Numerical 2D-PES of Reactions with PTSB

Table of Content

Aim and Reference

- Using several programs/scripts to general a numerical two-dimensional potential energy surface (2D-PES) of reaction with post-transition-state bifurcation (PTSB).
- Supporting information of Construction of Two-Dimensional Potential Energy Surfaces of Reactions with Post-Transition-State Bifurcations.
- **Author** : Hsiao-Han (Grace) Chuang - *Initial work* - 2018 May.

Processes to Generate 2D-PESs

Step 1: Calculate Stationary Points and IRCs.

Optimize two transition-state structures (TSSs), and then run the corresponding IRCs. After that, extract the last points of IRCs as the initial structures for local minima optimization.

- Code:
  - `checkGau`, `runIRC`

1. From organic chemistry, guess possible initial structures of TSSs and then follow the user guide in standard electronic package (i.e., Gaussian 09/16 in this project) to write input files.
   - TSSs have one and only one imaginary frequency and the corresponding vibration mode need to fit the guessed reaction mechanisms.
   - This try-and-error step needs well-trained organic chemistry background and good chemical intuition.
2. Double check the consistency of atomic index and ordering for all molecules.
   - Check the ordering of atoms between TSS1 and TSS2 systems. Reorder them via Gaussview if they are different.
3. Gain enough grid points of an IRC path

    - Use `runIRC` to test different combinations of keywords in route section.
    - Check the direction of IRC paths.

4. Double check the orientation for all systems.

    - Use *gaussview* or *jmol* to check the molecular orientation.
    - Show axes is easier to check.

5. Use `checkGau` to check all important information from Gaussian output files.

```
[Grace@elderberry UMP2]$ checkGau all

Program /home/Grace/EST-tool/checkGau will check and alanyse G09 output files
working directory: /work/Grace/ModifyIRC/run/PostTS_H3CO/OPT/UMP2

The extension of output file is : .log

Std-out the information of rawdata

Count the amount of jobs:
  Total:             9
Success:             9
   Fail:             0

----------------------------------------
Fail jobs are listed in file Fail.txt
----------------------------------------
Classification of the successful jobs:
     SP:             0
    MIN:             3
 Saddle:             2
    IRC:             4

Detail of output:

----- MIN -----
      Name             E(hartree)            ZPE(hartree)
  P1_TS2_R        -114.7242344388            0.0387990000
  P2_TS2_F        -114.7242344388            0.0387990000
         R        -114.7099047044            0.0385250000

----- Saddle -----
      Name             E(hartree)            ZPE(hartree)        #IF       IF(cm^1)
       TS1        -114.6541601598            0.0340640000         1      -2272.2000
       TS2        -114.7171727917            0.0370120000         1       -526.9842

----- IRC -----
      Name       #IRC      rel_R(hartree)            TS(hartree)          rel_P(hartree)
     TS1_F         10       -0.0474200000        -114.6541600000           -0.0011000000
     TS1_R         10        0.0000000000        -114.6541600000           -0.0372900000
```

Step 2: Asymmetric Cases: Generate Artificial Reaction Coordinate

For asymmetric cases, build the artificial reaction coordinate in the TSS1 forward direction.

- Code:
    - `runArticIRC.sh`, `genArticStruc.f90`
- Input:
    - $(selectCoord).log, TSS2.log
- Output:
    - Artic1D.xyz, Artic1D_PEC.dat

1. Compare the energy difference between TSS1 and TSS2:
    1. TSS1 > TSS2
        - Go to the following steps.
    2. TSS1 < TSS2 and the energy difference is small
        - Beyond this project, this is not a pitchfork model potential.

2. Select a point from TSS1 IRC forward direction.
    - Near the shoulder of its energy profile.
    - Near the gradient which is close to zero, or has a turn over point.
3. Execute `runArticIRC.sh` to generate a serious of structures which use `genArticStruc.f90` to build artificial reaction coordinate.
    - After it fulfills the criteria (i.e. energy difference between the last point and TS2 is less than 0.0001 hartree; less than 1 kcal/mol), the program will stop automatically.
    - Use `jmol` to check the structures via file *Artic1D.xyz*.
    - Use `gnuplot` to plot energy profile via file *Artic1D_PEC.dat*.

Step 3: Construct X- and Y-Axes (Optional)

Combine fragmented files into three files, x.xyz, y_F.xyz and y_R.xyz.

- Code:
    - `getIRCcurve`, `getIRCstruc`, `rev1Dstruc`, `checkGau`, `getCoord`

Symmetric Cases

- Output:
    - x.xyz, y_F.xyz, y_R.xyz

1. Copy 4 IRC output files from /OPT to /1D as the input files for the following steps.
2. Generate x.xyz
    1. Use `getIRCcurve` to extract energy profiles, and then use *gnuplot* to plot the potential energy curves.
        - The sign of coordinate may need to be modified; use *awk*.

```
[Grace@elderberry SP_MP2OPT]$ getIRCcurve TS1_F.log TS1_F.dat
Plot IRC curve
$1 = IRC output
$2 = IRC result
formate : x = mass-weighted coord, y = hartree
[Grace@elderberry SP_MP2OPT]$ cat TS1_F.dat
-1.05225        -114.701580
-0.94702        -114.697030
-0.84178        -114.691970
-0.73654        -114.686490
-0.63130        -114.680680
-0.52608        -114.674670
-0.42086        -114.668700
-0.31565        -114.663100
-0.21044        -114.658400
-0.10524        -114.655260
0.00000 -114.654160
```

    2. Use `getIRCstruc` to extract all the structures.
        - In the IRC file, it starts from TS structure by default. So use `rev1Dstruc` to reverse the direction for reverse direction.

```
[Grace@elderberry SP_MP2OPT]$ getIRCstruc TS1_F.log TS1_F.xyz

--- Program /home/Grace/bin/getIRCstruc extracts all structures of
--- G09 file TS1_F.log along IRC result

Output data: TS1_F.xyz

Total amount of points (include TS) is :  11

[Grace@elderberry SP_MP2OPT]$ cat TS1_F.xyz
5
0.0000
6 -0.003943 0.006419 -0.002357
1 -0.001095 -0.013007 1.083358
1 0.953703 -0.013007 -0.513923
1 -0.928714 -0.583410 -0.555152
8 -1.074321 0.611560 -0.642191
5
0.10524
6 0.553871 -0.290682 0.331085
1 0.553897 -0.310864 1.415844
1 1.509340 -0.310864 -0.182515
1 -0.407895 -0.846592 -0.243825
8 -0.519837 0.310598 -0.310740
```

   3. Combine the forward and reverse direction to form a whole path as x.xyz.
      - x.xyz = R $\to$ TSS1 $\to$ TSS2.
3. Generate y_F.xyz and y_R.xyz.
   - y_F.xyz = TSS2 $\to$ P1.
   - y_R.xyz = TSS2 $\to$ P2.
4. Check the orientation via *jmol* and then open the axes function; sometimes the TS structure and IRC structures has shift, and cannot properly overlap to each other.

Asymmetric Cases

- Input:
  - TS1_F.xyz, Artic1D.xyz
- Output:
  - TS1_F_Artic1D.xyz, x.xyz, y_F.xyz and y_R.xyz

1. Generate x.xyz
   1. For TS1 forward direction, combine the TS1_F.xyz from reactant to the selected point and Artic1D.xyz. After that, name the new file as TS1_F_Artic1D.xyz .
   2. Use *jmol* to double check the orientation of TS1_F_Artic1D.xyz .
   3. Combine TS1_F_Artic1D.xyz and TS1_R.xyz as x.xyz
      - x.xyz = R $\to$ TSS1 $\to$ selected point $\to$ TSS2
2. Generate y_F.xyz and y_R.xyz: same process as the symmetric case.

Step 4: Select 1D Grid Points for All IRC Paths

To reduce computational cost, select enough grid points in one-dimensional energy profile instead of using full grids.

- Code: `run1Dgrid.sh`, `get1Dgrid.sh`
  - `run1Dgrid.sh`: `getIRCcurve`, `getIRCstruc`, `selectIRCstruc.sh`, `writeGauInpV`
  - `get1Dgrid.sh`: `checkGau`, `rev1Dstruc`, `rot.py`
- Input:
  - Symmetric cases (5 files):
    1. header.dat (route section)

        2. TS1_F/R.log
        3. TS2_F/R.log.
    - Asymmetric cases (7 files):
        1. header.dat
        2. TS1_F/R.log
        3. TS2_F/R.log
        4. $(coordinate of select point).log
        5. Artic1D.xyz
- Output:
    - Symmetric cases
        - $(NGrid)_TS1_F.xyz
    - Asymmetric cases
        - $(NGrid)_TS1_F_Artic1D.xyz
    - $(NGrid)_TS1_R.xyz
    - $(NGrid)_TS2_F.xyz
    - $(NGrid)_TS2_R.xyz
    - E_scan.*.dat

1. Execute script `run1Dgrid.sh` to generate selected 1D grid points.
    - If the file, Artic1D.xyz, exists, then it is an asymmetric case. Or, it is a symmetric case. Do not remove it durning the calculation.
    - The default amount of selected grid point is 30, and it can be modified in *line 212* of `run1Dgrid.sh`.

```
210  function main(){
211      # userIO Ngrid
212      Ngrid=30 # Default grid points
213      purpose $Ngrid
```

    - `selectIRCstruc.sh` selects points along the new reaction coordinate in order to reduce the computational cost.

```
[Grace@elderberry SP_MP2OPT]$ selectIRCstruc.sh x.xyz
Extract the first structure (TS) or not : (y/n) n
Key-in the amount of selected structure: (integer) 39
[Grace@elderberry SP_MP2OPT]$ head -n 10 39_x.xyz
5
-1.89429
6 0.499552 -0.348438 0.298614
1 0.548719 -0.177585 1.376319
1 1.472074 -0.177585 -0.168361
1 0.231540 -1.399846 0.138407
8 -0.516661 0.371986 -0.308842
5
-1.84168
6 0.499832 -0.344886 0.298782
```

    - Use gnuplot to double check the geometries and the selected geometries.
2. Execute `get1Dgrid` to collect data.
    - Plot the potential energy curves of selected grid points to see if the selected points are reasonable.
    - **Double check the geometries! Really important!** Use gnuplot to check the geometries of x.xyz; it should start from R to TSS2. Recall that,
        - x.xyz = R $\to$ TSS1 $\to$ selected point $\to$ TSS2.

Step 5: Scan a 2D-PES

- Code:
  - Generate numerical 2D-PES
    - `getIRCvec`, `Vsca1D_IRCvec.f90`, `writeGauInpV`, `qsubGau`
  - Constrain optimization
    - `getGrad.sh`, `genGradStruc.f90`, `genNewPES.sh`
  - Collect rawdata
    - `getPESwStruc.sh`, `checkGau`
  - Plot contour and 3D version figure
    - `plot2DPES.py`
- Input:
  - header.dat, x.xyz, y_F.xyz and y_R.xyz
- Output:
  - $(Pts)_E.dat, $(Pts)_Struc.xyz

1. Generate a numerical 2D-PES
   1. Create sub-directories and named as *Forward* and *Reverse*, which is the directions along y-axis.
   2. In /1D directory, rename $(NGrid)_TS2_F.xyz as y_F.xyz, and $(NGrid)_TS2_R.xyz as y_R.xyz. After that, copy x.xyz, y_F.xyz and y_R.xyz from /1D to /2DPES.
   3. Execute `getIRCvec` to calculate translational vectors from y_F.xyz and y_R.xyz.

```
[Grace@elderberry SP_MP2OPT]$ getIRCvec 20_y_F.xyz

--- Program /home/Grace/bin/getIRCvec calculate the displacement difference, vector,
--- between two structures, and then create a serious of vector
--- from file 20_y_F.xyz

Output data: 20_y_F.vec.txt
```

   4. Generate a serious of structures via script `Vscan1D_IRCvec`, which has an output file, *pes.txt*.

```
[Grace@elderberry Forward]$ Vscan1D_IRCvec 39_x.xyz 20_y_F.vec.txt

-------------------------------------------------------
Program Vscan1D_IRCvec produces a serious of structures
along a serious of given vectors.
-------------------------------------------------------

-------------------------------------------------------
Auxiliary bash shell scripti: getIRCstruc, getIRCvec

Output file : pes.txt
-------------------------------------------------------
```

   5. Generate gaussian input files via `writeGauInpV`.

```
[Grace@elderberry Forward]$ writeGauInpV pes.txt ../header.txt
Please key-in the charge: 0
Please key-in the multiplicity: 2
[Grace@elderberry Forward]$ ls
10_0.com   13_2.com   17_3.com   21_1.com   25_13.com  29_14.com  32_7.com   36_8.com   5_8.com
10_10.com  13_3.com   17_4.com   2_12.com   25_14.com  29_15.com  32_8.com   36_9.com   5_9.com
10_11.com  13_4.com   17_5.com   21_2.com   25_15.com  29_16.com  32_9.com   3_6.com    6_0.com
10_12.com  13_5.com   17_6.com   2_13.com   25_16.com  29_17.com  3_2.com    37_0.com   6_10.com
10_13.com  13_6.com   17_7.com   21_3.com   25_17.com  29_18.com  33_0.com   37_10.com  6_11.com
```

   6. Submit all the gaussian input files via `qsubGau`.

```
[Grace@elderberry Forward]$ qsubGau all

Amount of jobs: 1770
Current PATH: /work/Grace/ModifyIRC/run/asymmetric/Nitrene/NCH1/2DPES/59_59/Forward

How many nodes do you want? 100

warning: "-j" option has already been set, overriding previous setting
Your job 420290 ("PortWine1.sge") has been submitted
warning: "-j" option has already been set, overriding previous setting
```

2. Collect rawdata
    1. Make sure all the single points are successfully calculated.

        - Count the amount of successful output files as the primarily checking. If it is different from the expect amount of jobs, go to the next step.

```
> cat *.log | grep -c Normal
```

        - Execute `checkGau` to extract the list of fail jobs (i.e. red box) and the name is recorded in *Fail.txt*.

```
[Grace@elderberry Reverse]$ checkGau all

Program /home/Grace/EST-tool/checkGau will check and alanyse G09 output files
working directory: /work/Grace/ModifyIRC/run/PostTS_H3CO/2DPES/IRC/Shift/QCISD/SP_MP2OPT/Reverse

The extension of output file is : .log

Std-out the information of rawdata

Count the amount of jobs:
  Total:           780
Success:           763
   Fail:            17
```

            - Usually, recalculate the fail jobs can solve the problem.

```
> for name in `cat Fail.txt`
> do
>   g16sub $name.com $name.log
> done
```

        - If above strategy cannot solve the problem, add other SCF keywords in the fail gaussian input files, eg. SCF=xqc.

    2. Use `getPESwStruc.sh` to extract electronic energy and structures.

```
[Grace@elderberry 59_59]$ ls
Forward  header.dat  Reverse  x.xyz  y_F.vec.txt  y_F.xyz  y_R.vec.txt  y_R.xyz
[Grace@elderberry 59_59]$ getPESwStruc.sh
-----------------------------------------------------------------------
Current folder: /work/Grace/ModifyIRC/run/asymmetric/Nitrene/NCH1/2DPES/59_59
It should have two sub-folders, "Forward" and "Reverse"

Output:
1. 59_E.dat
2. 59_Struc.xyz
-----------------------------------------------------------------------
[Grace@elderberry 59_59]$ ls
59_E.dat      Etmp_F.dat  Forward     OM.tmp   Stmp_F.xyz  x.xyz        y_F.xyz      y_R.xyz
59_Struc.xyz  Etmp_R.dat  header.dat  Reverse  Stmp_R.xyz  y_F.vec.txt  y_R.vec.txt
```

3. Plot contour and 3D version figure; execute step 1, 2 and 4 in `plotPESandTraj.py`, which is *line 215* , *line 218* and *line 224*.

```
213 def main():
214     # 1. Input
215     X, Y, E, pts_name, pts_coord = getInput()
216
217     # 2. plot 2D PES with important points
218     twoDwPts(X, Y, E, pts_name, pts_coord)
219
220     # 3. plot 2D PES with mapping trajectories
221     # twoDwTraj(X, Y, E)
222
223     # 4. plot 3D PES with important points
224     threeDwPts(X, Y, E, pts_name, pts_coord)
225
226     # 5. plot 3D PES with mapping trajectories
227     # threeDwTraj(X, Y, E)
```

4. Modify this potential if it is not 'reasonable'.
   - Code:
     - `getGrad.sh`, `genNewPES.sh`, `GenGradStruc.f90`
   - Add *force* in the header.dat file, and then recalculate all the grid points.
   - Use `getGrad.sh` to extract gradient and named as $name.grad.
   - Use `genNewPES.sh` to execute `GenGradStruc.f90`, and then generate several new potentials which are named as *G_*.xyz*.
     - Modify the range of *ds* in `GenGradStruc.f90`.
   - Follow the original processes to calculate those potentials.