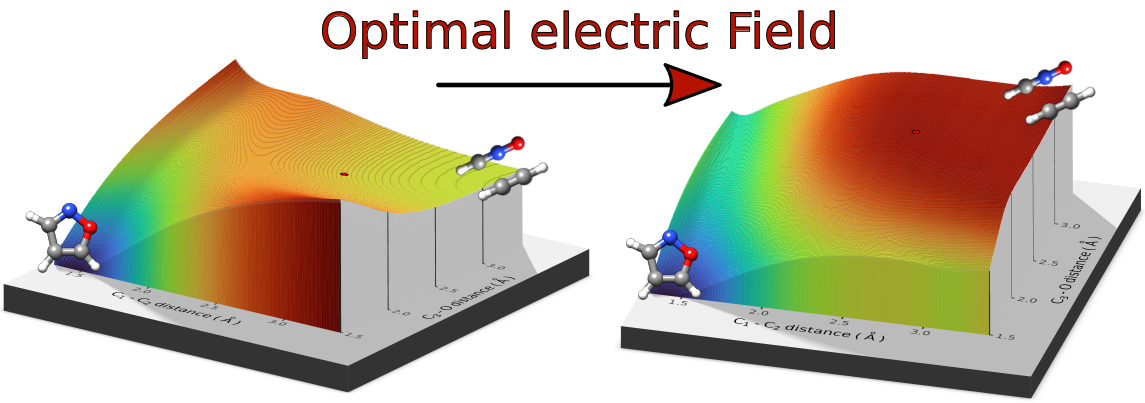


MANULS

Make chemicAl reactioNs spontaoUs via optimaL fieldS

Documentation and User Guide

Version 1.0.0

The Authors

March 30, 2023

Contents

1 Goal of the Program

This program calculates the electric field that can be used to make a chemical reaction barrierless, in other words the field that removes the energy difference between the reactants and the transition state.

The field found with this program is *optimal*, that is the electric field of the least intensity that can induce a barrierless chemical reaction.

This work lies in the context of electrostatic catalysis, that is the usage of electric fields to accelerate chemical reactions, and the core idea of this approach is that it can be demonstrated that the optimal electric field can be determined upon locating a special point of the potential energy surface (PES), the so-called "optimal bond-breaking point" (optimal BBP). The interested reader can find further information about the theoretical approach in section 6.1 and, with even further detail, in references XXXX-YYYY-ZZZZ.

An example of the results of this code can be found in fig. 2.

2 Using the Program

2.1 Installation

The code is simply a python script. The dependencies are:

- `Python 3`
- The `Numpy` library [1]

The script has been tested with

- `Python 3.8` and `Python 3.9`
- `Numpy 1.19.5` and `Numpy 1.24.1`

XXX can be dowloaded using the git clone YYYYYY command or using the green button at the top of the repository page **link**. After pressing the button the user can choose "download ZIP" to download the code.

2.2 Preparation of the Input

2.2.1 Preliminary Calculations

To run this program the user must possess the data that define a potential energy surface or a potential energy curve. For each geometry described in the curve or surface the program needs to read the electric dipole moment and the polarizability matrix. The user can perform the calculations with any kind of method and software with which it is possible to calculate the energy and the necessary properties of the system under examination.

2.2.2 Formatting the Data

The program starts reading two files. These files contain the data extracted from the previous calculations. One file contains the information regarding the potential energy surface and the dipole, and one file contains the polarizability matrices calculated for all the geometries.

We provide, along with this program, a script to extract and format the data from ORCA [2] scans, therefore we included section 2.2.3 to show how to use the script to automatically prepare the data.

For the users of all the other programs: don't worry, the preparation of the files takes a couple of minutes.

The program needs to read two files. Suppose that the calculation performed is a two-dimensional scan, and the two varibles scanned are "Var1" and "Var2". Then the first file must contain the data in the following format. The first two columns describe the point of the scanned surface. The third column contains the energy in that point, the fourth column contains the x component of the electric dipole moment vector in that point, the fifth column contains the y component of the electric dipole moment vector in that point and the sixth column contains the y component of the electric dipole moment vector in that point.

```
Var1_a Var2_a Energy Dip_x Dip_y Dip_z
Var1_a Var2_b Energy Dip_x Dip_y Dip_z
Var1_a Var2_c Energy Dip_x Dip_y Dip_z
Var1_a Var2_d Energy Dip_x Dip_y Dip_z
```

```
Var1_b Var2_a Energy Dip_x Dip_y Dip_z
Var1_b Var2_b Energy Dip_x Dip_y Dip_z
Var1_b Var2_c Energy Dip_x Dip_y Dip_z
.      .
.      .
.      .
```

For example: suppose that the user scanned two bonds from 1.5 Å to 2.0 Å with a step of 0.1 Å, the file needed for the program will be like the following.

```
1.5 1.5 Energy Dip_x Dip_y Dip_z
1.5 1.6 Energy Dip_x Dip_y Dip_z
1.5 1.7 Energy Dip_x Dip_y Dip_z
1.5 1.8 Energy Dip_x Dip_y Dip_z
1.5 1.9 Energy Dip_x Dip_y Dip_z
1.5 2.0 Energy Dip_x Dip_y Dip_z
1.6 1.5 Energy Dip_x Dip_y Dip_z
1.6 1.6 Energy Dip_x Dip_y Dip_z
.   .
.   .
.   .
2.0 2.0 Energy Dip_x Dip_y Dip_z
```

If the user wants to work with a 1-dimensional scan it is sufficient to not consider the second column and proceed in the same way.

The second file contains the polarizability matrices. The matrices must be reported in the order defined by the coordinates in the first file. That is, continuing to discuss the prototypical example, the first matrix must be calculated in the (1.5,1.5) point, the second in the (1.5,1.6) point, the third in the (1.5,1.7) point and so on. So, if the matrix is called A and the elements are a, the file is of the form

```
a_xx a_xy a_xz
a_yx a_yy a_yz ###### 1.5,1.5 point (do not include this comment in your file,
a_zx a_zy a_zz ###### the program reads only numbers)
a_xx a_xy a_xz
a_yx a_yy a_yz ###### 1.5,1.6 point
```

```
a_zx a_zy a_zz
a_xx a_xy a_xz
a_yx a_yy a_yz ###### 1.5,1.7 point
a_zx a_zy a_zz
.    .    .
.    .    .
.    .    .
a_xx a_xy a_xz
a_yx a_yy a_yz ###### 2.0,2.0 point
a_zx a_zy a_zz
```

The user can name these two files as he prefers.

2.2.3 ORCA Users

We provide a bash script called `data_extraction_ORCA.sh` that automatically produces the files containing the data for the code. At the moment this script works only for relaxed surface scans and only for methods in which the polarizability matrix is printed only once for each optimized geometry. For instance this script cannot be used with the MP2 method because the output of ORCA contains the polarizability computed at the HF level and at least one polarizability matrix computed after the perturbative correction. From our experience there are no problems with DFT methods.[1] The script must be run in the directory that contains the output files of the ORCA scan calculation. If the calculation input file is called `scan.inp` then the script needs the output file, that must be called `scan.out`, and the file that contains the information about the relaxed scan, that is `scan.relaxscanact.dat`, the latter file is automatically produced by ORCA.

To run the script the user needs to run

```
./data_extraction_ORCA.sh name_input_without_extension
```

if the input is called `scan.inp` then the command is

[1] We are working on improving the script, in order to make it more general

```
./data_extraction_ORCA.sh scan
```

The script outputs two files:

- `name_input_grid.txt` (`scan_grid.txt`)
- `name_input_polarizability.txt` (`scan_polarizability.txt`).

The first file contains the scanned variables, the energies and the dipole components; this file, as explained in section 2.3, will be passed to the `data` variable in the python code. The second file contains the polarizability matrices and will be passed to the `polar` variable in the python code.

2.3 Running the Code

Once the user has produced, or manually or with the script, the necessary files the procedure is almost over.

The user now just needs to modify a few lines of the XXXX file and let it run. After opening the python program the user will see the following part of the code. This is the only section that needs to be modified by the user.

```
########################################################################
#################### USER DEFINED SECTION  ##########################
########################################################################

# This is the only section that needs to be modified, the rest is up to the code

# Loading of the files containing the data, please see the documentation

data=np.loadtxt('/path/to/file/scan_grid.txt')
polar=np.loadtxt('/path/to/file/scan_polarizability.txt')



# "pointsx" and "pointsy" = number of points scanned for each variable
pointsx=int(20)
```

```
pointsy=int(20)

# "step_x" and "step_y" = step size used in the scan
step_x=0.05
step_y=0.075


# number of directions used for the search of the optimal field,
# usually the default gives a satisfactory time/precision ratio.
n_directions=100000

#geometry of the reactants
geometry_reactant_x=1.5
geometry_reactant_y=1.6
```

It is more immediate to consider an example: suppose that the user scanned 20 points for each variable, with a step size of 0.05 for the x variable and 0.075 for the y variable. Moreover the user knows that the reactants are described in the point x=1.5 and y=1.6.

It is necessary to insert the path to the file containing the geometries, the energies and the dipoles in the `data` variable and the path to the file containing the polarizability matrices in the `polar` variable.

Then the values of the parameters `pointsx` and `pointsy` must be modified in order to be equal to the number of points scanned in each directions, in this case we have `pointsx=20` and `pointsy=20`.

Similarly the `step_x` and `step_y` parameters must be contain the step size used in the scan, the two step sizes can be different. In this case we have `step_x=0.05` and `step_y=0.075`.

The user can enter (**this is optional**) the value of the variables under consideration in the reactant geometry in the `geometry_reactant_x` and `geometry_reactant_y` variables. In this case the user is scanning two bond distances and the user knows that these bond distances in the reactant geometry are 1.5 and 1.6 (according to the unit of measure used in the quantum chemical calculations), then the input file will contain `geometry_reactant_x=1.5` and `geometry_reactant_y=1.6`. It is not necessary to insert a value that has been scanned, the program will find the point on the grid

closest to the geometry of the reactants written by the user. This step ensures that the potential energy surface is planar in the region between the reactants and the optimal bond-breaking point and, therefore, that the reaction is perfectly catalyzed.

If the user does not have a reasonable guess of the geometry of the reactants or is not interested in this functionality it is sufficient to put `geometry_reactant_x=-1000` and `geometry_reactant_y=-1000`, this will let the code skip the check about the energies.

We leave `n_directions=100000`, as default. This variable represents the number of directions in which the code looks for the optimal electric field. In principle, the higher this number, the better.

Lastly, the user just needs to run the code using the following command:

```
python3 XXXX.py
```

2.4 Output of the Code

The code outputs one file called `external_electric_fields.txt` and one called one file called `coordinates_bbps.txt`.

One example of the `external_electric_fields.txt` output file is the following:

```
Geometry of the optimal BBP
variable 1 = 2.65
variable 2 = 2.625
gradient extremal condition 3.691592286784939e-08
norm original gradient 0.030537826008480887
------------------------------------------------------------
------------------------------------------------------------
Optimal electric field (a.u.)
[-0.07662794  0.00314703 -0.04301013]
Amplitude of the optimal electric field (a.u.)
0.08792961021219792
Direction of the optimal electric field
[-0.87146913  0.03579036 -0.48914272]
Optimal electric field (V/m)
[-3.94036832e+10  1.61826951e+09 -2.21167039e+10]
```

```
Amplitude of the optimal electric field (V/m)
4.5215e+10
Original Gradient
[0.0217156 0.0135684]
Perturbed Gradient
[ 1.79688707e-05 -2.87160654e-05]
f Function
7.676886891329399e-06
------------------------


Sub-optimal electric field (a.u.)
[-0.0765421  -0.00878018 -0.04300376]
Amplitude of the sub-optimal electric field (a.u.)
0.08823325644500997
Direction of the sub-optimal electric field
[-0.86749711 -0.099511   -0.48738724]
Sub-optimal electric field (V/m)
[-3.93595395e+10 -4.51495098e+09 -2.21134306e+10]
Amplitude of the sub-optimal electric field (V/m)
4.5371e+10
Original Gradient
[0.0217156 0.0135684]
Perturbed Gradient
[ 9.19169561e-06 -1.46998098e-05]
f Function
7.3864089164456315e-06
```

The output file contains, at its top, the information about the optimal bond-breaking point. First of all the file contains its location on the grid supplied by the user and secondly two values to assess the quality of the result. The `gradient extremal condition` value must be as small as possible and the `norm original gradient` must be different from zero. For further details we refer the interested reader to section 6.

Secondly the file contains the optimal external electric field, its magnitude and its direction, along with the gradient of the original/unperturbed PES. The output contains also the gradient of the PES perturbed by the optimal electric field, this vector should be close to the zero vector. Lastly, as a sanity check, the code prints also the f function,

its value should be close to zero (see section 6).

The code also prints all the fields that satisfy the requisites for the optimality but are not the optimal one. These fields are labeled as "Sub-optimal" electric field and are printed mainly for the interested user.

The `coordinates_bbps.txt` contains the first n points in which gradient extremal condition is the lowest, that is the points in which the norm of the product between the hessian and the gradient is the lowest. Between these points the optimal bond-breaking point is the point in which the norm of the gradient is maximal. This file is for debugging purposes and for the interested user, but it may be useful in some cases (see section 5.1).

3 Visualization of the Results

We provide two simple python scripts to visualize the effect of the optimal electric field on the potential energy surface/curve.

The scripts rely on the `Matplotlib` library [3] and they produce a representation of the unperturbed PES and of the PES perturbed by the external field. The scripts are called `MANULS_plot_1D.py` and `MANULS_plot_2D.py`

Also in this case the user just needs to modify a small part of the script to let it run. The part to be modified is the following

```
########################################################################
#################### USER DEFINED SECTION  ##########################
########################################################################

# This is the only section that needs to be modified, the rest is up to the code

# Loading of the files containing the data, please see the documentation
data=np.loadtxt('/path/to/file/scan_grid.txt')
polar=np.loadtxt('/path/to/file/scan_polarizability.txt')


# "pointsx" and "pointsy" = number of points scanned for each variable
pointsx=int(20)
pointsy=int(20)

#the optimal electric field
```

```
#e=np.array([field_x,  field_y,  field_z])
e=np.array([-0.00395539,  0.00419754,  0.00640764])
```

The user needs to insert the location of the files that contain the information about the geometries, the energies, the dipole vectors and the polarizability matrices in the same way as described in section 2.3. Then the user must specify the number of points scanned and the external electric field to be applied. To see the effect of the optimal electric field on the PES it is sufficient to copy the optimal field from the `external_electric_fields.txt` and paste its components in the `e` array.

To run the script it is sufficient to type

```
python3 MANULS_plot_2D.py
```

or

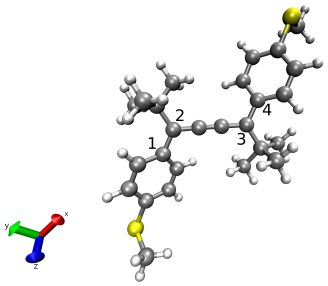
```
python3 MANULS_plot_1D.py
```

4 Examples

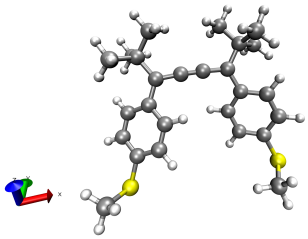
4.1 One-Dimensional Example

As a one-dimensional example we consider the s-trans to s-cis isomerization of a [3] cumulene derivative. In this simple case the modification of the central dihedral angle is sufficient to describe the isomerization. Figure 1 shows the geometry of the s-trans isomer, the s-cis isomer and the transiton state.

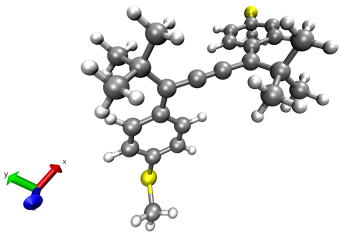
The first thing to do is to build the potential energy curve. For this prototypical example we employ the UHF method with the 6-31G* basis set, the calculations are performed with the broken symmetry formalism. We scan the central dihedral angle, that is the angle defined by the $C_1 - C_2 - C_3 - C_4$ atoms, with a step of 5° and for each step we perform a constrained geometry optimization fixing the dihedral angle. By convention the s-cis isomer is described by a 0° dihedral angle, while the s-trans is described by a 180° dihedral angle. **Since we are interested in the s-trans to s-cis**



(a) *s-trans isomer*



(b) *s-cis isomer*



(c) *transition state*

Figure 1: Representation of the structure of the stationary points of the potential energy curve along with the numbering used to define the central dihedral angle

isomerization it is extremely important to scan from 180° to 0° and not vice versa, otherwise the derivatives and the location of the optimal BBP may be affected.

Once the calculation is done we build the files that the code needs to read. The file that contains the geometry, the energies and the dipole vector looks like the following and we call it `cumulene_scan_grid.txt`. For each step of the scan we report in this file the geometry parameter scanned, the energy of the optimized structure and the electric dipole moment vector of the optimized structure.

```
180.00000000 -1798.16949020       0.27163      -0.46748       0.34143
175.00000000 -1798.16960962       0.23543      -0.46536       0.38220
170.00000000 -1798.16959900       0.19476      -0.45679       0.41608
165.00000000 -1798.16943112       0.15904      -0.45591       0.45390
160.00000000 -1798.16913860       0.12159      -0.45236       0.49016
155.00000000 -1798.16875906       0.07603      -0.43843       0.52132
150.00000000 -1798.16830133       0.03513      -0.42975       0.55562
.............................................................
  0.00000000 -1798.16980280      -0.88252      -0.20722       1.18954
```

The file that contains the polarizability matrices looks like the following and we call it `cumulene_scan_polarizability.txt`. For each step of the scan we put in this file the polarizability matrix of the optimized structure.

```
 421.59312     13.45912     -6.36864
  13.45912    260.40956    -39.33521
  -6.36864    -39.33521    212.02226
 422.36059     13.07928     -6.15173
  13.07928    259.57308    -38.65672
  -6.15173    -38.65672    212.10784
 422.28933     12.56068     -5.75969
  12.56068    259.11684    -38.12368
  -5.75969    -38.12368    212.39407
 421.44753     11.92634     -5.45400
  11.92634    258.49941    -36.91505
  -5.45400    -36.91505    213.25232
  ..............................
 318.02045    -50.38743     36.70763
 -50.38743    250.30417     60.73073
  36.70763     60.73073    311.24669
```

Now we just need to modify the 1-D version of the code. The code needs to read the `cumulene_scan_grid.txt` file and store it in the `data` variable and to read the `cumulene_scan_polarizability.txt` file and store it in the `polar` variable, therefore the first thing to do is to write down the path to those two files. Then we have scanned 37 points with a step of 5°, so we put `pointsx=int(37)` and `step_x=5.0`. We leave `n_direction` as default. We see from the data above that the energy minimum closest to the s-trans isomer is defined by a 175° dihedral angle, so we put `geometry_reactant_x=175`. The code should be similar to the following

```
########################################################################
#################### USER DEFINED SECTION  ############################
########################################################################

# This is the only section that needs to be modified, the rest is up to the code

# Loading of the files containing the data, please see the documentation

data=np.loadtxt('/path/to/file/cumulene_scan_grid.txt')
polar=np.loadtxt('/path/to/file/cumulene_scan_polarizability.txt')



# "pointsx" and "pointsy" = number of points scanned for each variable
pointsx=int(37)


# "step_x" and "step_y" = step size used in the scan
step_x=5.0

# number of directions used for the search of the optimal field,
# usually the default gives a satisfactory time/precision ratio.
n_directions=100000

#geometry of the reactants
geometry_reactant_x=175
```

Now we can save the changes, close and run the code

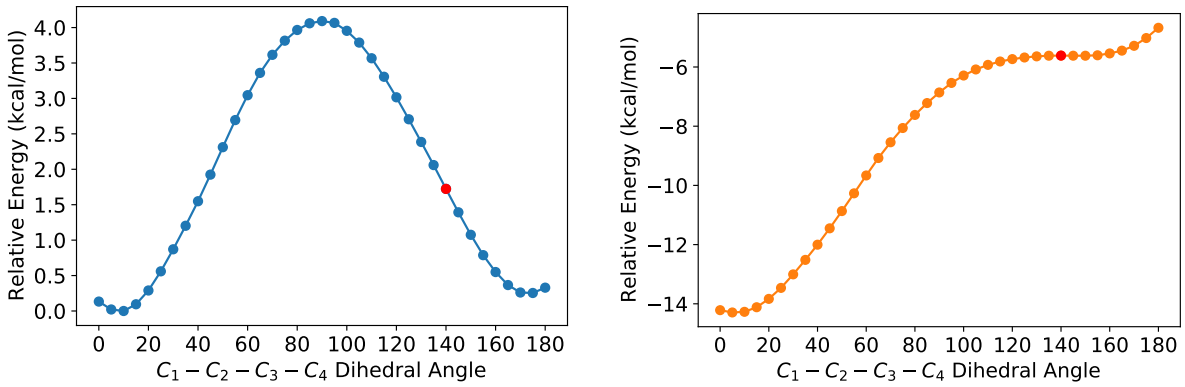
```
python3 MANULS_1D.py
```

For this system the code took a couple of minutes to run on our desktop machines. After the code completed its job we can inspect the `external_electric_fields.txt` file. We see the following

```
Geometry of the optimal BBP
variable 1 = 140.0
gradient extremal condition 3.965660428184293e-11
norm original gradient 0.00010626099999626604
-----------------------------------------------------
-----------------------------------------------------
Optimal electric field (a.u.)
[-0.00395539  0.00419754  0.00640764]
Amplitude of the optimal electric field (a.u.)
0.008621035084984683
Direction of the optimal electric field
[-0.4588063   0.48689487  0.74325646]
Optimal electric field (V/m)
[-2.03394146e+09  2.15846133e+09  3.29494194e+09]
Amplitude of the optimal electric field (V/m)
4.4331e+09
Original Gradient
0.00010626099999626604
Perturbed Gradient
1.3552527156068805e-20
f Function
1.0485415336859148e-11
------------------------
```

We see that the optimal bond-breaking point corresponds to a dihedral angle equal to 140° and that the optimal external electric field is in the $(-0.459, 0.487, 0.743)$ (**the direction is computed with respect to the frame of reference of the molecule**) direction and its amplitude is 0.0086 a.u. = $4.43 \cdot 10^9$ V/m.

Lastly we compute the potential energy curve under the effect of the optimal electric field (see section 3). Figure 2 compares the potential energy curve with and without the effect of the field and the figure nicely shows how the energy barrier is completely removed.



(a) *without field* (b) *with field*

Figure 2: Potential energy curve that describes the s-trans/s-cis isomerization of a [3]-cumulene derivative. (a) Without optimal field (b) with optimal field. The energy values are relative to the energy minimum of the PES calculated without the effect of the electric field. The red dot represents the location of the optimal bond-breaking point.

5 Tips and Tricks

5.1 Choose correctly the optimal BBP

To obtain "correct" results it is fundamental to choose correctly the optimal bond-breaking point.If the code does not produce a field that nicely removes the barrier the problem probably is in the choice of the optimal bond-breaking point. Unfortunately there is not a single gradient extremal point, but a family of gradient extremal points. As explained in XXX for the purposes of this program the optimal bond-breaking point can be located starting from the IRC [4] trajectory. The point, along the IRC trajectory, in which the gradient is maximum is a good starting point for the location of the optimal bond-breaking point. In other words it is a reasonable choice to scan the PES around this point.

In general, ***as a rule of thumb***, the optimal bond-breaking point is not too far from the IRC trajectory and should be somewhere in the middle between the transition state and the reactants. So, if the program finds as an optimal bond-breaking point a point in which the first or second coordinate is close to the transition state be careful and check the results.

From the tests we performed the proposed algorithm is quite robust but we imple-

mented a way to specify the coordinates of the optimal bond-breaking point in case the point chosen by the algorithm is not correct. We show how to fix this problem with an example.

Suppose that the scan is performed from 1.5 Å to 2.0 Å with a step of 0.025Å, and we know that the transition state is at x=1.90 Å and y=1.95 Å. We build the input files, we run the code, and we do not find a field that removes the barrier.

We inspect the `coordinates_bbps.txt` file and we see the following

```
Geometry of the BBP
variable 1 = 1.875
variable 2 = 1.9
index BBP = [19. 22.]
gradient in this point = [0.0264526 0.0152584]
gradient norm in this point = 0.030537826008480887
gradient extremal condition = 3.691592286784939e-08
#########

Geometry of the BBP
variable 1 = 1.6
variable 2 = 1.525
index BBP = [9. 5.]
gradient in this point = [0.013018  0.0077364]
gradient norm in this point = 0.015143322256429637
gradient extremal condition = 4.996371408428265e-07
#########

Geometry of the BBP
variable 1 = 1.65
variable 2 = 1.625
index BBP = [15. 14.]
gradient in this point = [0.0217156 0.0135684]
gradient norm in this point = 0.025606029796159
gradient extremal condition = 7.830945739279751e-07
#########
```

The point chosen as optimal bond-breaking point is the first one, since the norm of the gradient is the highest and in this point x=1.875 Å and y=1.90 Å. This is very close to the transition state and can give erroneous results for the reasons explained above.

The second best candidate for the role of optimal bond-breaking point is the third

point, where x=1.65 Å and y=1.625 Å, because it has the second highest gradient norm, so we can pick this as the optimal BBP.

To do so it is sufficient to modify the `MANULS_2D.py` file, in order to tell the code the correct position of the optimal BBP. We need to find the foollowing lines

```
obbp_idx=indexes[gradient_maxima[np.argmax(array_for_max_gradient)]]
obbp_x=int(obbp_idx[1])
obbp_y=int(obbp_idx[0])
```

and substitute the index of the new optimal BBP, in this case (**the order is important, x and y are switched)**

```
obbp_idx=indexes[gradient_maxima[np.argmax(array_for_max_gradient)]]
obbp_x=int(14)
obbp_y=int(15)
```

Then we can re-run the code and, hopefully, get the correct results. Notice that after this modification in the `external_electric_fields.txt` the value of the gradient extremal condition will not be updated, but can be found in the `coordinates_bbps.txt` file.

6 Further Details

6.1 Theoretical Background

6.2 Code Overview

The code works as follows:

1. The code reads the data from the files supplied by the user and reshapes the data in matrices, in this way it is easier to implement the derivatives.

2. The code locates the optimal bond-breaking point. To do so the hessian and the gradient of the original/unperturbed PES are computed in each point. If the unperturbed hessian is called H and the unperturbed gradient is called g for each point of the grid the code computes $\|H \cdot g\|$ (the norm of this dot product).

Then it picks the `n_bbps` points in which $\|H \cdot g\|$ is the smallest and the optimal bond-breaking point is the point, chosen between these `n_bbps`, in which the unperturbed gradient is maximum. The default for `n_bbps` is 5, it can be changed by the user.

3. The code generates several points on the unit sphere. The points are generated using the Fibonacci sphere algorithm. Each point, along with the origin, defines a unit vector. The unit vectors generated are stored and used later. In principle, the higher is the number of directions generated, the better is the quality of the result.

4. From now on the code works only on the optimal bond-breaking point. The code picks a direction computed in point 2 and calculates the "ingredients" needed for the analysis. It computes the first and second derivatives of the dipole, of the f_e vector, the hessian and the gradient of the original PES. (see eqs. XXX of YYY)

5. With the direction of the field fixed the code computes the amplitude of the electric field by solving the quadratic equation XXXX

6. Now the code found the amplitude and the direction of the field. It needs to check if the field satisfies the requirements for the optimality. To do so there are three (plus one optional) tests:

   (a) The code computes the f function (see eq XXX in YYY), the absolute value should be as close as possible to zero. Numerically speaking if the absolute value of the f function is below a threshold the test is passed.

   (b) The code computes the quantity $\frac{m \cdot e}{\|g\|}$, the absolute value of this quantity should be as close as possible to 1, if the absolute value of this quantity is over a threshold the test is passed.

   (c) The code computes the norm of the gradient of the PES perturbed by the external field computed in the previous points. This quantity should be as close as possible to zero, if this quantity is below a threshold the test is passed.

   (d) (**Optional**) The mathematical nature of the problem admits many solutions, this optional test helps to select the one relevant to the chemical problem under consideration. The user has to input the coordinates that define (or

that are close to) the reactants. Then the code finds the point of the grid that is closest to these coordinate, that is the point on the grid closest to the reactants. Now the code checks, in the point closest to the reactants, that the energy under the effect of the field is equal or higher to the energy of the optimal bond-breaking point, this ensures that the optimal field correctly catalyzes the reaction [2].

If all the tests are passed the field is stored. The procedure is repeated for all the directions generated by the Fibonacci algorithm.

7. Lastly the code checks all the fields that satisfy the requirements for the optimality and picks, as the optimal external electric field, the field with the lowest amplitude.

6.3 Thresholds

In the previous section we discussed the usage of the code without discussing the numerical aspects.

For each test described in section 6.2 is defined a numerical threshold. All the threshold are defined a few lines below the end of the "user defined section".

- `f_tol=5e-5` is the threshold on the f function
- `m_tol=0.999` is the threshold on the $\frac{m \cdot e}{\|g\|}$ value
- `g_pert_tol=5e-5` is the threshold on the norm of the perturbed gradient
- `energy_tol=-0.0048` is the threshold on the difference between the energy of the optimal BBP under the effect of the field and the energy of the reactants under the effect of the field. The default value is the equivalent, in Hartrees, of 3 kcal/mol. In practice the energy of the reactants under the effect of the field cannot be lower than 3 kcal/mol with respect to the energy of the optimal BBP.

[2] The comparison between the energies of these two points takes into account the numerical error of the quantum chemical calculations

7 How to Cite

8 Contacts

If something is unclear or for any kind of question you are encouraged to write to the maintainer of this repository (marco.severi6@unibo.it) or to post in the Discussions section of the MANULS Github repository (https://github.com/MSeveri96/MANULS). Every comment, question or suggestion is more than welcome!

If you spot a bug in the code (thanks!) you can write to the maintainer or post in the Issues section of the MANULS Github repository.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[2] F. Neese, "Software update: The ORCA program system-Version 5.0," *WIREs Computational Molecular Science*, vol. 12, no. 5, e1606, 2022, ISSN: 1759-0884. DOI: 10.1002/wcms.1606. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1606.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[4] K. Ishida, K. Morokuma, and A. Komornicki, "The intrinsic reaction coordinate. An ab initio calculation for HNC→HCN and H+CH4→CH4+H," *J. Chem. Phys.*, vol. 66, no. 5, pp. 2153–2156, Mar. 1977, ISSN: 0021-9606. DOI: 10.1063/1.434152. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.434152.