particle rate coefficients," Chapter 2, 29 Dec 2014.

SECTION 5

Preprocessing

Blurb

5.1 Splitting the Molecules' Level Lists

5.1.1 Input File

The resolution required to the rate coefficients (ro-vibrational state-to-state, or vibrational-specific, or coarse-grained) must be specified through `CoarseAIR.inp` before running trajectories. In particular:

- `Levels File Name, Molecule 1 =` name under which levels file for molecule 1 is to be saved, along with format (e.g. `FromUMN_Sorted.inp`)
- `Min Energy for Cutting Levels, Molecule 1 =` lower bound for energy cutoff for each level
- `Min Time for Cutting Levels, Molecule 1 =` lower bound for time cutoff for each level
- `Sorting Method, Molecule 1 =` method of binning the levels for calculating weighted average of rates. The methods are:
  - State-specific: all states are computed and weighted in a single bin.
  - Vib-specific: the states are binned based on vibrational quantum number, and weighted within each bin.

  - RoVib-CG: the states are binned based on local regions of vibrational and rotational quantum numbers in the J-v space, and weighted within these regions.
  - From-File: the states are binned with a custom binning input, taken from the specified input file.

- Any of the other inputs mentioned in Sec. 2.3.2 and, if running those sections, Sec. 3.1.1 and Sec. 4.1.1.

5.1.2 Executing the Code

1. Create a run directory in ~/WORKSPACE/CoarseAIR/;
2. Once in the run directory, copy an example of input folder (e.g., scp -r ../coarseair/input/'System'/input/ ./)
3. Modify ./input/CoarseAIR.inp:
   - Select:
     Preprocessing Energy Levels List? = yes
   - Change the lines based on Sec. 2.3.2 and Sec. 5.1.1
4. Copy the CoarseAIR.sh file and launch it (see Sec. 2.3.1).

5.1.3 Output File Formats

The output files are located in /Test/'System'/'Molecule'/Bins_#/, denoted as `QNsFirst.csv` and `QNsEnBin.csv`. The former file contains the following data columns, from left to right:

1. Vibrational quantum number, v;
2. Rotational quantum number, J;
3. Number of levels with given v and J;
4. Percent of total trajectories assigned to this level

The latter file contains the following data columns, from left to right:

1. Index of specified level;
2. Vibrational quantum number, v;
3. Rotational quantum number, J;
4. Energy of level, E (in Eh);
5. Degeneracy of level, for computing average rates;
6. Index of bin to which level belongs

SECTION 6

Simulating Trajectories

Blurb

6.1 Executing the Running Trajectories Program

1. Once in the run directory, modify ./input/CoarseAIR.inp:
   - Select:
     Running Trajectories? = yes
   - Change the lines based on Sec. 2.3.2 and Sec. 6.2.
2. Copy the CoarseAIR.sh file and launch it (see Sec. 2.3.1).

6.2 Input File

The following lines in `CoarseAIR.inp` control how the code runs collision trajectories:

- `Relative Translational Energy Model =` distribution used to sample the energy of the system. Current options are:
  - `Uniform` Distribution, at a specified translational energy
  - `Boltzmann` Distribution, at a specified translational temperature
  - `FixedTotEn`, or a specified, fixed total energy of the system
- `Nb of Translational Temperatures =` number of translational (thermodynamic) temperatures at which to compute rates

- `Nb of Internal Temperatures =` number of internal (rovibrational) temperatures at which to compute rates
- `Using Proportional Allocation? =` flag for specifying whether to evenly allocate trajectories to each level bin, or instead to allocate proportionally with the number of levels in each bin
- `Nb of Total Trajectories =` number of total trajectories to run. Specify only if using proportional allocation
- `Nb of Trajectories per Level / Bin =` number of trajectories to allocate for each level or bin. Specify only if not using proportional allocation
- `Generate a CPU-Clock Dependent Seed? =` specifies whether to generate the random seeding based on the computer's clock
- `Initial Random Nb Seed =` seed number to be used, if not clock-based
- `Initial Distance between Target and Projectile =` distance between target (stationary) and projectile species, in a.u. (e.g. 25.d0)
- `Nb of PESs =` number of distinct PES models to be used
- `Model, PES 1 =` model to be used for the first PES, from the `PES_Factory_Class.F90` file mentioned above (`... 2 =` 2nd, etc)
- `Degeneracy, PES 1 =` degeneracy of first PES, or number of PESs that are equivalent to the first PES
- `Initial Impact Parameter Model =` method of binning initial impact parameters b_i for simulating trajectories:
  - `Normal`: sample b_i from a normal distribution, with a specified mean b
  - `Random`: sample b_i randomly (uniformly), with specified mean b
  - `Stratified`: sample b_i with based on annular or circular strata, with specified size and proportion of trajectories per stratum
  - `FixedAngMom`: sample b_i based on a fixed total angular momentum, specified by the user
- `Method for Initial Vibrational Quantum Nb =` how to choose initial vibrational quantum number v_i for computing trajectories:
  - `Fixed`: specify v_i to use
  - `Uniform`: choose v_i randomly based on a uniform distribution of all v
  - `Boltzmann`: choose v_i randomly based on a Boltzmann distribution of all v
  - `MostProbable`: choose v_i to be the most probable v for the given initial state

  - `ArbitraryDistribution`: choose v_i from an arbitrary distribution, specifying its unit seed

- `All / Only Even / Only Odd values for Rotational Quantum Nb =` whether to sample over all rotational quantum numbers J, only the even ones, or only the odd ones

- `Lower Bound for Position of Centrifugal Barrier Max =` lower bound of the expected location of the peak potential of the centrifugal barrier. Similarly for `Lower Bound for Position...`

- `Write the ____ File? =` specifies whether to write the given file. If the line for a given file is not present in `CoarseAIR.inp`, it may be found in `Input_Class.F90`

- `Try to increase stepsize? =` specifies whether the numerical integration stepsize should be allowed to increase, or instead held fixed

- `Backward Integrating? =` flag for whether to use forward or backward integration

- `Initial time step =` starting t_0, in a.u.

- `Stepsize increase damping factor =` factor by which stepsize increases should be scaled for reasonable convergence

- `Max distance between atom pair [bohr] =` specifies maximum allowable distance between any two atoms, in bohr, past which integration should cease

- `Max trajectory time [a.u.] =` specifies maximum allowable flight time, in a.u., past which integration should cease

- Any of the other inputs mentioned in Sec. 2.3.2, Sec. 5.1.1, Sec. 3.1.1, and Sec. 4.1.1 that are relevant here.

6.3 Output File Formats

The output files are located in `Test/T_'T1'_'T1'/Bins_X_Y`. Each Bin folder contains trajectories simulated using levels from that Bin, and there are subfolders for each `Node` and `Processor`, with the ability to view trajectories run on levels from that processor. The trajectory data is located in `trajectories.csv`, and contains the following data columns, from left to right:

1. Index of specified trajectory;

2. Index of the PES used with that trajectory;

3. Maximum allowable impact parameter, b_{max};

4. Initial impact parameter, b_i;

5. Initial rotational quantum number, J_i;
6. Initial vibrational quantum number, v_i;
7. Initial arrangement of species, arr_i;
8. Final rotational quantum number, J_f;
9. Final vibrational quantum number, v_f;
10. Final arrangement of species, arr_i

Note that J_f and v_f are not always integers, and so they may have to be rounded in post-processing. Additionally, the arrangements arr_i and arr_f are expressed as $16m+n$, where m and n use the following numerical convention:

- m refers to the atom pair in the final molecule (Pair 1 is $N_A N_B$, Pair 2 is $N_A N_C$, Pair 3 is $N_B N_C$)
- n denotes the final state of the system:
  - $0 =$ bound
  - $1 =$ quasi-bound
  - $2 =$ dissociating
  - $3 =$ predissociating

The reason for multiples of 16 stems from the fact that the code, in its current state, can handle trajectories of systems of 4 species, each with the possibility of being in one of the above 4 states.
Additionally, for visualizing the trajectories, within the `Bins_X_Y/Node_X/Proc_X` folder, there are additional folders for each trajectory, with a series of files named `XXEvo.vtk.'timestep'`, that can be imported into Paraview or MATLAB.

SECTION 7

Post-processing the Trajectories: Cross Sections and Rate Coefficients

This part of the manual deals primarily with post-processing the colissional data, which means computing the trajectory statistics in order to obtain cross sections and rate coefficients. The section assumes that, for the current chemical system, levels have been already pre-processed and trajectory already simulated (see Sec.s 5 and 6).

7.1 Post-processing Trajectories

7.1.1 Executing the Code

1. Once in the run directory, modify ./input/CoarseAIR.inp:
   - Select:
     Postprocessing Trajectories? = yes
   - Change the lines based on Sec. 7.1.2
2. Copy the CoarseAIR.sh file and launch it (see Sec. 2.3.1).

7.1.2 Input File

The following lines in `CoarseAIR.inp` control how the code computes cross-sections and rate coefficients:

- Distinguish between PESs? = yes/no flag for whether to split the collisions written in the trajectory files based on the PESs on which they have been computed. If yes, a set of cross sections will be computed for the first PES, another set for the second PES, etc.

- Preserving Even / Oddness of Condition 1 = yes/no flag for whether to apply quantum rules.
  For 3-atoms systems, condition 1 is on the j-q.n., condition 2 is on the v-q.n., and condition 3 is on the arrangement.
  For 4-atoms systems, condition 1 is on the j-q.n. of the first molecule, condition 2 is on the v-q.n. of the first molecule, condition 3 is on the j-q.n. of the second molecule, condition 4 is on the v-q.n. of the second molecule and condition 5 is on the arrangement;

- Identical Diatoms? = yes/no flag for whether to compute statistics accounting for identical diatoms (i.e., AB + AB).
  Note: in order for the diatoms being considered identical, they also need to be treated identically (e.g., both vibrationally-specific or both coarse-grained with the same number of groups);

- Writing Trajectory Files in Binary Format? = yes/no flag for whether to re-write the trajectory ASCI files in binary format;

- Distinguish Exchanges between Each Other? = yes/no flag to specify if exchanges between a molecule with two or more identical atoms should be considered two separate exchange reactions;

- Distinguish Exchanges from Inelastic? = yes/no flag to specify if an atom exchanging with another atom of the same element should be considered an exchange rather than an inelastic collision;

- Writing Rate Files in Binary Format? = yes/no flag to specify whether the output files for rate coefficients should be written in binary format;

- Temperature for Distribution Function Computation [K] = temperature to specify an additional equilibrium distribution .

7.2 Output File Formats

7.2.1 Cross Sections

The output file for cross sections is statistics.csv, located in Test/T_'TTran'_'TInt'/Bins_'iBin'_'jBin'/. If the collision is represented with $A_i + B_j \longrightarrow C_k + D_l$, then iBin and jBin indicate the levels/groups of A and B, respectively (Note: If a 3-Atoms system is being simulated, then jBin=0). The data in the sub-columns correspond to, from left to right:

1. Rotational quantum number of molecule C, J_k;

2. Vibrational quantum number of molecule C, v_k;

3. Rotational quantum number of molecule D, J_l;

4. Vibrational quantum number of molecule D, v_l;
5. Arrangement of final state (see Sec. 6.3);
6. Rotational quantum number of molecule A, J_i;
7. Vibrational quantum number of molecule A, v_i;
8. Rotational quantum number of molecule B, J_j;
9. Vibrational quantum number of molecule B, v_j;
10. Arrangement of initial state (see Sec. 6.3);
11. Cross-section area, σ_i (in a.u.^{2});
12. Standard Deviation of integral cross-section area (in a.u.^{2}).

Note: If a 3-Atoms system is being simulated, the two columns corresponding to molecule B and the two corresponding to molecule D are missing.

7.2.2 Rate Coefficients

We are now looking for the rate coefficients corresponding to the collisions $A_i + B_j \longrightarrow \cdots$.
To this collision initial state is given an identification index, *iProc*.
For a 3-atoms system, $iProc = i$ (i.e., it is equal to the level/group index of molecule A).
For a 4-atoms system, instead, it is given by the following formula:

$$iProc = (i-1) * N^B_{Bins} + j \tag{1}$$

where where j and N^B_{Bins} indicate the initial level/group and the total number of levels/-groups of molecule B, respectively. The list of rate coefficients is output in Proc'iProc'.csv, located in the Text/'System'/Rates/T_'TTran'_'TInt'. Opening the file in a spreadsheet editor, the following information is displayed in the header, from top to bottom, left-to-right:

- `System`: the chemical system being simulated;
- `PES`: the PES used in the simulation. If multiple PESs have been employed through an MC strategy, `Merged` will be shown;
- `In. Molecules`: name of the molecules from which the collisions started
- `No Processes`: overall number of possible processes taking place because of the collision;
- `No Trajectories`: number of overall trajectories simulated from the initial state being analyzed;
- `In. Lev.s/Bins`: index for the initial levels/groups being analyzed ;

- `Part. Func.s Ratios`: contribution to the molecule partition functions of `In. Lev.s/Bins`;
- `Tran Temperature`: translational temperature used in simulation;
- `Int Temperature`: internal temperature, used for sampling the levels inside each group.

The three columns below the header correspond to, from left-to-right:

1. Process index, d (see Sec.7.2.3);
2. Rate coefficient, $k_{i,j}$ (if 3-atoms system) or $k_{ij,kl}$ (if 4-atoms system); in $[cm^s/s]$;
3. `Rate SD`: $k^{SD}_{i,j}$ (if 3-atoms system) or $k^{SD}_{ij,kl}$ (if 4-atoms system); in $[cm^s/s]$.

7.2.3 Processes Indexing

For the following lines,:

$$\begin{cases} d_{mol} = 1, & \text{if } mol\text{-th molecule dissociated;} \\ d_{mol} = s+1, & \text{otherwise ,} \end{cases}$$

where s is the molecule's final level/group.

For 3-atoms systems, the process indexes, d, are assigned based on the following cases:

- Inelastic collision, $AB_i + C \longrightarrow AB_j \equiv C$:

$$d = 1 + d_1, \tag{2}$$

- Exchange to the second pair, $AB_i + C \longrightarrow AC_k \equiv B$:

$$d = 1 + (\mathcal{N}_1^G + 1) + d_2. \tag{3}$$

- Exchange to the third pair, $AB_i + C \longrightarrow BC_l \equiv A$:

$$d = 1 + (\mathcal{N}_1^G + 1) + (\mathcal{N}_2^G + 1) + d_3. \tag{4}$$

For 4-atoms systems, the process indexes, d, are assigned based on the following cases:

- Inelastic collision, $AB_i + CD_j \longrightarrow AB_k + CD_l$:

$$d = 1 + (d_1 - 1)(\mathcal{N}_6^G + 1) + d_6. \tag{5}$$

- Exchange to the second pair, $AB_i + CD_j \longrightarrow AC_m \equiv BD_n$:

$$\begin{aligned} d = 1 &+ (\mathcal{N}_1 + 1)\ (\mathcal{N}_6 + 1) + \\ &+ (d_2 - 1)\ (\mathcal{N}_5^G + 1) + d_5. \end{aligned} \tag{6}$$

- Exchange to the third pair, $AB_i + CD_j \longrightarrow AD_p + BC_q$:

$$\begin{aligned} d = 1 &+ (\mathcal{N}_1 + 1)\ (\mathcal{N}_6 + 1) + \\ &+ (\mathcal{N}_2 + 1)\ (\mathcal{N}_5 + 1) + \\ &+ (d_3 - 1)\ (\mathcal{N}_4^G + 1) + d_4. \end{aligned} \tag{7}$$

Note: $d = 1$ is not outputted anymore by CoarseAIR, but it has been left for consistency with old versions.

SECTION 8

PyCoarseAIR: A Python Tools For Postprocessing

`PyCoarseAIR`:
PyCoarseAIR is a collection of Phyton scripts for postprocessing the databases of *ab initio* rate coefficients resulting from the CoarseAIR code [1] (https://github.com/simoneventuri/CoarseAIR) implementing the quasi-classical trajectories (QCT) method.

8.1 Quickstart

8.1.1 Requirements

1. The following Python libraries need to be installed:

   - `numpy`, fundamental package for scientific computing (https://numpy.org/);
   - `matplotlib`, for static, animated, and interactive visualizations (https://matplotlib.org/);
   - `pandas`, data analysis and manipulation tool (https://pandas.pydata.org/);
   - `csv`, implementing classes to read and write tabular data in CSV format (https://docs.python.org/3/library/csv.html);
   - `h5py`, a Pythonic interface to the HDF5 binary data format (https://www.h5py.org/).

   All these libraries can be easily installed through pip3 (i.e., by typing: "`pip3 install numpy`" from the terminal).

2. The PyCoarseAIR code can be cloned as a GitHub repository: https://github.com/simoneventuri/PyCoarseAIR (i.e., from the terminal, type:

"`git clone https://github.com/simoneventuri/PyCoarseAIR`").
For the remaining part of this manual, the path to the local PyCoarseAIR folder will be referred to as
`PyCoarseAIRFldr`.

3. The .hdf5 file corresponding to the *ab initio* database for the desired chemical system needs to be downloaded from https://www.chess.aerospace.illinois.edu/databases. For the remaining part of this manual, the path to the folder containing the downloaded file will be referred to as `DtbHDF5Fldr`, while `SystOI` will be used for indicating the name of the desired chemical system.

4. In `PyCoarseAIRFldr/exec/PyCoarseAir.py` (i.e., the main file of the program), the following paths need to modified:

   - `PyCoarseAIRFldr`, based on point 2 of this list;
   - `DtbHDF5Fldr`, based on point 3 of this list;
   - `DtbWriteFldr`, pointing to the folder where PyCoareseAIR is required to write the postprocessed kinetics database;
   - `OutputWriteFldr`, pointing to the folder where PyCoareseAIR is required to save its output files and plots.

8.1.2 Executing PyCoarseAIR

The code can be executed from the `PyCoarseAIRFldr/exec/` folder by typing:
"`python3 ./PyCoarseAir.py PyCoarseAIRFldr/input/SystOI`"
(e.g., "`cd /home/venturi/WORKSPACE/PyCoarseAIR/exec/;`
`python3 ./PyCoarseAIR.py ../input/N3_NASA/`").

8.1.3 PyCoarseAIR Output Files

The files containing the desired rate coefficients can be found in the folder `PyCoarseAIRFldr/kinetic/SystOI/`, while the lists of levels/groups for each of the molecules in the system are located in `PyCoarseAIRFldr/thermo/SystOI/`.
In `PyCoarseAIRFldr/thermo/SystOI/`, the code also writes one file for each of the molecules containing the levels / groups distribution functions at the initial temperature. All the output files just mentioned are compatible with the PLATO library by Dr. A. Munafò [2-5] for the solution of the master equation and CFD calculations.

8.2 A Brief List of PyCoarseAIR's Fundamental Input Variables

- `TranVec`: Vector of translational temperatures for which we are interested in loading the corresponding rate coefficients;
- `T0`: Initial temperature of further heat bath simulations. The distribution functions corresponding to T0 will be computed and written for all the molecules composing the mixture;

- `WriteDiss_Flg`: With `WriteDiss_Flg=True`, the dissociation rate coefficients will be uploaded from the database and written locally.
- `CorrFactor`: If `CorrFactor` $\neq 1$, the dissociation rates loaded from the database are multiplied by `CorrFactor` before being written;
- `WriteDissInel_Flg`: With `WriteDissInel_Flg=True`, the dissociation-inelastic rate coefficients (e.g., the ones corresponding to the processes AB+CD=AB+C+D) will be uploaded from the database and written locally (only for 4-atoms system).
- `WriteInel_Flg`: With `WriteInel_Flg=True`, the inelastic rate coefficients will be uploaded from the database and written locally.
- `WriteExch_Flg`: With `WriteExch_Flg=True`, the exchange reaction rate coefficients will be uploaded from the database and written locally.
- `WriteExoth_Flg`: With `WriteExoth_Flg=True`, the exothermic rate coefficients will be written locally. With `WriteExoth_Flg=False`, the endothermic rate coefficients will be written locally.
- `WriteQB_IntFlg`: Integer flag for considering the quasi-bound levels.
  With `WriteQB_IntFlg=0`, only bound levels are considered; with `WriteQB_IntFlg=1`, only quasi-bound levels are considered; with `WriteQB_IntFlg=2`, both bound and quasi-bound levels are considered.
- `WriteFormat`: With `WriteFormat='PLATO'`, the utput data will be produced in PLATO's [2-5] format. With `WriteFormat='csv'`, the utput data will be written in .csv files. With `WriteFormat='custom'`, the user can customize the format by modifying the files `/PyCoarseAIR/src/Objects/Molecule.py` and/or `/PyCoarseAIR/src/Objects/System.py`

8.3 References

[1] Venturi *et al.*, In preparation

[2] A. Munafò, A. Alberti, C. Pantano, J. Freund, and M. Panesi, "Modeling of laser-induced breakdown phenomena in non-equilibrium plasmas," in 2018 AIAA Aerospace Sciences Meeting.

[3] A. Munafò, N. N. Mansour, and M. Panesi, "A reduced-order NLTE kinetic model for radiating plasmas of outer envelopes of stellar atmospheres," Astrophys. J. 838, 126 (2017).

[4] A. Munafò, Y. Liu, and M. Panesi, "Physical models for dissociation and energy transfer in shock-heated nitrogen flows," Phys. Fluids 27, 127101 (2015).

[5] A. Munafò, A. Alberti, C. Pantano, J. Freund, and M. Panesi, "A computational model for nanosecond pulse laser-plasma interactions", J. Comput. Phys. 406, 109190 (2020).