2D-PES of Reaction with PTSB

Table of Content

Introduction of PTSB

What I know

1. Some products' ratio cannot be explained by a single-step mechanism (cf. Chemical Science, 2016, 00, 1-3)
2. Selectivity of chemical reaction is dominant by dynamic effect in post-transition-state bifurcation (PTSB). (cf. Nature Chemistry, 2014, 6, 104-111.)
3. Traditional IRC cannot describe PTSB.
4. Most PTSBs calculate AIMD trajectories to interpret dynamic effect.
5. From organic chemists point of view, potential energy surface (PES) is a tool to give a reasonable chemical picture.
6. Full PES is a NP problem.

My strategies

1. Select two important degrees-of-freedom in multi-dimensional space to build a 2D-PES.
   - Combine the normal coordinate and 'local mode picture'.

        - Analysis the reaction path from TSS1 to P1, and then construct the reaction coordinate from VRI region to TSS2.
    - Using the character of VRI for asymmetric case (cf. International Journal of Quantum Chemistry 2015, 115, 258-269).
        - VRI: its zero eigenvalue is orthogonal to the gradient

2. If 2D-PES cannot be built, then build the leading reaction pathways.

PTSB systems

1. Symmetry cases:

    1. CH3O isomerization (International Journal of Quantum Chemistry 2015, 115, 258-269)
    2. C2H4 + HX, X=F,Cl and Br.

2. Asymmetry cases:

    1. CH2O + CH3Cl (International Journal of Quantum Chemistry 2015, 115, 258-269)
    2. HCN (Theoretical Chemistry Accounts 2004, 112, 40-51)

Getting Start

---

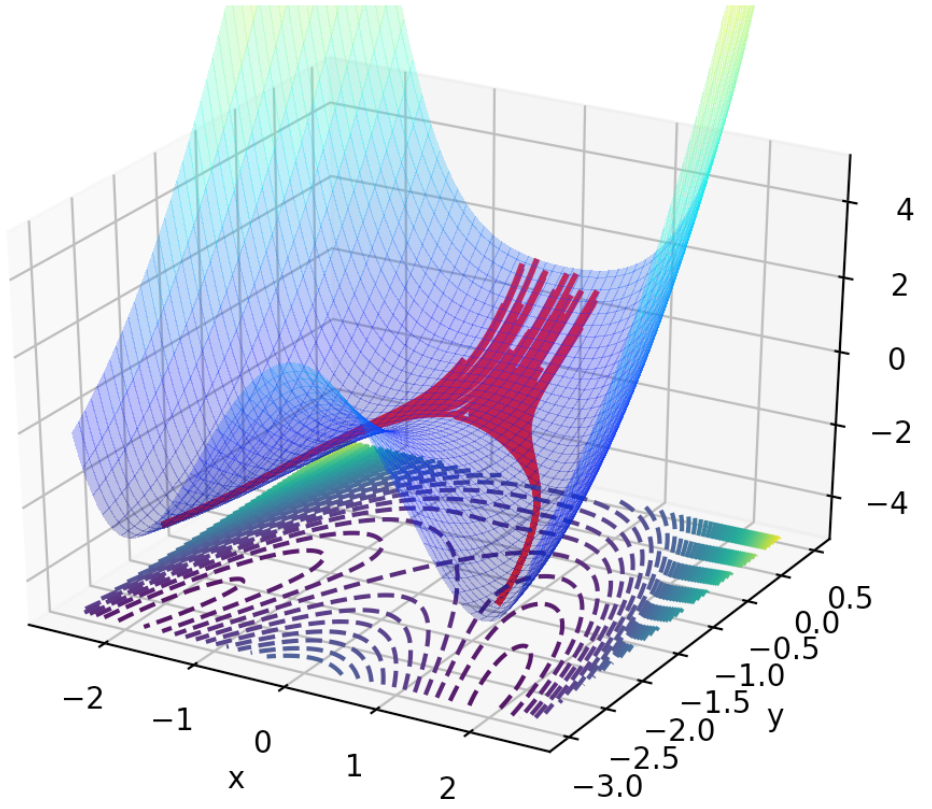
1. Analytical 2D-PES practices

- Goals

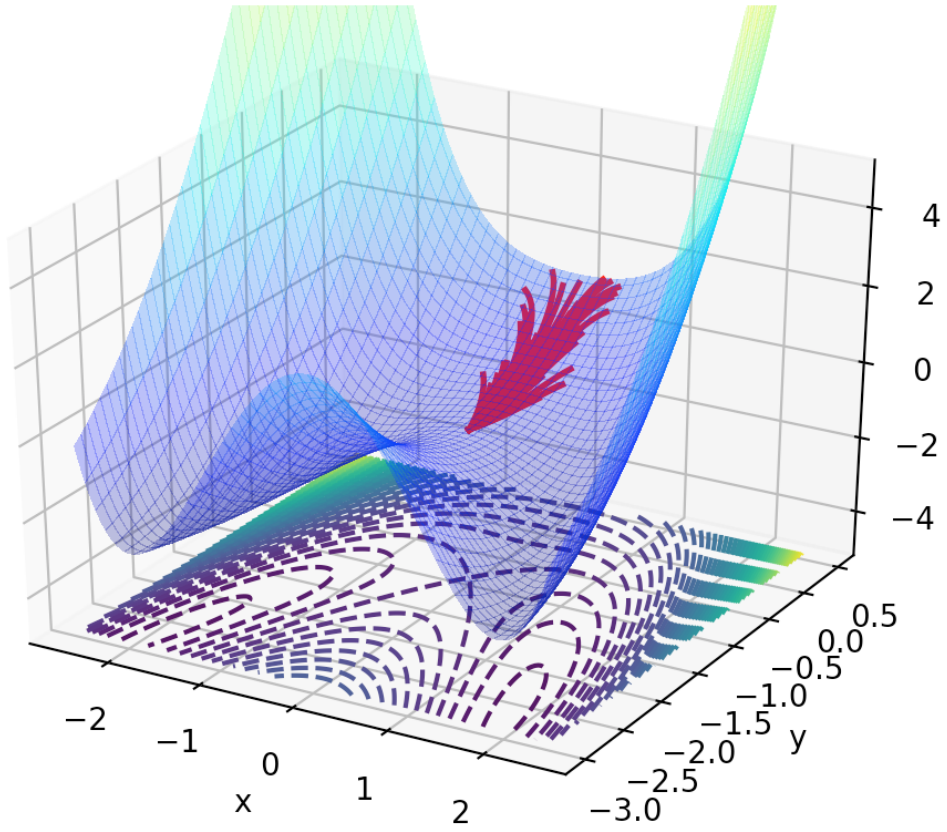
    1. Practice how IRC work
    2. Search the bifurcated pathways

- Search the bifurcated pathways

    - Running steepest decent path from different starting points (VRI point and the points nearby). If the gradient becomes zero, it will switch to the direction of the Hessian eigenvector with a negative eigenvalue.



- Searching the first order saddle point.



2. Numerical 2D-PES with the projection of quasi-classical dynamic trajectories

- **Overview all processes**

1. Calculate normal IRC paths
2. Asymmetric cases: generate artificial reaction coordinate
3. Construct x- and y-axes (optional)
    1. Symmetric cases
    2. Asymmetric cases
4. Select 1D grid points for all IRC paths
5. Scan a 2D-PES
6. Project dynamic trajectories on this 2D-PES

Step 1: Calculate normal IRC paths

- Working directory:
    - remote: run/$(system name)/OPT
- Code:
    - `checkGau`, `run.sh`

1. Quickly run the standard procedure; guess the TSS between two minimums.
    - Use `checkGau` to check all important info. 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
```

2. Re-run the whole process again, because
    1. double check the accuracy of assigned mechanism
    2. double check the atomic index and order for all systems are consistent
        - check the order of atoms between TS1 and TS2 systems. Reorder them via gaussview if they are different.
3. Gain enough grid points of an IRC path
    1. use the homemade script `run.sh` to test different combinations

    2. the direction of IRC path in the output files may be wrong, thus the name of output file may need to be modified manually.
4. Double check the orientation for all systems.
    - Use *gaussview* or *jmol* to check the molecular orientation.

Step 2: Asymmetric cases: generate artificial reaction coordinate

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

- Working directory:
    - remote: run/$(system name)/Artic1D
- Code:
    - `runArticIRC.sh`, `genArticStruc.f90`
- Input:
    - $(selectCoord).log, TS2.log
- Output:
    - Artic1D.xyz, Artic1D_PEC.dat

1. Compare the energy difference between TS1 and TS2:
    1. TS1 > TS2
        - go to the following steps.
    2. TS1 < TS2 and the energy difference is small
        - beyond this project, talk to Grace :)
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 program `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)

The following steps is part of the detail in the macro script, `run1Dgrid.sh`. Step 3 can be skipped if using above script, or, if something wrong that this step can be a reference to debug.

- Working directory:
    - remote: run/$(system name)/1D
- Code:
    - `getIRCcurve`, `getIRCstruc`, `rev1Dstruc`,`checkGau`,`getCoord`

Symmetric cases

- Input:
    - TS1_F/R.log, TS2_F/R.log
- 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

- Working directory:
    - remote: run/$(system name)/1D
- Code: `run1Dgrid.sh`, `get1Dgrid.sh`
    - `run1Dgrid.sh`: `getIRCcurve`, `getIRCstruc`, `selectIRCstruc.sh`, `writeGauInpV`
    - `get1Dgrid.sh`: `checkGau`, `rev1Dstruc`, `rot.py`
- Input:
    - Symmetric cases:
        1. header.dat (route section)
        2. TS1_F/R.log
        3. TS2_F/R.log.
    - Asymmetric cases:
        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. If one wants to change it, go to this part of `run1Dgrid.sh`, and then uncomment *line 211* and comment *line 212*.

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

- This script uses `selectIRCstruc.sh` to select 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 script `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

- Working directory:
  - remote: run/$(system name)/2DPES
- Code:
  - Generate numerical 2D-PES
    - `getIRCvec`, `Vsca1D_IRCvec.f90`, `writeGauInpV`, `qsubGau`
  - 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 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)
```

Step 6: Project dynamic trajectories on this 2D-PES.

- PATH
  - Working directory:
    - remote: run/$(system name)/Traj
  - ProgDyn source code and relative script:
    - remote:
- Code:
  - Calculate trajectories (call *ProgDyn*) and collect rawdata:
    - `submitTraj.sh`, `collectTraj.sh`, `manyTraj.sh`, `CountProd.f90`
  - Map trajectories:
    - `adjustTraj.sh`, `rot.py`, `findCoord.sh`, `MapTraj.f90`
  - Plot potential energy surface with trajectories:
    - `plot2DPES.py`
- Workflow:
  - Remote:
    - /ProgRaw $¥to$ /DoneTraj $¥to$ /totTraj $¥to$ /RP1.reorder and /RP2.reorder
      - /ProgRaw: rawdata for executing *ProgDyn*
      - /DoneTraj: directories with successful trajectories
      - /totTraj: collect all successful trajectories in this directory

        - /RP$n.reorder: classify trajectories into 2 groups
    - Local:
        - /RP$n.reorder $¥to$ /RP$n.reorder.rot $¥to$ /RP$n.reorder.rot.coord
            - /RP$n.reorder.rot: TODO:
            - /RP$n.reorder.rot.coord: TODO:

1. Calculate dynamic trajectories from the program, *ProgDyn* (author: D.A. Singleton), in /ProgRaw.
    - Prepare input files for *ProgDyn*.
        - *.log: or called it as freqinHP; gaussian output file.
        - job: configuration for scheduling system.
        - progdyn.conf: configuration for electronic structure calculation.
        - proganal: criteria to stop dynamic calculation.
    - Execute script `submitTraj.sh` to replicate /template directory to several copies, /n$i, $i=1,2,3..., and then submit them automatically.
        - Double check the PATH in this script, which are *line 9* to *line 11*.
        - Change the amount of duplicate files by modify the variable, $finalfile, which is *line 13*.

```
 9  home='/work/Grace/Tantillo_Dynamic/H3CO/QCISD/ProgRaw'
10  doneTraj='/work/Grace/Tantillo_Dynamic/H3CO/QCISD/DoneTraj'
11  tempDir='/work/Grace/Tantillo_Dynamic/Program/templet' #do not change this line
12  inifile=1 # first number of file -> n$inifile
13  finalfile=40 # final number of file -> n$finalfile
```

        - Check the function *main()* for this process, which uncomment step 1 and 2; *line 107* and *line 109* to *line 111*.

```
104  # main program
105  function main(){
106      # Step 1. stdout the purpose
107      purpose
108      # Step 2. Initialize and for the first time to submit all jobs
109      cd $home
110      rm -f runlist.dat faillist.dat worklist.dat dirlist.dat
111      submit1st $inifile $finalfile
112      # Step 3. Count the total amount of trajectories
113      # countTraj # output: $home/dirtraj.dat
114      # Step 4. Move all sucessful file to DoneTraj
115      # move2Done # output: $home/faillist.dat and $home/worklist.dat
116      # Step 5. resubmit fail lsit
117      # resubmitFail
118      # rm -f runlist.dat faillist.dat worklist.dat dirtraj.dat dirlist.dat
119  }
```

2. Collect successful trajectories and re-calclulate fail trajectories; step 3 to 5 in `submitTraj.sh`, in /ProgRaw and /DoneTraj.
    - Execute script `submitTraj.sh` again to classify trajectories into successful and fail categories.
        - Move successful trajectories to directory /DoneTraj.
        - Resubmit fail directories in /ProgRaw. Sometimes, this step may needs to iterate several times until get the enough amount of trajectories. For example, one wants more than 100 trajectories.

```
[Grace@elderberry Program]$ ./submit.sh
------------------------------------------------------------
Calculate quasi-classical trjaectories by Signlton's program.
------------------------------------------------------------

1. First time to execute this script:
        check the path is correct or not!!

2. Following check the status of trajectories:
Move all sucessful directories to /work/Grace/Tantillo_Dynamic/H3CO/QCISD/Do
neTraj
and resubmit fail dir.

Current total trajectories in this dir.: 4773
```

3. Collect trajectories rawdata, and then group them into two groups.
    - Execute `collectTraj.sh` to collect trajectories from /DoneTraj to /totTraj.

```
[Grace@elderberry Program]$ ./collectTraj.sh
------------------------------------------------
Collect trajectory
from /work/Grace/Tantillo_Dynamic/H3CO/QCISD/DoneTraj
to /work/Grace/Tantillo_Dynamic/H3CO/QCISD/totTraj
and then,
1. rename them
2. change atomic number from character to integer
------------------------------------------------
Total amount of trajectories: 4773
```

    - In /totTraj, execute `manyTraj.sh` to call `CountProd.f90`, and then move selected trajectories to /RP1.reorder and /RP2.reorder.
        - Prepare the following input files for `manyTraj.sh`
            - defR.dat: criteria for defining reactant.
            - defProd1.dat and defProd2.dat: criteria for defining product 1 and 2.
        - Cut the redundant trajectory which is excess on this PES.
        - List all the detail information if there are more than 100 trajectories.

```
[Grace@elderberry Program]$ ./manyTraj.sh
If there are more than 100 trajectories,
print the result one-by-one.
Traj1 R P1
Traj10 R P1
Traj100 R P1
Traj1000 R P2
Traj1001 R P2
```

        - Standard output the statistic report in the end.

```
------------------------------------------------------------

    Analyse the initial/final point of all the trajectories

output:
    files
        1. /RP1list.dat
        2. /RP2list.dat
    directories
        1. /work/Grace/Tantillo_Dynamic/H3CO/QCISD/RP1.reorder
        2. /work/Grace/Tantillo_Dynamic/H3CO/QCISD/RP2.reorder

std-out:
total traj.: 4773

OtherOther: 0
RR: 785
P1P1: 0
P2P2: 0
OtherR: 0
OtherP1: 0
OtherP2: 0
P1P2: 0

RP1: 1996
RP2: 1992
------------------------------------------------------------
```

4. Map trajectories on this 2D-PES; 1. rotate trajectories and 2. search their corresponding coordinates.
    - In the end, we will have those output directories:
        - /RP1.reorder.rot
        - /RP2.reorder.rot
        - /RP1.reorder.rot.coord
        - /RP2.reorder.rot.coord
    - Copy rawdata from remote to local.

      ```
      scp -P #port $USER@$IP:/$REMOTE_PATH /$LOCAL_PATH
      ```

        - Because the orientation of trajectory and the structures on 2D-PES are usually different, such that the trajectories are needed to be modified.
        - It is easier to manipulate them on local terminal for visualized purpose. Thus, copy directories /RP1.reorder and /RP2.reorder to local terminal.
    1. Execute `adjustTraj.sh` to call `rot.py`: rotate all the structures of trajectories.
        - Prepare the following input files:
            - $(NGrid)_Struc.xyz: all the structures in this 2D-PES.
            - /RP$n.reorder: trajectories in two groups.
        - In `adjustTraj.sh`, modify the name of input file which is *line 20*, and the PATH from *line 24* to *line 28*.

          ```
          18  rotProg='/Users/Grace/Google_Drive/Code/GitHub/ModifyIRC/src/rot.py'
          19  # first input, $1 ##################################################
          20  tsPESname='59_Struc.xyz'
          21  ####################################################################
          22  tsPESpath='/Users/Grace/Google_Drive/Code/GitHub/ModifyIRC/run/NCH1/'
          23  # second input, $2
          24  iniRP1='/Users/Grace/Google_Drive/Code/GitHub/ModifyIRC/run/NCH1/Traj/RP1.reorder/'
          25  iniRP2='/Users/Grace/Google_Drive/Code/GitHub/ModifyIRC/run/NCH1/Traj/RP2.reorder/'
          26  ####################################################################
          27  finRP1='/Users/Grace/Google_Drive/Code/GitHub/ModifyIRC/run/NCH1/Traj/RP1.reorder.rot/'
          28  finRP2='/Users/Grace/Google_Drive/Code/GitHub/ModifyIRC/run/NCH1/Traj/RP2.reorder.rot/'
          ```

        - Input directories $¥to$ output directories.

    - /RP1.reorder $¥to$ /RP1.reorder.rot
    - /RP2.reorder $¥to$ /RP2.reorder.rot
  - While executing, print out the RMSD for each trajectory before/after rotation.

```
GracedeMBP-2:src Grace$ ./adjustTraj.sh
1.6744  0.0714
1.6694  0.0818
1.6748  0.0815
1.6726  0.0782
1.7281  0.0899
1.6771  0.0618
1.6832  0.0895
1.6789  0.0686
1.6380  0.0587
1.6517  0.0942
1.7022  0.0701
1.6713  0.0763
1.6895  0.0463
```

RMSD (After rotation)

RMSD (Before rotation)

2. Execute `findCoord.sh` call `MapTraj.f90`: searching the corresponding coordinates of trajectories.
   - /RP1.reorder.rot -> /RP1.reorder.rot.coord
   - /RP2.reorder.rot -> /RP2.reorder.rot.coord

```
GracedeMBP-2:src Grace$ ./findCoord.sh

-------------------------------------------------------------------------
|  Purpose:                                                             |
|   Mapping the trajectories to numerical PES by RMSD                   |
|                                                                       |
|  Limitation:                                                          |
|   1. the potential should be square; amount of point from x = y       |
|   2. atomic number in trajectory must be integer                      |
|                                                                       |
|  Output file:                                                         |
|   coord.rot.reorder.Traj1                                             |
-------------------------------------------------------------------------
```

5. Plot 2D and 3D figures with the projection of trajectories; execute step 1, 3 and 5 in `plotPESandTraj.py`, which is *line 215* , *line 221* and *line 227*.

```
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)
```

Authors

- Hsiao-Han (Grace) Chuang - *Initial work* - 2018 May to 2019 June.

Acknowledgments

- Thanks for the MOST exchange project, 107-2917-I-002-004.