Hands-On Session

Preparation and Execution of Surface Hopping Trajectories with SHARC and Linear Vibronic Coupling Models for Sulfur Dioxide

1 What this exercise is about

- Obtaining a parameter set for an LVC model of SO_2 using TDDFT or CASSCF
- Using the LVC model to run a swarm of surface hopping trajectories
- Analyzing and interpreting the results and comparing to literature

2 Introduction

Sulfur dioxide (SO_2) is an important atmospheric pollutant. It is emitted by volcanoes and the industry, accumulates in the air, and participates in the occurrence of acid rain and cloud formation. The reason for this are some photoreactions that SO_2 undergoes after excitation with UV light from the sun. This makes the photodynamics of SO_2 at energies of about 4.0 eV very interesting. In fact, these dynamics have been studied since the 1930s, but the absorption spectrum and ultrafast dynamics have not been understood until the 2010s [1, 2, 3]. One of the reasons for this is the presence of 5 excited states at these energies — 2 singlet states ($^1B_1$, $^1A_2$) and 3 triplet states ($^3B_1$, $^3A_2$, $^3B_2$). A 1D cut through the 3D PESs of these states is shown in Figure 1, showing that there are several crossings between the singlet and triplet states. Figure 2 additionally shows a 2D cut through the PESs, illustrating the different conical intersections that arise between the states. Due to these crossings and intersections, the dynamics of SO_2 is relatively complex for such a small molecule. The presence of these crossings in principle makes it possible that after excitation to a singlet state, part of the wave packet converts to a triplet, a process that is called intersystem crossing (ISC) and which is only possible due to the relativistic spin-orbit couplings. Until a few years ago, it was not clear whether ISC occurs in SO_2 at all, which triplet state is involved, and how fast the ISC happens. These questions were only recently answered by a collaboration of pump-probe spectroscopy [1], quantum dynamics simulations [2], and surface hopping computations [3].

In the present exercise, we will use surface hopping simulations to simulate the ultrafast ISC dynamics of SO_2. However, the simulations in Ref [3] were performed at a very high ab initio level of theory, so that each surface hopping trajectory took a computation time of about 140 hours. Hence, our surface hopping simulations will employ the linear vibronic coupling (LVC) model, which is a general way to define mathematical functions of PESs for multiple states. It is in a sense a force field for nonadiabatic dynamics.

The main goal of the simulations is to obtain the time-dependence of the excited-state populations. This information will allow us to specify which singlet and triplet states are participating in the dynamics and which do not. Additionally, we will investigate what kind of molecular motion is induced by the excitation (bond length changes, bending motion).

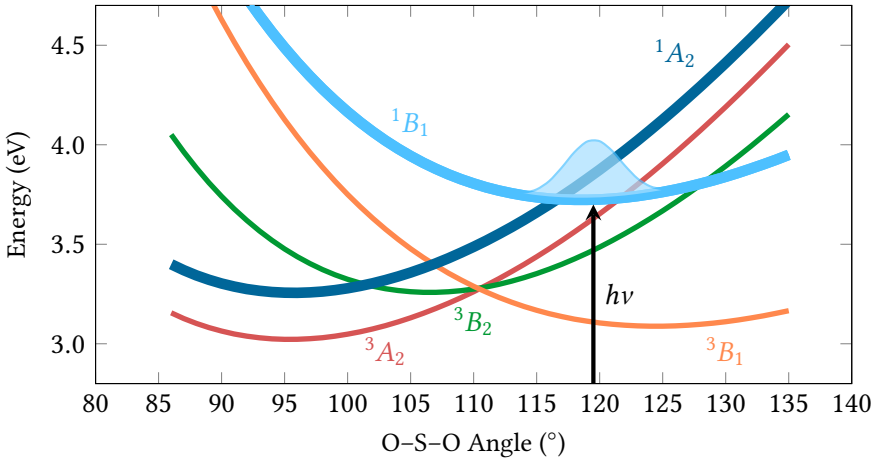

Figure 1: 1D PES scan along the bending angle of SO_2 for the 5 relevant **diabatic** states.

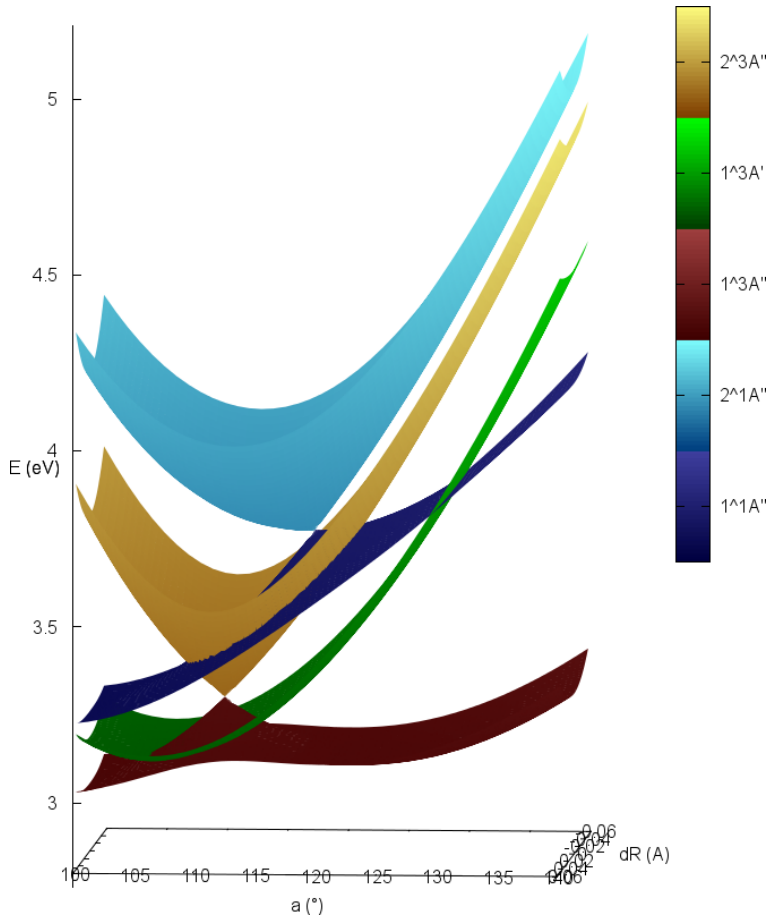

Figure 2: 2D PES scan along bending angle and asymmetric stretch of SO_2 for the 5 relevant **adiabatic** (but spin-diabatic) states, showing the two conical intersections between the two $^1A''$ states and between the two $^3A''$ states.

3 Background

This exercise consists of two steps. In the first step, we will parametrize an LVC model for SO_2 from a few DFT and TDDFT calculations. This includes:

1. Doing an optimization and frequency calculation with ORCA.
2. Extracting the computed normal mode information and converting to SHARC format, producing a file called `V0.txt`.
3. Running a set of TDDFT calculations with the SHARC–ORCA interface.
4. Extracting the excited-state results and converting to SHARC format, producing a file called `LVC.template`.

These steps can alternatively be carried out with SA-CASSCF and OpenMolcas. In any case, the result of these calculations are the files `V0.txt` and `LVC.template`. These fully specify the potential energy surfaces of our molecule, so that no more electronic structure calculation is needed once the files are available. From the frequency calculation, we also need to keep the Molden file with the normal modes.

In the second step, using the produced files, we will prepare and run a swarm of SHARC trajectories. This step is almost fully automatized in a provided Bash script, so that you only need to adjust the script slightly and then can directly carry out the trajectories.

You should prepare your shell with the following commands:

```
(base) user@node: ~> export SHARC=/projects/academic/cyberwksp21/Software/sharc2.1.1/sharc-master/bin/
(base) user@node: ~> export SCRADIR=/panfs/panfs.cbls.ccr.buffalo.edu/scratch/grp-cyberwksp21/$USER
(base) user@node: ~> module load orca-5-0-3
```

4 Parametrizing an LVC model

In this section, we prepare an LVC parameter set step by step. The description will be given for TDDFT with ORCA. If you want to use CASSCF with OpenMolcas, it is recommended to first do the TDDFT parametrization anyways and then adapt the steps accordingly.

4.1 Obtaining the `V0.txt` file

These calculations will be directly carried out with ORCA, without any SHARC tools. First, create an empty directory:

```
(base) user@node: ~> mkdir optfreq
(base) user@node: ~> cd optfreq
```

Create a file with the initial geometry for SO_2:

```
3

S 0. 0. 0.
O 0. -1.5 0.8
O 0. +1.5 0.8
```

Call the file `geom.xyz`.

Then create the input file for ORCA:

```
! opt freq D3 b3lyp def2-tzvp rijcosx
* xyzfile 0 1 geom.xyz

# do not forget a newline after the file name
```

Call the file `orca.inp`.

Now we have prepared the necessary input for ORCA. As yesterday, we need a run script for SLURM, which could look like this:

```
#!/bin/bash
module use /projects/academic/cyberwksp21/Modules
module load orca-5-0-3
export SCRADIR=/panfs/panfs.cbls.ccr.buffalo.edu/scratch/grp-cyberwksp21/$USER/ORCA.$$
cd $SLURM_SUBMIT_DIR
cp orca.inp geom.xyz $SCRADIR
cd $SCRADIR
orca orca.inp > $SLURM_SUBMIT_DIR/orca.log
```

Call the script `run.sh`. Then you can simply send it to the queue:

```
sbatch -A cyberwksp21 -p valhalla -q valhalla -c 1 run.sh
```

It should take about 3–4 minutes.

After completion, `****ORCA TERMINATED NORMALLY****` should be written at the end of the output file. If you scroll up a bit from there, you can find the output block with the `VIBRATIONAL FREQUENCIES`, the `NORMAL MODES`, and the `IR SPECTRUM`. This is the information that we are interested in for parametrizing the normal mode coordinates for the LVC model.

We can extract the normal mode information in two steps. First, we convert the ORCA standard output into Molden format, producing `orca.log.molden`. Second, we convert the Molden format into the `V0.txt` file format, which transforms the normal mode coordinates into mass-frequency weighted coordinates and adds the masses to the information in the file.

```
(base) user@node: ~> python2 $SHARC/ORCA_freq.py orca.log
(base) user@node: ~> python2 $SHARC/wigner.py -l orca.log.molden
```

The second command creates the `V0.txt` file.

4.2 Obtaining the `LVC.template` file

Leave the directory for the optimization and frequency calculation:

```
(base) user@node: ~> cd ..
```

We have to carry out the subsequent ORCA TDDFT calculations through the SHARC–ORCA interface, so that all relevant information is automatically extracted. The SHARC–ORCA interface requires an `ORCA.template` file which contains all the relevant quantum chemistry settings. This file has a simple keyword–argument structure and should look like this:

```
basis def2-tzvp
functional bp86
dispersion D3
charge 0 +1 0
ri rijcosx
keys tightscf
```

For more information, please see the SHARC manual.

With the `V0.txt` and `ORCA.template` files at hand, we can now setup the required TDDFT calculations. This setup is done with the interactive script `setup_LVCparam.py`, which works analogously to the other interactive scripts that you have already used.

Here is the required input:

```
  ================================================================================
||                                                                              ||
||                    LVC parametrization for SHARC dynamics                    ||
||                                                                              ||
||                     Author: Simon Kropf and Sebastian Mai                    ||
||                                                                              ||
||                                  Version:2.1                                 ||
||                                   01.09.19                                   ||
||                                                                              ||
  ================================================================================

This script automatizes the setup of excited-state calculations
in order to parametrize LVC models for SHARC dynamics.

------------------------V0.txt file------------------------

If you do not have an ground-state file, prepare one with 'wigner.py -l <molden-file>'!

Please enter the filename of the ground-state file.
Ground-state filename: [V0.txt] (autocomplete enabled) optfreq/V0.txt

File "optfreq/V0.txt" contains 3 atoms and we will use 3 frequencies/normal modes.
(others are zero)

----------------------Number of states----------------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: 3 0 3

Number of states: [3, 0, 3]
Total number of states: 12

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
1       MOLPRO (only CASSCF)
2       COLUMBUS (CASSCF, RASSCF and MRCISD), using SEWARD integrals
3       Analytical PESs
4       MOLCAS (CASSCF, CASPT2, MS-CASPT2)
```

```
5       ADF (DFT, TD-DFT)
6       TURBOMOLE (ricc2 with CC2 and ADC(2))
7       LVC Hamiltonian
8       GAUSSIAN (DFT, TD-DFT)
9       ORCA (DFT, TD-DFT, HF, CIS)
10      BAGEL (CASSCF, CASPT2, (X)MS-CASPT2)

Interface number: 9
The used interface will be: ORCA (DFT, TD-DFT, HF, CIS)

-----------------Spin-orbit couplings (SOCs)----------------

Do you want to compute spin-orbit couplings?

Spin-Orbit calculation? [True] <ENTER>
Will calculate spin-orbit matrix.

--------------------Analytical gradients--------------------

Do you want to use analytical gradients for kappa terms? [True] <ENTER>

Analytical gradients for kappas: True

----------Analytical nonadiabatic coupling vectors----------

Do you want to use analytical nonadiabatic coupling vectors for lambdas: False

------------------------Normal modes------------------------

Do you want to make LVC parameters for all normal modes? [True] <ENTER>

We will use the following normal modes: (7~9)

-----------------------Displacements-----------------------

Do you want to use other displacements than the default of [0.05]? [False] <ENTER>

Script will use displacement magnitudes of:

(7~9): 0.05

----------------------Intruder states----------------------

Intruder states can be detected by small overlap matrix elements.
Affected numerical kappa/lambda terms will be ignored and not written to the parameter file.

Do you want to check for intruder states? [True] <ENTER>

Ignore problematic states: False

------------------One-/Two-sided derivation-----------------

One-/Two-sided derivation of normal modes.
Choose for which normal modes you want to use one-sided derivation (7~9):
[None] (range comprehension enabled) <ENTER>

One-sided derivation will be used on: [None]
```

```
  ============================================================================
||                            ORCA Interface setup                            ||
  ============================================================================


------------------------Path to ORCA------------------------


Please specify path to ORCA directory (SHELL variables and ~ can be used,
will be expanded when interface is started).

Path to ORCA: [$ORCADIR] (autocomplete enabled) <ENTER>

----------------------Scratch directory---------------------

Please specify an appropriate scratch directory. This will be used to run the ORCA calculations.
The scratch directory will be deleted after the calculation. Remember that this script cannot check
whether the path is valid, since you may run the calculations on a different machine.
The path will not be expanded by this script.
Path to scratch directory: (autocomplete enabled) $SCRADIR/LVC/ORCA

------------------ORCA input template file------------------

Please specify the path to the ORCA.template file. This file must contain the following keywords:

basis <basis>
functional <type> <name>
charge <x> [ <x2> [ <x3> ...] ]

The ORCA interface will generate the appropriate ORCA input automatically.

Valid file "ORCA.template" detected.
Use this template file? [True] <ENTER>

------------------Initial restart: MO Guess-----------------

Please specify the path to an ORCA gbw file containing suitable starting MOs for the ORCA calculation.
Please note that this script cannot check whether the wavefunction file
and the Input template are consistent!

Do you have a restart file? [True] n
--------------------ORCA Ressource usage--------------------

Please specify the number of CPUs to be used by EACH calculation.

Number of CPUs: 1
Memory (MB): [1000] <ENTER>

----------------------WFoverlap setup----------------------

Path to wavefunction overlap executable: [$SHARC/wfoverlap.x] (autocomplete enabled) <ENTER>

State threshold for choosing determinants to include in the overlaps
For hybrids (and without TDA) one should consider that the eigenvector X may have a norm larger than 1
Threshold: [0.99] 0.9997

  ============================================================================
||                               Run mode setup                               ||
  ============================================================================
```

```
------------------------Run script------------------------

  This script can generate the run scripts for each initial condition in two modes:

    - In mode 1, the calculation is run in subdirectories of the current directory.

    - In mode 2, the input files are transferred to another directory (e.g. a local
      scratch directory), the calculation is run there, results are copied back and
      the temporary directory is deleted. Note that this temporary directory is not
      the same as the "scratchdir" employed by the interfaces.

  Note that in any case this script will create the input subdirectories in the
  current working directory.

  In case of mode 1, the calculations will be run in:
  '/projects/academic/cyberwksp21/Students/smai/sharc_tests/LVC-handson/BP86'

Use mode 1 (i.e., calculate here)? [True] <ENTER>

----------------------Submission script----------------------

During the setup, a script for running all initial conditions sequentially in batch
mode is generated. Additionally, a queue submission script can be generated for all
initial conditions.
Generate submission script? [False] <ENTER>


#########################Full input#########################

[...]

Do you want to setup the specified calculations? [True] <ENTER>

  ================================================================================
||                           Setting up directories...                            ||
  ================================================================================

Progress: [================================================] 100%
```

The setup script has automatically created run scripts for all the subdirectories. However, we need to add to these run scripts the loading of the ORCA module. This can be accomplished like this (write everything on one line):

```
user@node: ~> for i in DSPL*; do sed -i
              '2i module load orca-5-0-3'
              $i/run.sh; done
```

This loop will add the a line to each of the run scripts. Then the jobs are in principle prepared.

Because we require with the central geometry overlaps, we must finish `DSPL_000_eq` before launching any of the other jobs. Hence, run the first calculation:

```
user@node: ~> cd DSPL_000_eq
user@node: ~> sbatch -A cyberwksp21 -p valhalla -q valhalla -c 1 run.sh
```

Then wait the approximately 1 minute for completion. If successful, the directory should now contain a file called `QM.out`. Then go back and launch all other jobs (again, write the loop as one line):

```
user@node: ~> cd ..
user@node: ~> for i in DSPL*p DSPL*n; do cd $i;
              sbatch -A cyberwksp21 -p valhalla -q valhalla -c 1 run.sh
              cd -; done
```

These computations should take only a few minutes. You can check that everything worked by looking whether each of the seven `DSP_*` subdirectories contains a file called `QM.out`.

The last step is to extract the relevant information (vertical excitation energies, spin–orbit couplings, dipole moments, and gradients from `DSPL_000_eq`; vertical excitation energies and wave function overlaps from the other folders). This is automatically done by `create_LVCparam.py`:

```
(pysharc) user@node: ~> conda activate py27
(py27) user@node: ~> python $SHARC/create_LVCparam.py
```

Here, no further input is required, because all input from the setup script is saved in the files within `DSPL_RESULTS`. This last step produces the file `LVC.template`. It is good practice to document the employed level of theory by renaming the template file, e.g.:

```
(py27) user@node: ~> mv LVC.template LVC.D3-BP86_def2-tzvp.template
```

4.3 Parametrization with CASSCF and OpenMolcas

If you want to use OpenMolcas for parametrization, you also have to carry out the two steps. In the first step, create the geometry file, then call `$SHARC/molcas_input.py`. This is an interactive script that allows you to setup simple OpenMolcas input files. Setup an optimization and frequency calculation, using your desired basis set and active space. For SO_2, CAS(6,4) would be the smallest viable choice. Larger active spaces would be CAS(6,6) and CAS(12,9). You should also choose the number of states for state-averaging (recommended: 3 or 4 singlets, 3 triplets). In all cases, you should perform a single point calculation at the initial geometry first to check that the active orbitals are correct. Then you can use these orbitals as guess for the optimization.

Once the optimization and frequency calculation is finished, OpenMolcas should produce a file called `MOLCAS.freq.molden`. This can be directly converted with `wigner.py -l` to yield the `V0.txt` file.

For the second step, you need to prepare a `MOLCAS.template` file. If you run `$SHARC/molcas_input.py` again in the same folder as the optimization, it should enable you to easily generate the desired template file. With the `V0.txt` and `MOLCAS.template` files, you can then setup the excitation calculations. Choose 3 or 4 singlets and 3 triplets. After setup, run the calculations, then extract the `LVC.template file`.

5 Running the surface hopping trajectories

As written above, the entire surface hopping simulation is controlled by the Bash script `do_sharc.sh`. It performs 5 sequential steps:

1. Sampling initial conditions from a Wigner distribution. This requires the Molden file with the results of the frequency calculation.
2. Setting up and running the vertical excitation calculations at all sampled geometries using the LVC model.
3. Extracting the vertical excitation calculations and choosing the initial active states based on an infinitely short laser pulse.
4. Setting up and running the surface hopping trajectories.
5. Extracting the relevant output information from the produced files.

After the script is finished, you will thus only have to deal with all ensemble-based analysis steps, like checking for errors, statistical analyses, and the generation of the figures.

In the Bash script, there are only four variables that need to be changed to control the simulation:

- `frq_file`: This is the frequency calculation file that is needed for the initial condition generation. You need to provide a path to your file here. Alternatively, you can use the LVC parameter set from the original publication.[4]
- `LVC_file`: This file contains the LVC parameters. You need to provide a path to your file here. Alternatively, you can use the LVC parameter set from the original publication.[4] Note that the path to the `V0.txt` file is contained inside `LVC.template`. Hence, it does not need to be given separately, but you need to check whether the path inside the file is correct.
- `NINIT`: This is the number of initial conditions to be generated. Because almost all initial conditions will generate a trajectory (at least for the particular case here), this variable also controls the number of trajectories. Choose this variable after discussing with the instructor and the other participants.
- `states`: This is a string with the number of states per multiplicity. It must match the numbers of states given in the `LVC.template` file.

After you prepared the settings, you can execute `do_sharc.sh`:

```
(base) user@node: ~> sbatch -A cyberwksp21 -p valhalla -q valhalla -c 4 do_sharc.sh
```

Here, choose the number of CPU cores (option `-c`) after discussing with the instructor and the other participants.

After the trajectories and analyses are finished, the script will print how many initial conditions and trajectories were generated and how long they are (see inside the `slurm-*.out` file). It will also print the paths to your trajectories, which you can share with the other students in the course for better statistics in your analysis. Of course, you should only share trajectories if you used the same level of theory.

6 Analysis

In this section, you will analyze your results.

6.1 Analysis of Single Trajectories

Optionally, if you want to understand better how the surface hopping trajectories work, you can plot them. On the CCR cluster, this can be accomplished as follows. Change into one of the directories, generate a `gnuplot` plot script, and execute it:

```
(py27) user@node: ~> cd TRAJ/Singlet_1/TRAJ_00001
(py27) user@node: ~> python $SHARC/make_gnuscript 3 0 3 0 0 0 0 0 -png > plot.gp
(py27) user@node: ~> gnuplot plot.gp
```

The `-png` option will modify the script so that it does not produce interactive, zoomable plots, but rather pictures of the plots that you can download and inspect on your local computer.

6.2 Statistical Analysis

The Bash script will not carry out the statistical analysis of the trajectories. This analysis can be carried out manually with the following commands. For the statistical analysis, first go to the `TRAJ` directory and make a subdirectory for the results:

```
(py27) user@node: ~> cd TRAJ
(py27) user@node: ~> mkdir results
(py27) user@node: ~> cd results
```

Populations In order to compute the diabatic (or adiabatic) populations, you can use the interactive script `populations.py` from SHARC:

```
(py27) user@node: ~> $SHARC/populations.py
```

After you have started the script, it will ask you a number of questions:

- Path to trajectories: If you only analyze your trajectories, you can enter `../Singlet_1/`, then `../Singlet_2/`, etc., and then `end`. If you also want to include the results of other students, you have to add also the paths to their trajectories
  (e.g., `/ /projects/academic/cyberwksp21/Students/..../TRAJ/Singlet_1/`).
- Analyze mode: For diabatic populations, use mode `20`.
- Run data extractor: Say `no` (extractor was already run by the Bash script).
- Number of states: Simply press `ENTER`.
- Normalize: Simply press `ENTER`.
- Simulation time: Enter `700` (femtoseconds).
- Save data for bootstrapping: Simply press `ENTER`.
- Gnuplot script: Either say `yes` and enter a file name, or say `no`.
- Do you want to do the analysis? Simply press `ENTER`.

The script will create a file called `pop.out` that contains the populations. These can be plotted as usual with `gnuplot` (the second command is just a single line):

```
gnuplot> set term dumb
gnuplot> p "pop.out" u 1:3 w l, "" u 1:4 w l, "" u 1:($5+$8+$11) w l,
         "" u 1:($6+$9+$12) w l, "" u 1:($7+$10+$13) w l
```

The organization of the file with the diabatic populations is:

- Column 1: Time

- Column 2: S_0
- Column 3: $^1B_1$
- Column 4: $^1A_2$
- Columns 5, 8, 11: The three components of the triplet $^3B_1$
- Columns 6, 9, 12: The three components of the triplet $^3B_2$
- Columns 7, 10, 13: The three components of the triplet $^3A_2$

Instead of using the `dumb` terminal of `gnuplot` to get an ASCII art plot, you can also switch to the `pngcairo` terminal and save the plot as a picture:

```
gnuplot> set term pngcairo
gnuplot> set out "populations_diab.png"
gnuplot> p "pop.out" u 1:3 w l, "" u 1:4 w l, "" u 1:($5+$8+$11) w l,
         "" u 1:($6+$9+$12) w l, "" u 1:($7+$10+$13) w l
```

The `set term pngcairo` and `set out` lines can also be used to the plots mentioned below. Note that it is required to run a `p` or `replot` command after `set out`, otherwise the output file will be empty.

Optionally, you can also compute the adiabatic populations, just run `populations.py` in mode `8`. The resulting `pop.out` has these columns:

- Column 1: Time
- Column 2: S_0
- Column 3: S_1 (lower adiabatic singlet)
- Column 4: S_2 (upper adiabatic singlet)
- Columns 5, 8, 11: The three components of the triplet T_1
- Columns 6, 9, 12: The three components of the triplet T_2
- Columns 7, 10, 13: The three components of the triplet T_3

Geometric Parameters The bond lengths and angles were already calculated by the Bash script. This data can be collected with `$SHARC/data_collector.py`:

```
(py27) user@node: ~> $SHARC/data_collector.py
```

After you have started the script, it will ask you a number of questions:

- Path to trajectories: If you analyze only your own trajectories, you can enter `../Singlet_1/`, then `../Singlet_2/`, then `end`. If you also want to include the results of other students, you have to add also the paths to their trajectories
  (e.g., `/scratch/indiumX/baYY/Ex_BIG2/TRAJ/Singlet_1/`).
- File path: Select the `./Geo2.out` file.
- T column: `1`.
- X column: Enter `2` here. Later, run the data collector again and use `3`, then `4`.
- Y column: Simply press `ENTER`.
- Show workflow: Simply press `ENTER`.
- Smooth trajectories: Simply press `ENTER`.
- Synchronize: Simply press `ENTER`.
- Convolution in X direction: Enter `yes`.
- Kernels: Simply press `ENTER`.
- Width of function: Choose `1.0` for column 2, `0.02` for columns 3 and 4.
- Grid size: Choose `75` or slightly higher if you want.

- Xrange: Choose `1.0`.
- For all remaining questions, simply press ENTER (6 times).

For the other runs, you can simply change the X column and the width in `KEYSTROKES.data_collector` and run the script again:

```
(py27) user@node: ~> $SHARC/data_collector.py < KEYSTROKES.data_collector
```

The script will create three files for each run (9 in total), but only the `type3` files with file names `collected_data_1_X_0_sy_cX.type3.txt` (with `X` = 2, 3, or 4) are interesting here. They can be plotted with `gnuplot`:

```
gnuplot> set term pngcairo
gnuplot> set out "col2.png"
gnuplot> set view map
gnuplot> sp "collected_data_1_2_0_sy_cX.type3.txt" u 1:2:3 w pm3d
```

For all three of these plots, the x axis is time. For the file you obtain from column 2, the y axis is the bending angle of SO_2 in degrees. For column 3, the y axis is half the difference between the bond lengths $r_d = 0.5(r_{S-O_1} - r_{S-O_2})$ (in Ångstrom). For column 4, the y axis is the average of the bond lengths $r_a = 0.5(r_{S-O_1} + r_{S-O_2})$ (in Ångstrom). The z value (given by the color) is proportional to the number of trajectories. Note that these plots cannot be made with the `dumb` terminal.

6.3 Questions

In this exercise, you should prepare an LVC parameter set and then run around 200 trajectories. You may also share trajectories with each other for analysis.

Then, perform the statistical analysis for the diabatic populations. Try to use as many trajectories as possible for the populations analysis, also including the trajectories of the other students if possible. To do this, communicate with each other and share the paths to the trajectory folders. Note that you can access the scratch of another computer by exchanging `/public/arara/scratch/` with `/scratch/indiumX/` (this exchange has been already applied to the paths given by the Bash script). **You can share the paths to all trajectories with your co-students on Moodle in the Questions & Answer Forum. In this way, everyone can later repeat the statistical analysis with a large ensemble. Additionally, you can of course use the Zoom chat function, but this is not persistent.**

Also carry out the statistical analysis for the three geometric parameters. For this analysis, it is sufficient to analyze your own trajectories (optionally, you can include trajectories of other students, but this is not required).

Create plots of the diabatic populations as well as of the geometric parameters. Use these plots to answer the questions below.

Questions:

1. Plot the diabatic populations. Also show the evolution of the geometric parameters (bond angle, bond length difference, average bond length).
2. Describe and discuss the diabatic populations from the simulations. Which is the initial state? How is the dynamics within the two singlet states? Besides the singlet dynamics, are the triplet states involved? If yes, which triplets are involved?
3. Compare the diabatic populations to: Ref [2] Figure 2, Ref [3] Figure 6(b), and Ref [4] Figure 4(b). Do the singlet populations agree? Do the simulations agree on the total triplet yield and the relative importance of the triplets? Discuss which of these four simulations produces the most accurate results, and explain why this is the case.
4. Describe the motion of the molecule after excitation, based on the three plots of the geometric parameters. For each of the three degrees of freedom, what kind of motion is observed? **Bonus**: Compare and discuss differences with Ref [3] Figure 7 (middle row).

It is not necessary to read all of the papers [2, 3, 4]! You can focus on the abstracts and the discussions that are directly related to the mentioned figures. Note that the data for the population plots from Refs [2, 3, 4] are available in the exercise files.

References

[1] I. Wilkinson, A. E. Boguslavskiy, J. Mikosch, D. M. Villeneuve, H.-J. Wörner, M. Spanner, S. Patchkovskii, A. Stolow: Non-adiabatic and intersystem crossing dynamics in SO_2 I: Bound state relaxation studied by time-resolved photoelectron photoion coincidence spectroscopy, *J. Chem. Phys.*, **140**, 204301 (2014).

[2] C. Lévêque, R. Taïeb, H. Köppel: Communication: Theoretical prediction of the importance of the $^3B_2$ state in the dynamics of sulfur dioxide, *J. Chem. Phys.*, **140**, 091101 (2014).

[3] S. Mai, P. Marquetand, L. González: Non-adiabatic dynamics in SO_2: II. The role of triplet states studied by surface-hopping simulations, *J. Chem. Phys.*, **140**, 204302 (2014).

[4] F. Plasser, S. Goméz, S. Mai, L. González: Highly efficient surface hopping dynamics using a linear vibronic coupling model, *Phys. Chem. Chem. Phys.*, **21**, 57 (2019).