PES2MP - Quick Guide

Quantum Dynamics Lab, Indian Institute of Technology Ropar

v. 2025.R3

Section 1

Introduction

1.1 About PES2MP program

PES2MP is a **potential energy surface (PES)** builder that can automate PES **generation** and **augmentation** using **Neural Networks** (N-dimensional augmentation) while also automating **analytical fitting** of PES into **Radial Terms** using Legendre polynomials (2D PES) and Spherical Harmonics (4D PES) for studying **bound states** and **cross-sections** (for rotational (de-)excitation) of any rigid rotor with an atom (2D PES) or another rigid rotor (4D PES).

Such 2D/4D collisions are important for astrochemical applications where molecules are usually **not** in local thermodynamic equilibrium (LTE) and collide primarily with He and H_2, the two most abundant molecules in gaseous clouds, exchanging energy. The theoretically obtained collisional rate coefficients are useful for studying observed line intensities (from ISM gas clouds), giving us physical properties like temperature, density, and chemical composition of these non-LTE regions. Another application of these collisions is the experimental study of ultracold collisional pairs, where we can use external fields to influence reaction dynamics and even tune interaction strength.

The current Python-based program automatically creates PES for various collisional pairs, e.g. atom - atom (1D), rigid rotor - atom (2D), and rigid rotor - rigid rotor (4D). The program calculates and moves respective Rigid Rotor(s) (RR) to the center of mass (COM) and generates PES using Psi4 (both for rough estimation and CBS calculations). The program can also generate input files for external calculations using Molpro/Gaussian/Psi4, along with publication-ready plots with customization (title, axis, etc.) in pdf/eps formats. The users can also generate PES using sparse coordinates for heavier molecules. This sparse PES can then be augmented using an ensemble Neural Networks (NN) model based on the high-level library of TensorFlow, i.e. **Keras**, where users can choose multiple models to avoid dependency on a singly trained machine learning (ML) model. The final part of the program does multipole expansion of the PES to obtain radial terms that can be analytically fitted into a series of Slater functions for various MOLSCAT calculations, such as cross-sections for rotational transitions and bound states, with and without external magnetic fields.

To begin any collisional study, the users must provide atoms and bond lengths (optimized or experimental) for each collider, along with the charge and multiplicity of the fragments. The users can set parameters to generate PES internally and/or generate Psi4, Molpro, and Gaussian input files (CP corrected or CBS), simultaneously for external calculations on servers. Apart from the step size of Jacobi coordinates, users can leave the rest of the parameters to their default. There are auxiliary bash and Python scripts to run calculations, extract PES, and convert it to cm^{-1} scale.

For the NN model, a cutoff value for high energy is required, as the program separates minima and high-energy regions to preserve the features that are lost when both regions are trained together. The accuracy of the final result (NN-generated PES), both from individual and ensemble NN models, is verified using a residual plot.

Currently, the program uses Scipy and pyshtools to carry out multipole (MP) expansion of 2D and 4D PES by calculating Legendre and spherical harmonics terms, respectively. The analytical fitting of these radial terms can be carried out using either $\sum_i \alpha_i R^{-\beta_i}$, $\sum_i \alpha_i e^{-\beta_i R}$, or mixed functions. The program can also recreate the surface from these analytically fitted terms to check the accuracy of the fit (a residual plot is also generated by PES2MP). The final output, i.e., coefficients of these mathematical expressions, is provided in MOLSCAT readable format.

The subsequent sections, i.e,. Section 2 and Section 3 cover the installation and usage process (of PES2MP) using Character User Interface (CUI) and Graphical User Interface (GUI), respectively. Section 4 covers basic GUI examples to make users familiar with various input files of PES2MP. The final section, i.e., Section 5, discusses briefly the capabilities of PES2MP.

Section 2

Installation and Usage using CUI

This section covers installation instructions using the Character User Interface.

2.1 Instructions for Installation

There are two options for makefiles :

1. Quick install :: **Filename**: *install_pes2mp_quick.sh* || **Conda environment**: *pes2mp_q*)
2. Conda install :: **Filename**: *install_pes2mp.sh* || **Conda environment**: *pes2mp*)

The quick version uses **python -m pip** command (faster but does not check the compatibility among packages) while the normal version uses **conda** command to install required packages. While the conda installation is recommended, it can sometimes be painfully slow to solve environment conflicts among packages, and therefore, it is suggested that users install both makefiles in separate terminals. If the conda installation doesn't finish in 24 hours (i.e., conda is stuck in a loop and keeps solving the environment), just cancel the installation and use the quick version. If both install normally, users can use either of the two environments to run the program.

**Users can optionally install GPU-accelerated TensorFlow (requires editing of install file: See Section 2.2) in the quick environment (MacOS only). This will allow them to switch between the GPU and the CPU to create the NN models.*

2.1.1 Prerequisite

Anaconda
Install anaconda by visiting anaconda webpage i.e. `https://www.anaconda.com/download`.
It can create environments with specific Python versions, and two environments will not be affected by any change in the base (root) environment. For our work, we shall create **pes2mp** environment utilizing conda and **pes2mp_q** environment utilizing pip for installation of various packages (**makefile is provided**). The essential packages are: Psi4, pyshtools, TensorFlow, SciPy, Matplotlib, tqdm, and many more.

2.1.2 Installing and Running PES2MP

1. Download the zip file from GitHub
2. Go to "**makefile**" folder
3. Open **two** terminal windows (one for quick install and the other for conda install) and type the following commands: Users can install either the quick version or the normal version. The instructions are provided for both versions (read above). The quick version installation does not interfere with the conda installation. The quick version installs packages without rigorously checking for inter-compatibility and is therefore faster to install. However, conda installation ensures that all packages are compatible and are therefore slow to install.

Terminal 1 (Quick Install)		Terminal 2 (Conda Install):
`$ chmod +x install_pes2mp_quick.sh`	||	`$ chmod +x install_pes2mp.sh`
`$ ./install_pes2mp_quick.sh`	||	`$ ./install_pes2mp.sh`

4. Adding pes2mp command to bashrc/bash_profile/zshrc (Optional):

   After installation, the command to run the program can be shortened by adding a few lines in bash_profile/zshrc (Mac OS) or bashrc (Ubuntu). This is not essential, but it makes the program more user-friendly. Modify .bash_profile/zshrc (MacOS) or .bashrc (Most Linux, including Ubuntu) in the terminal by typing:

   Mac OS || Ubuntu (Linux):

   ```
   $ open -t ~/.bash_profile   ||   $ gedit ~/.bashrc
   ```

   New Mac machines should use 'open -t ~/.zshrc'. Now the GUI text editor will open the bashrc/bash_profile/zshrc. Paste the following lines at the bottom of the text file, save, and exit.

   ```
   pes2mp ()
   {
           python3 pes2mp.py $1
   }
   ```

5. **To run the program on Terminal**:

   (a) **Copy** the **pes2mp.py** and **pes2mp_driver.py** files into a folder of your choice.

   (b) Now **copy the input file(s)** that you want to execute (e.g. pesgen1D.py) in that folder.

   (c) Open the **terminal** at the folder (make sure the conda base environment is showing).

   (d) **Activate** conda **environment** (pes2mp) using either command depending on installation:

   ```
   $ conda activate pes2mp   ||   $ conda activate pes2mp_q
   ```

   (e) Now type the command in the terminal (e.g., Input file = pesgen1D.py):

   Step 4 completed || Step 4 incomplete:

   ```
   $ pes2mp pesgen1D   ||   $ python3 pes2mp.py pesgen1D
   ```

   The first command works only if the bashrc/bash_profile/zshrc is edited. If somehow, the bashrc could not be found/edited, the users can still run the program using the longer script:

   Important: The input file must have the "**.py**" extension e.g. plot2D.py. However, while executing (running) the program, simply type **pes2mp plot2D** without the "**.py**" extension.

6. Users can keep multiple input files (within the same folder) with the same 'Project_name' variable (the variable is set inside the input files) to execute them in sequence, such as:

   (a) PES Generation (Internal Psi4 or External Psi4, Molpro or Gaussian)

   (b) Optional: NN Augmentation and PES plotting

   (c) Optional: Fitting PES into a Function (for good fit, use the same function for radial terms)

   (d) Multipole Expansion of PES, and

   (e) Fitting Radial Terms into a function (this automatically gives the &POTL file (*MOLSCAT_POT.txt*) for the functions to be used in MOLSCAT).

   After PES generation (in cm^{-1} scale), the subsequent PES2MP input calculations can be carried out in sequence. For external PES/radial terms, users must ensure that the file name and data separator are correct. The variable names for PES/radial terms (and other options) are set inside the respective input files.

The above task can be automated using the GUI interface. Several templates for each collision type (1D/2D/4D) are provided with the PES2MP program. The input file (templates) examples for GUI execution are provided inside the **GUI_examples** folder. The users can simply reuse the folder depending on the task and rename it. The environment name, folder location, and Project name are set in the GUI interface. For more details on using the GUI, see the next Section.

2.2 PES2MP Input Files *vs* Standalone Python Scripts & Auxiliary Codes

2.2.1 PES2MP Input Files:

As discussed earlier, there are input files (in Python format) that are read by the PES2MP program. To run the PES2MP program, keep the 3 Python files, i.e.:

```
(1) pes2mp.py, (2) pes2mp_driver.py and (3) inputfile.py
```

in the same folder. After which follow Step 5 of the previous subsection to activate the conda environment for pes2mp and run the program.

2.2.2 Auxiliary Codes:

For calculating the **PES using external programs** like Molpro, Psi4, and Gaussian, the batch scripts are provided in **/input_files/aux_scripts/** folder. Similarly, once the calculations are finished, a Python script to collect the results (in the required format) is provided as well.
To run auxiliary scripts, use the following commands:

```
Python scripts : $ conda activate pes2mp
                 $ python3 script.py

Bash scripts   : $ chmod +x script.sh
                 $ ./script.sh
```

- Apart from PES calculations and results, there are similar scripts to generate **MOLSCAT** input files, run them, collect results (cross-sections), and calculate **rate coefficients** for excitation and de-excitation transitions.
- There is also a Python script to convert energy from Hartree to cm^{-1} (which takes the asymptotic energy of each angle to **remove size-consistency error** as faced by CI and F12 methods).

GPU acceleration for NN model training (on MacOS only): The script to install required libraries is provided in the quick version of the PES2MP install. Uncomment the same and the program will automatically use the available Apple Silicon GPU for training the NN model with $2 - 10\times$ speed gain (no separate command or modification of code is needed).

```
# To use GPU acceleration in MacOS, uncomment the lines below
#python -m pip uninstall tensorflow
#python -m pip install tensorflow-macos
#python -m pip install tensorflow-metal
```

∗ For more information on input/output files, refer to the programmer's manual.

Section 3

Installation and Usage using GUI

3.1 Instructions for Installation

After downloading and extracting the zip file for PES2MP from GitHub, run the following command to open the PES2MP installer.

```
$ python3 installer_gui.py
```

This will open the following window:

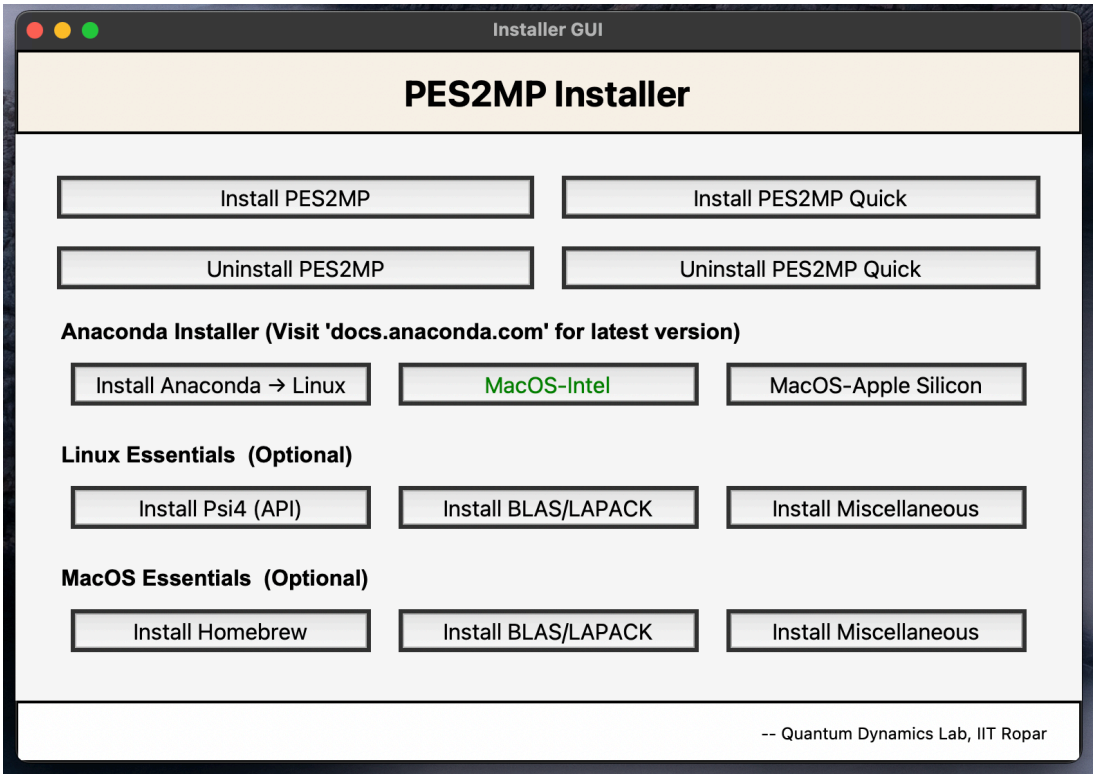


Figure 3.1: GUI Installer

- Select the 'Install Anaconda $->$' option to begin the installation process. Choose either Linux, MacOS (Intel), or MacOS (Apple Silicon) based on your hardware and OS. Once clicked, a new terminal will open and install Anaconda. Follow the on-screen instructions and select the appropriate options to finish the installation. Once installation is done, close the terminal.

  # The scripts for anaconda installs are present in *'/makefiles/anaconda_install'* folder. Users can also update the script for installing any newer version of Anaconda if the one present in the script is no longer maintained.

- Once Anaconda is installed, click on either the 'Install PES2MP' or 'Install PES2MP Quick' option. A new terminal will open and the scripts automatically create environments and install the required Python packages

needed for the PES2MP program. Follow on-screen commands, and once the installation is finished, close the terminal. *To learn more about the difference between the two installation options, see CUI installation.*

(OPTIONAL INSTALLS)

- Linux users can also install Psi4 (Standalone application for external calculations) directly for Ubuntu. For MacOS visit Psi4 website.
- The BLAS and LAPACK libraries (along with gfortran for Linux and gcc for MacOS) can be installed using the on-screen command. They are needed for installing the MOLSCAT program.
  * ***For MacOS users only*** *:: Homebrew is an essential package manager for installing any library in MacOS. Therefore, MacOS users must install Homebrew before installing BLAS/LAPACK or Miscellaneous!*
- The miscellaneous packages contain several well-known packages like htop, cmake, etc. Users can open the *'/makefiles/macos(linux)_essentials'* folder to check the packages in *install_etc.sh* file.

Once PES2MP is installed, close the GUI installer. Type the script for opening the PES2MP GUI program (at PES2MP home directory, i.e., *PES2MP-main/* folder).

```
$ python3 pes2mp_gui.py
```

This will open the following window:

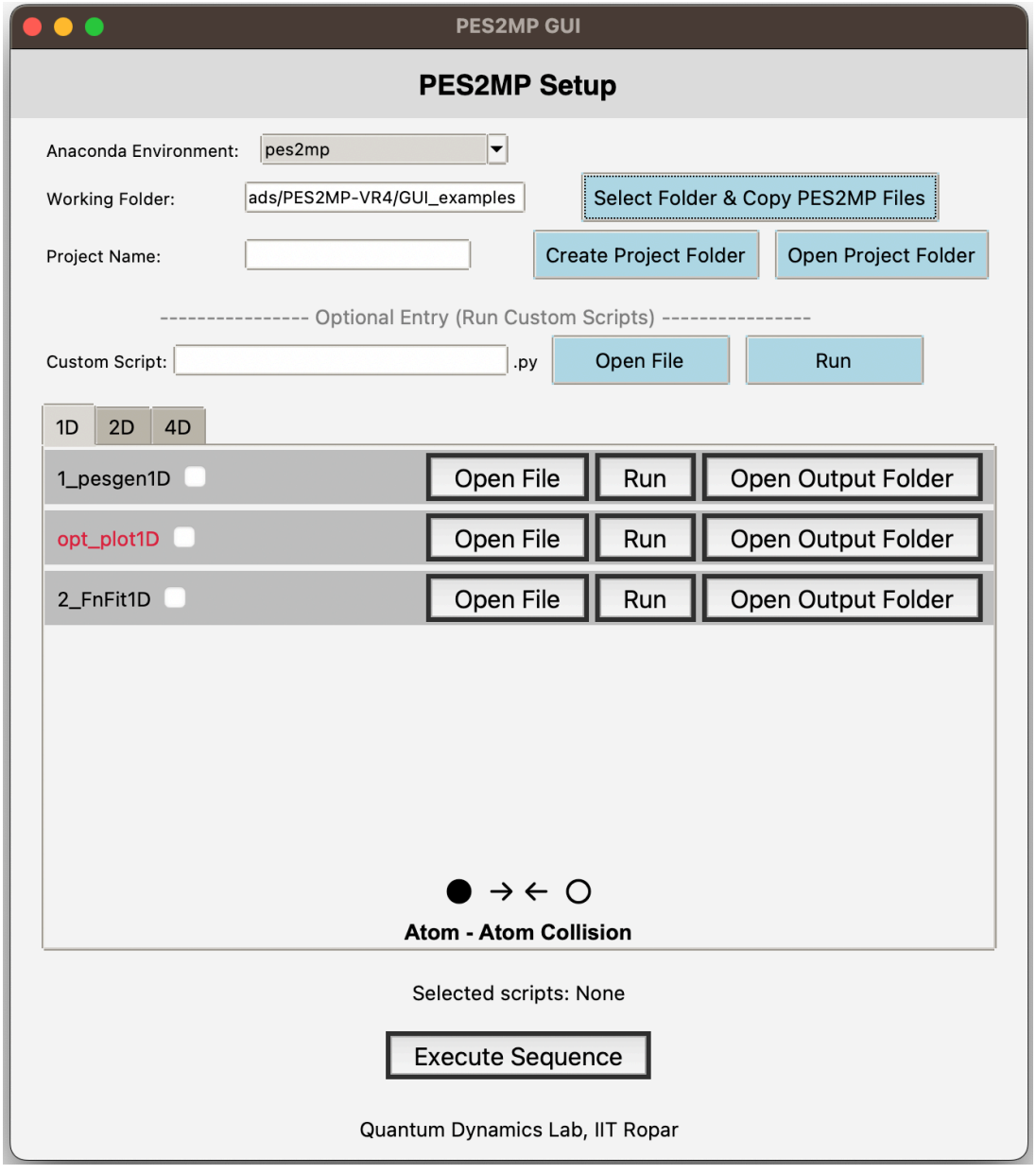
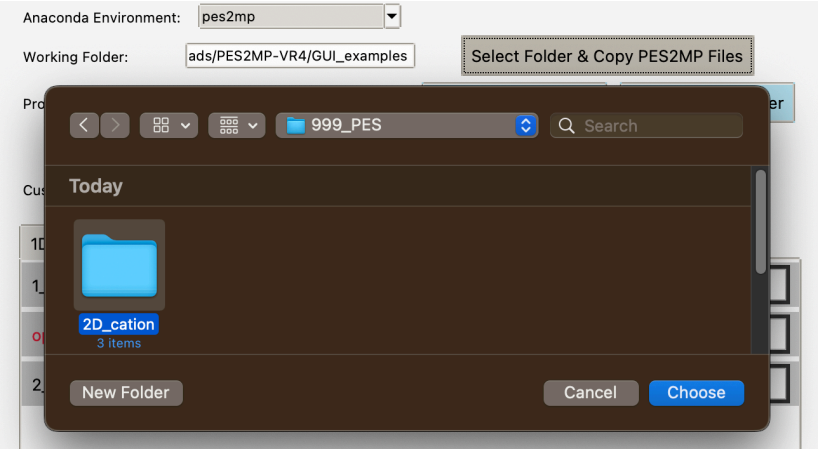


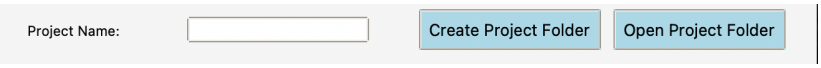
Figure 3.2: GUI Program

- Select the Anaconda Environment $->$ pes2mp or pes2mp_q (whichever version was installed previously).

- Select the folder where input files for each module (1_PESGen1D.py, 2_FnFit1D.py, etc.) are located. * *The default directory is /GUI_examples/.*



- The GUI interface automatically copies the *pes2mp.py* and *pes2mp_driver.py* files in the selected folder. If files already exist, the user is asked if they want to overwrite the files.
- Enter a Project Name such as '*testrun1*'. The separate project name ensures that output files from different runs are not overwritten. The output files are located inside '*/$input_file_Folder/Projects/$Project_Name/*'. The Project folder can be created and opened using the below commands:

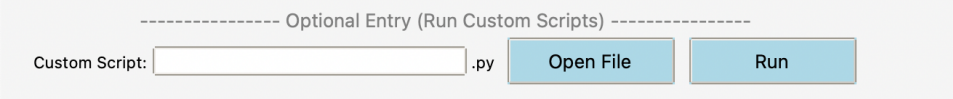


- Users can open the input file by clicking on '*Open File*' to make changes in the system or associated parameters. Use *'run'* command to run an input file. To see output files, users can click on '*Open Output Folder*'

———————————————————- How to use CUI input files with GUI ———————————————————-

The input file names are tied to the GUI program, i.e., they cannot be changed (as the GUI uses these names to distinguish between different input files). The example files for each 1D, 2D, and 4D collision with their numbered (1_PESGen1D.py) and optional files (opt_plot2D.py) are provided inside */GUI_examples/* folder. To run custom scripts with non-standard names, use the 'Optional Entry' option explained below:

- If there are non-standard scripts, i.e., Python input files not shown in 1D/2D/4D Tabs (e.g., different scripts for creating PES files), they can be run using the optional entry shown below:



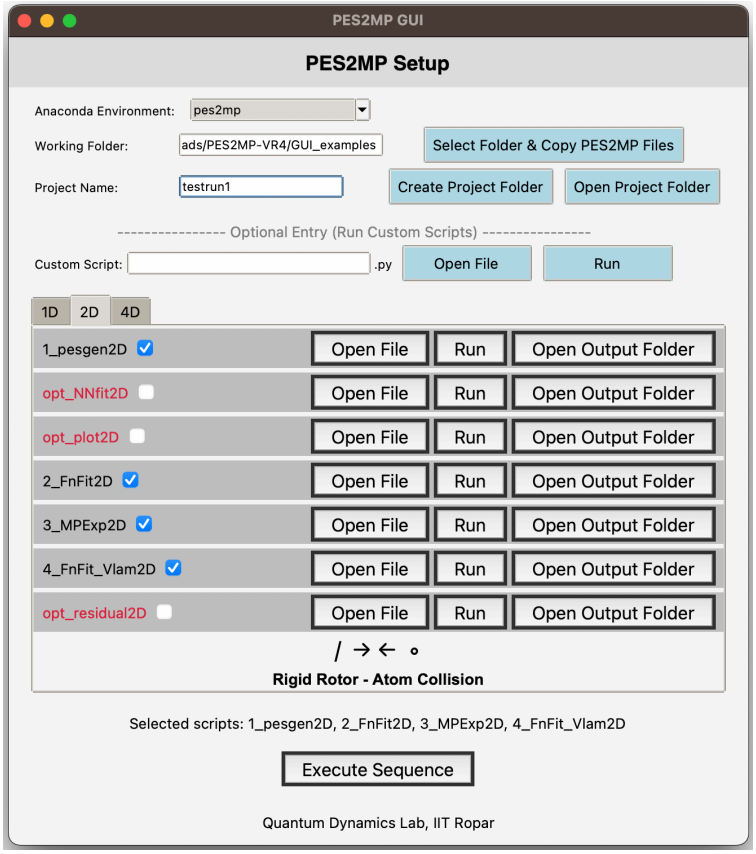
- Also, running the program using CUI takes the project name from the input file, whereas the GUI on the other hand, takes the project name in its interface. Therefore, users who want to use input files provided inside the '*/input_files/*' folder with the GUI must keep in mind the following changes:

1. Users can either rename the required input files or run the same using the custom script option.
2. The only change inside the GUI input file is the project name:

```
# CUI Project name CUI simply takes the project name in the input file
Proj_name  =  'testrun1_C2_He'  # This line has to be replaced!):
#-------------------------------------------------------------------------------------------#
# Replace the above code with these two lines (Project name is read from the GUI interface):
import os
Proj_name  =  os.getenv("Proj_name", "default_project_name")
#-------------------------------------------------------------------------------------------#
```

——

- Users can run the individual module using the 'Run' command or by selecting the tick option and running multiple modules in sequence. The sequence is shown above the 'Execute Sequence' button. If nothing is selected, the 'Selected script:' shows 'None'.



- Clicking on different tabs selects different collision types, e.g., 2D and 4D collisions. The files for different collisions (1D/2D) can be placed in the same folder, as different dimension files use different extensions, i.e., 1D, 2D, and 4D. However, the program only allows one collision type to run in sequence to prevent confusion.

Additional Information:

- The GUI program can run the individual modules of PES2MP like PESGen, FnFit, etc., either individually or in sequence as desired.
- If a module in sequence run fails (due to erroneous input), the previous module outputs are preserved. Therefore, the program can be restarted from the failed module in the next run.
- Since the input files are written in Python, users can use its format (e.g. # for comment) and packages (like os, numpy, etc.) as needed.
- The Psi4 package used internally is installed with PES2MP (no separate installation is needed).
- The GUI generates a log file inside */Projects/Proj_name* folder for each run (the log file has the same name as the Proj_name).

∗ For more information on input/output files, refer to the programmer's manual.

Section 4

GUI Examples

The following calculations (with input files) are provided in the GUI_examples folder. The same is provided for users to run the scripts and get familiar with the input files of PES2MP. Here, we shall briefly discuss the importance of each case. For full details (on their output), see the programmer's manual.

1. 0_ExtPES: Generating 1D, 2D, and 4D PES files for external calculations (on a server).
2. 1_NoFitPES: Generating 1D, 2D, and 4D PES files internally using Psi4:
   (a) 1D: Input files for handling (i) an anion, (ii) a cation, and (iii) a non-singlet (UHF) species. It also has examples for generating (iv) BSSE corrected and (v) CBS extrapolated PES.
   (b) 2D: Example for generating a (i) rough PES of C_2-He collision, (ii) expansion into radial terms, and (iii) fitting into mathematical expression (also generating MOLSCAT's potential (&POTL) file).
   (c) 4D: Example for generating a (i) rough PES of C_2-H_2 collision, (ii) expansion into radial terms, and (iii) fitting into mathematical expression (also generating MOLSCAT's &POTL file).
3. 2_FnFitPES: Generating 1D, 2D, and 4D PES files internally using Psi4 and fitting them into a mathematical expression (series of Slater functions) to interpolate or extrapolate missing data points. This example also has results for MOLSCAT output with rate coefficients.
   *The Slater ($\alpha e^{-\beta R}$) functions used for fitting PES and radial terms have the same β values (user input):
   (a) 1D: Input files for handling (i) C-He collision, (ii) re-plotting PES with different ranges, texts, plot parameters, or file name extensions, and (iii) fitting the PES into four simple mathematical functions.
   (b) 2D: Example for generating (i) a rough and a high-level PES of C_2-He collision, (ii) fitting them into a mathematical expression, (iii) expansion into radial terms, and (iv) using the same mathematical expression to fit radial terms (also generating MOLSCAT's &POTL file).
   (c) 4D: Example for generating a (i) rough PES of C_2-H_2 collision, (ii) fitting the PES into a mathematical expression, (iii) expansion into radial terms, and (iv) using the same mathematical expression to fit radial terms (also generating MOLSCAT's &POTL file).
4. 3_NNFitPES: Generating sparse (angular terms) 2D and 4D PES files internally using Psi4 and then fitting into a PES-specific ensemble NN model.
   *The NN model has custom activation functions (negative GELU and Gaussian), and a constraint on R: $ae^{-b(R-c)}/x$ to ensure desirable physical behavior at very-short and long-range regions when extrapolating:
   (a) 2D: Example for generating (i) a sparse PES for C_2-He collision, (ii) fitting into the NN model, (iii) fitting NN augmented PES into analytical expression and expansion into radial terms, and (iv) using the same mathematical expression to fit radial terms (also generating MOLSCAT's &POTL file).
   (b) 4D: Example for generating (i) a sparse PES for C_2-H_2 collision, (ii) fitting into the NN model, (iii) fitting NN augmented PES into analytical expression and expansion into radial terms, and (iv) using the same mathematical expression to fit radial terms (generating MOLSCAT's &POTL file).

Usually, fitting the PES into a function can have several advantages, such as saving disk space (coefficients along with the code take much less space than a full PES), and application in quantum dynamics calculations. The time and effort required to find a suitable function that fits a PES can improve the dynamical results of underlying

rotational and vibrational states. Therefore, we shall now see a few test cases to understand how a suitable PES can be generated/fitted for our application. In the previous examples, the users saw how a rough PES can save them a lot of time before starting a high-level calculation. So, we shall now proceed to show a few test cases considering the same with 1D files:

4.1 999_PES cases:

Here we shall see why generating a physically correct PES can be challenging and how to generate a PES from scratch using Psi4 with PES2MP that can be used for further applications.

4.1.1 Fitting 1D PES into mathematical expression $f(R)$

There are three cases: 1D_anion, 1D_cation, and 1D_neutral. In each case, users can see three different folders inside $/Projects/$, which are (a) *rough*, (b) *cbs*, and (c) *fit*.

- In *rough/* folder, a rough PES is calculated using HF-D4/cc-pvdz.
- In *cbs/* folder, PES is generated using a CBS method inbuilt into Psi4: 'sherrill_gold_standard' (SGS).
- In *fit/* folder, the *ab initio* PES file (in cm^{-1}) is copied from the *cbs/* folder. Then the PES is fitted into (a) an analytical expression that fits the full range (custom function) and (b) custom function that fits minima region along with two other functions that also fit very-short range ($\alpha e^{-\beta R}$) and long-range ($\gamma R^{-\kappa}$) regions for correct physical behavior.

Fig. 4.1 shows the *ab initio* PES, the function-fitted (also referred to as custom function fitted in the manual) PES, and function-fitted PES along with constraints on high-energy (HE) and long-range (LR) regions (HELR PES).

The function (custom-function) used to fit the full range of PES (for anion, cation, and neutral collision) is:

$$a_1e^{-5x} + a_2e^{-3.75x} + a_3e^{-3x} + a_4e^{-2.75x} + a_5e^{-2x} + a_6e^{-1.75x} + a_7e^{-1x} + a_8e^{-0.75x} \tag{4.1}$$

This mathematical expression (Eq. 4.1) is provided in the input file as:

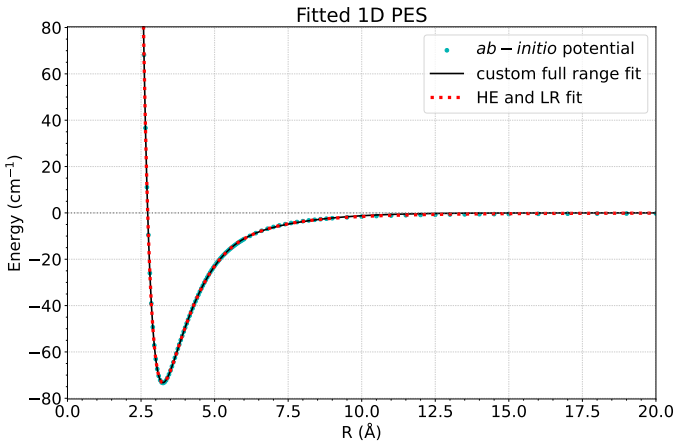
```
cutoff = 300        # Energy cutoff in cm-1. PES features are preserved till the cutoff

def fnfit_custom(x, a1,a2,a3,a4,a5,a6,a7,a8):
    import numpy as np
    return  a1*np.exp(-5*x)+a2*np.exp(-3.75*x)+a3*np.exp(-3*x)+a4*np.exp(-2.75*x) + \
            a5*np.exp(-2*x)+a6*np.exp(-1.75*x)+a7*np.exp(-1*x)+a8*np.exp(-0.75*x)
initial_val  = [1e4]*8                  # Enter initial guess
```

where the mathematical expression is passed as a '*fnfit_custom*' function and its cutoff (trims very high energy points) and initial guess are passed separately. (*An educated initial guess ensures convergence!)

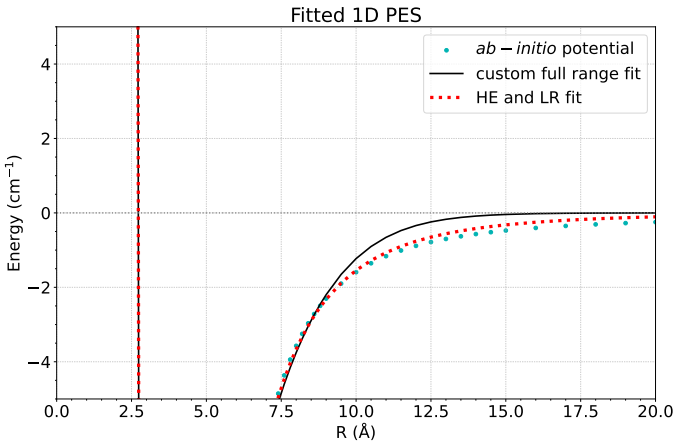
As we can see in Fig. 4.1, the function provides a good overall fit but may deviate slightly when zoomed in to the spectroscopic scale. For this, users can use HELR fit as shown below.

Apart from fitting the full PES range using Slater functions, the high-energy (HE) and long-range (LR) regions can be further constrained to follow a certain shape. For this, physically correct functions are used for a certain range of *ab initio* data. The fitted outputs of HE-LR regions are then merged with the previously fitted data (in the minima region, as there may be missing data points) to give the final PES. The example functions for each HE and LR are provided below.

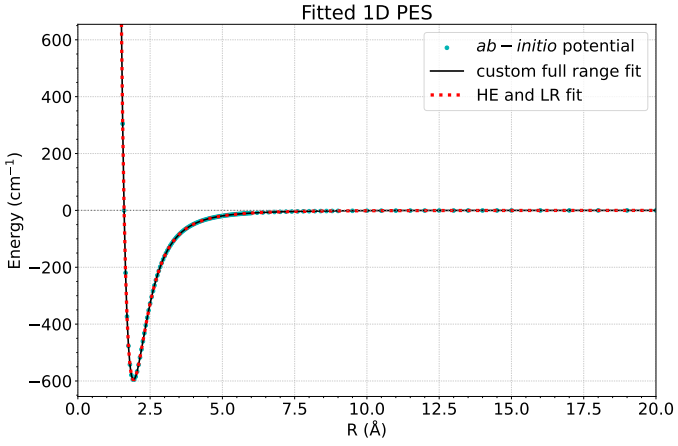
```
#-------------------------------------------------------------------------#
he_fit = True           # To fit high Energy region (Set True)
he_cutoff = 10000       # end value: fit data points till he_cutoff (cm-1)
he_min_offset = 4       # start value: fit from 4 positions before minima

def fnfit_he(x, he1,he2):             # Function for fitting high energy Region
    import numpy as np
    return  he1*np.exp(-he2*x)
he_initial_val  =  [1e4,1]            # Enter initial guess
#-------------------------------------------------------------------------#
```



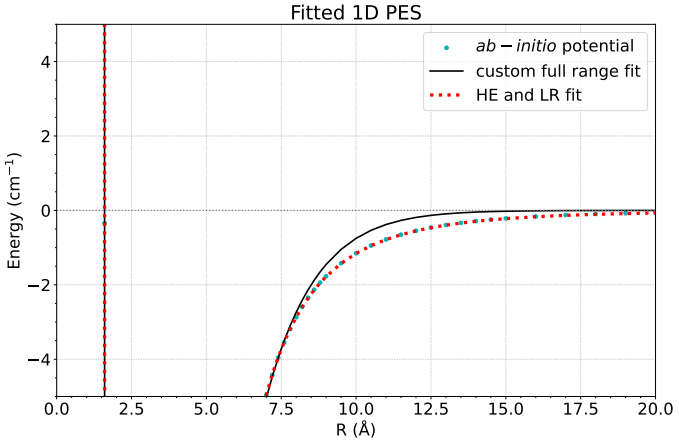
(a) Anion (F^--He)



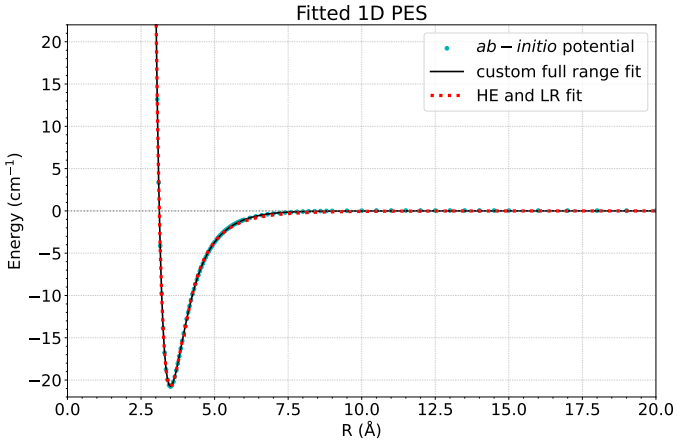
(b) Zoom (F^--He)



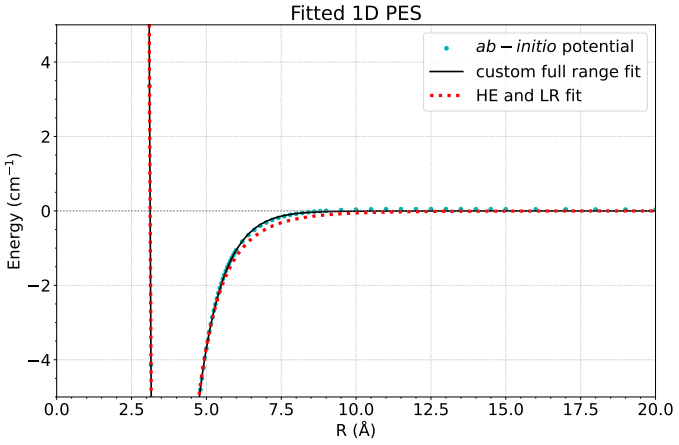
(c) Cation (Li^+-He)



(d) Zoom (Li^+-He)



(e) Neutral (Ar-He)



(f) Zoom (Ar-He)

Figure 4.1: Fitting results of fitting a 1D collisional PES both in full range and zoomed in. The blue dots are *ab initio* data points (SGS), the black line represents the custom function (Eq. 4.1) that fits the full range of the PES till cutoff, and the red dotted line represents the high-energy and long-range fitted PES (HE-LR) where the minima region is fitted using the custom function (Eq. 4.1) while the HE and LR regions are fitted using Slater ($\alpha e^{\beta R}$) and inverse power ($\alpha R^{-\beta}$) functions, respectively.

```
#-------------------------------------------------------------------------------#
lr_fit = True           # To fit Long Range Region (Set True)
lr_min_offset = 40      # fit from 80 positions before end value

def fnfit_lr(x, lr1,lr2):              # Function for fitting Long Range Region
    import numpy as np
    return  lr1*np.power(x,-lr2)
lr_initial_val  =  [1e4,4]             # Enter initial guess
#-------------------------------------------------------------------------------#
```

The high-energy region is approximated using a single Slater function where both coefficients are optimized.

Since we have two cases, a charge–quadrupole interaction (follows R^{-4} function) and London dispersion (van der Waals' follows R^{-6} function) at long range, we can either set them to exact values as shown below:

```
def fnfit_lr(x, lr1):              # Function for fitting Long Range Region
    import numpy as np
    return  lr1*np.power(x,-4)
lr_initial_val  =  [1e4]             # Enter initial guess
```

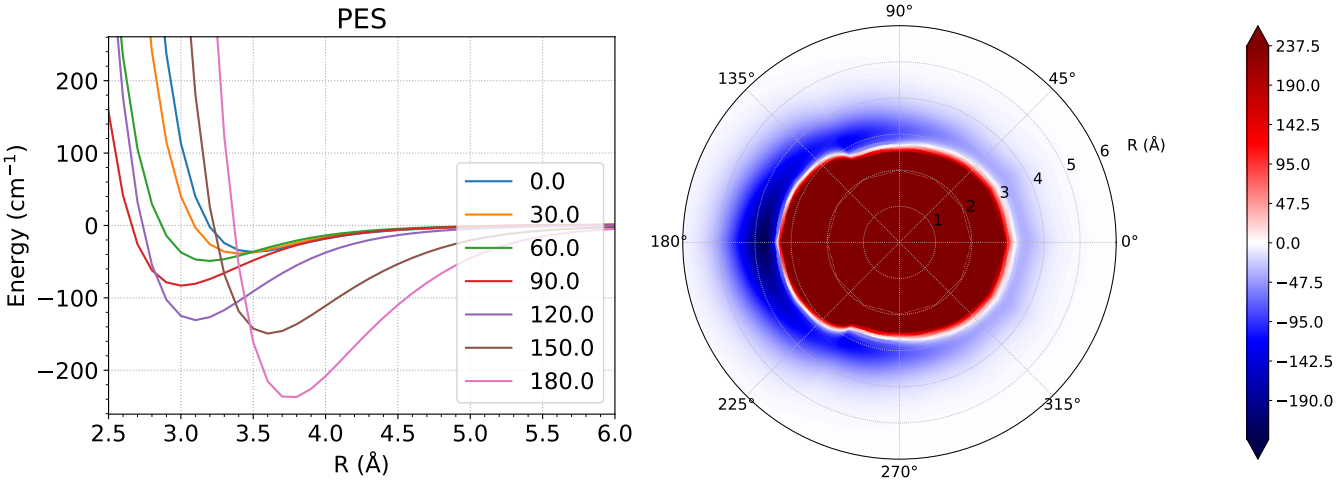
or allow them to relax with 4/6 as the initial guess (as in the previous case). The previous case is important if several long-range interactions average out (happens for H_2 collisions) to give an effective decay function.

The high energy cutoff (he_cutoff value) can be used to set an upper limit for HE fit (in cm^{-1}). The program will fit the PES from the offset, i.e. N^{th} point before minima (for each angle in case of 2D/4D) till the he_cutoff value.

For long-range, only the offset value is used, which indicates that fitting will begin from the N^{th} value before the final (R) asymptotic data point (for each angle in case of 2D/4D). For example, if the offset is 40 and there are 140 *ab initio* data points in an angle, the fitting will start from (140-40=) 100^{th} data point till the last, i.e., 140^{th} point.

4.1.2 Fitting non-symmetric 2D PES (HCO^+-He) into mathematical expression $f(R)$

Before proceeding with full PES calculation, a rough PES is generated in the intended range of R, θ and plotted to see if any additional data points will be needed.

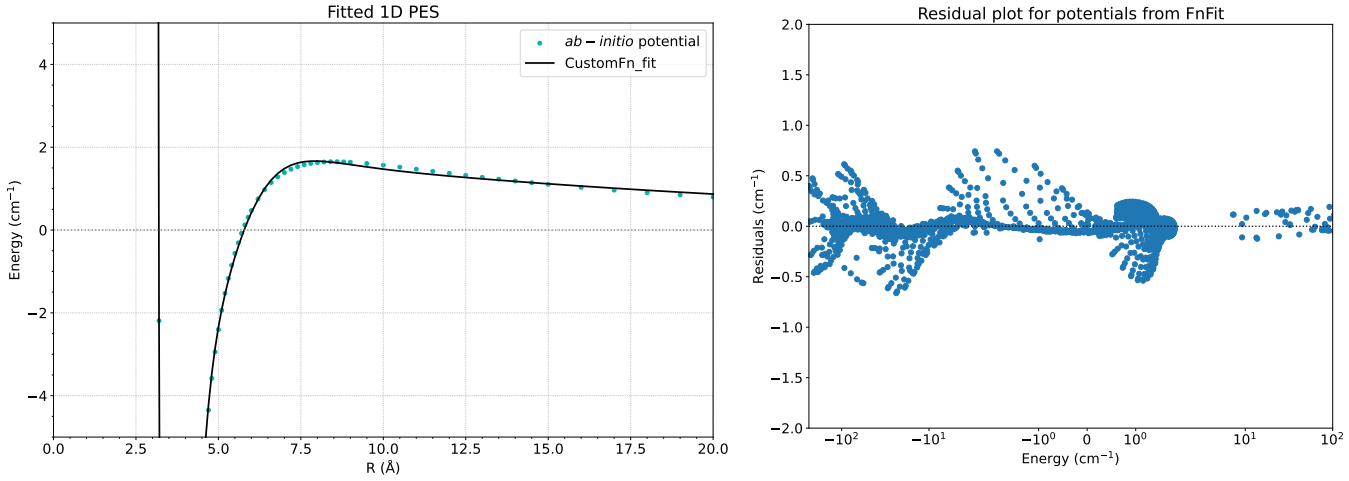


(a) (b)

Figure 4.2: Rough PES (a) *R vs E* plot at various angles θ (°) and (b) polar plot for HCO^+-He collision calculated using Psi4 (HF-D4/cc-pvdz).

As seen in Fig. 4.2, the global minimum is at 180°. While the range is good for 0° and 180°, the potential at 90° and 2.5 Å is very close to 0 cm^{-1} which registers a maximum of $\backsim$ 150 cm^{-1}. Therefore, in the high-level *ab initio* PES, the range must be changed from 2.5 Å to 2.0 Å.

In the above case, there is also a small repulsive region before He reaches the minima, as shown in Fig. 4.3(a) due to electrostatic repulsion. This can cause problems in long-range fitting. Since the *ab initio* range spans from 2.0 Å to 100Å, the long-range region of PES does not need extrapolation, and using an incorrect long-range fit with R^{-x}



(a) (b)

Figure 4.3: (a) PES at $\theta = 0^\circ$ showing repulsive behavior for HCO^+-He collision. (b) The residual plot (excluding very high energy regions) shows the fitting error (into Slater functions) for the PES at various energy regions.

can change the shape of this region. Therefore, to preserve this feature, only high-energy fitting is done, keeping LR fitting 'False'.

Why is high-energy fitting needed? Due to a large number of Slater functions in the fitting function, the custom-fitted PES may go to $-\infty$ at $R \to 0$, which is corrected by HE fitting. Both custom-fitted and HE-fitted PES (*Fnfit_PES.dat, HELRfit_PES.dat*) are located inside *Projects/Proj_name/PESFnFit/* alongside the original PES and Residual data (*Original_PES_sort_cm.dat, residuals_Fnfit_PES.dat*).

The custom function is similar to one in Eq. 4.1 where the last β coefficient is changed from -0.75 to -0.05 to capture the slow decaying tail of this repulsive PES. The PES is then corrected for the high-energy regions, and residuals are calculated to obtain the fitting error. The plot shown in Fig. 4.3(b) can be obtained from the *Projects/Proj_name/PESFnFit/Plots* folder.

∗***There is no separate residual plot for custom and HE-fitted PES?*** Since HE-Fit uses the custom-fitted data after the cutoff, the residual plot in the minima region remains the same for both. If needed, users can modify the existing code (refer to the Programmer's manual) and plot the same.

4.2 Conclusion!

Overall, the users have several options to generate and clean the PES. If a PES has no missing data points, users can directly proceed with MP expansion and fit it into a suitable function. Otherwise, users can fit the PES into functions of *R* and get the required missing data. Here, they can choose to either use: (a) a custom-fitted function, or (b) a custom-fitted function with special constraints on high-energy (HE) and long-range (R) regions. Finally, for heavier molecules, users may opt for NN augmentation (of angles and *R*) using a sparse *ab initio* PES and then proceed as required.

∗ For more information on input/output files of the GUI run, refer to the programmer's manual.

Section 5

Capabilities

The input files for each are provided with the program.

1. **PES Calculation :: 1D/2D/4D rigid rotor PES.**
   Simple templates (input_files.py) are provided for Psi4, Gaussian, and Molpro, which automatically create required input files (for each individual coordinate in XYZ format) for respective *ab initio* software packages. Sample input files are provided for single-point, BSSE-corrected, and CBS-extrapolation schemes for Molpro. Gaussian template defaults to BSSE corrected PES, while Psi4 can be used both internally (uses API interface in PES2MP) or externally (recommended for large calculations).

2. **Plots :: Plot PES, residual and curve fits.**
   PES plots are available as 1D (R vs E) and polar (R,θ vs E). For 2D/4D PES, certain angle combinations can be chosen for plots to prevent cluttering. Residual (error) plots can be automatically generated for NN/radial-terms generated PES. The curve fit with various functions can also be visualized along with any extrapolation function (if used). Apart from these, many other relevant plots are also provided for the NN module.

3. **NN augmentation :: Create NN model to augment and/or get missing data points.**
   The program can be utilized to separately fit minima and high-energy regions for a good fit. The underlying package for the NN model is TensorFlow, which can learn multiple outputs simultaneously and is not restricted by dimensionality (i.e., data can have N inputs and M output columns). The program also keeps boundary elements (of input features) in the training dataset to prevent boundary errors. There are two models: PES-specific (default) and Generic. The PES-specific model uses Gaussian activation functions (custom-made) and a modified Slater function for high-energy and long-range extrapolation. In the generic model, GELU is the default activation function. For finding the most suitable model (layers/nodes/activation function), Keras-tuner's (package) Bayesian Optimization tuning is used, which utilizes the Gaussian process to find optimum parameters within a reasonably defined search space.

4. **Multipole expansion :: Calculate radial terms (v_Λ).**
   The radial terms (v_Λ) are obtained by expanding the interaction potential (2D/4D PES) in a set of orthogonal functions (Legendre [2D] and two Spherical Harmonics (SH) [4D]) replacing angular coordinates ([θ] and [ϕ, θ_2, θ_1]). The current implementation takes PES (must not have any coordinate missing) and multiples it by the pseudo-inverse matrix containing Legendre/SH coefficients to get radial terms. **Warning**: The number of angles must not be less than λ as it will result in a poor fit, which can be visualized by recreating the PES.

5. **Curve fit :: Fit 1D PES and/or radial terms into analytical expression.**
   The radial terms (v_Λ) can be fitted into a series of power/exponential functions provided with the PES2MP package. There is also a Python script for generating functions that can be directly used in the input file for a more complex fit. Extrapolation schemes (R^{-6}, R^{-4}, R^{-3}, R^{-x}, etc.) are also available with plots (1D). After fitting the PES into MOLSCAT readable functions, the PES can be recreated (with a residual plot) to verify whether a good fit is obtained or not.

6. **MOLSCAT readable output :: Available for general-purpose version of POTENL subroutine**
   The curve fit produces an output that is MOLSCAT readable for direct utilization in **&potl** block. The feature is available in a general-purpose version of the subroutine POTENL of MOLSCAT 2020. The same can be utilized to calculate bound states or cross-sections for rotational (de-)excitation of two colliding species at cold and ultracold temperatures.