OpenPath mini-help

P. Fleurat-Lessard, P. Dayal
Laboratoire de Chimie de l'ENS Lyon, 46 allée d'Italie, F-69364 Lyon Cedex 7

Feb. 2011

1 introduction

`OpenPath` is a program that can optimize a reaction path between two structures. The algorithm to optimize the path is close to the string method. The originality of this program lies in the coordinate set it can use to generate and optimize the path.

This program is an independant program that calls standard electronic structure codes to get the energies and forces it needs to optimize the reaction path. For now, it is coupled to Gaussian, MOPAC2009, Vasp and Turbomole.

2 Installation

We suppose here that you will install `OpenPath` into a directory called `OpenPath`. First, create the directory:

```
mkdir OpenPath
cd OpenPath
mv ../OpenPath.tgz .
```

Uncompress the archive:

```
gunzip OpenPath.tgz
tar -xvf OpenPath.tar
```

You should now have this `Mini_help.pdf` file and 4 directories (doc, src, utils, examples).

3 Compilation

Go to the directory in which you have uncompressed the files. Change to the `src` directory. Edit the `Makefile` to change the `Machine` description according to the compiler you want to use. Main choices are: gfortran, g95, ifort, pgf, xlf and pathscale for now. You might also have to check that the locations of the libraries are ok. Type `make`. You should now have a file called `Path.exe` in this directory, as well as two utilities called `xyz2scan` and `xyz2path` located in the `utils` directories. You should copy all these executables to your `/bin` directory (or any place from which they can be executed).

4 Use

4.1 Path calculation

To call `OpenPath`, you can type: `Path.exe Input_file Output_file` The input file `Input_file` is based on a namelist and looks like:

```
 &path
 nat=3, ! Number of atoms
 ngeomi=3, ! Number of initial geometries
 ngeomf=12, !Number of geometries along the path
 OptReac=.T., ! Do you want to optimize the reactants ?
 OptProd=.T., ! Optimize the products
 coord='zmat', ! We use Z-matrix coordinates
 maxcyc=31, ! Max number of iterations
 IReparam=2,! re-distribution of points along the path every 2 iterations
 ISpline=50, ! Start using spline interpolation at iteration 50
 Hinv=.T. , ! Use inverse of the Hessian internally (default: T)
 MW=T, ! Works in Mass Weighted coordiante (default T)
 PathName='Path_HCN_zmat_test', ! Name of the file used for path outputs
 prog='gaussian',! we use G03 to get energy and gradients
 SMax=0.1 ! Displacement cannot exceed 0.1 atomic units (or mass weighted at. unit)
 /
  3
 Energy :      0.04937364
 H     0.0000     0.0000     0.0340
 C     0.0000     0.0000     1.1030
 N     0.0000     0.0000     2.2631
  3
 Energy :      0.04937364
 H     0.0000     1.1000     1.1030
 C     0.0000     0.0000     1.1030
 N     0.0000     0.0000     2.2631
3
 CNH
H 0.000000    0.000000    3.3
C 0.000000    0.000000    1.1
N 0.000000    0.000000    2.26
%chk=Test
#P  AM1 FORCE

 HCN est bien

0,1
H 0.000000    0.000000    0.000000
C 0.000000    0.000000    1.000
N 0.000000    0.000000    3.00
```

Compulsory variables are:

NGeomi: Number of geometries defining the Initial path

NGeomf: Number of geometries defining the Final path

Nat : Number of atoms

Other options

Input: string that indicates the type of the input geometries. Accepted values are: Cart (or Xmol or Xyz) or Vasp

Prog: string that indicates the program that will be used for energy and gradient calculations. Accepted values are: Gaussian, Vasp or Ext.
In case of a Gaussian calculations, input must be set to Cart. One example of a gaussian input should be added at the end of the input file.See example file `Test_HCN_zmat_g03.path`.
In the case of a VASP calculation, if input is set to Cart, then the preamble of a VASP calculation should be added at the end of the input file. See example file `Test_VASP_cart.path`.
In the case of a VASP calculation, one should also give value of the `RunMode` variable .

RunMode: This indicates wether `OpenPath` should use VASP routine to calculate the energy and gradient of the whole path or not. User has two options. One is to calculate the energy and gradient of each point sequentially. This is usefull when running on one (or two) processors. In this case, `RunMode` should be put to `SERIAL`.When running in parallel with 8 or more processors, one can use VASP to calculate simultaneously the energies and gradients for all points, as in a normal NEB calculation. In this case, `RunMode` must be set to `PARA`.

ProgExe: Name (with full path) of the executable to be used to get energies and gradients. For example, if VASP is used in parallel, one might have something like:
`ProgExe='/usr/local/mpich/bin/mpirun -machinefile machine -np 8 ~/bin/VASP_46'`.
Another option that I use, is to put `ProgExe='./run_vasp'` and I put every option needed to run VASP into the `run_vasp` file.

PathName: Prefix used to save the path. Default is Path

Poscar: string that will be used as the prefix for the different POSCAR files in a VASP calculations. Usefull only if PathOnly=.TRUE., not used for internal calculations.

IGeomRef: Index of the geometry used to construct the internal coordinates. Valid only for Coord=Zmat, Hybrid or Mixed

Fact: REAL used to define if two atoms are linked. If $d(A,B) \leq fact * (rcov(A) + rcov(B))$, then A and B are considered Linked.

debugFile: Name of the file that indicates which subroutine should print debug info.

Coord: System of coordinates to use. Possible choices are: - CART (or Xyz): works in cartesian - Zmat: works in internal coordinates (Zmat) - Mixed: frozen atoms, as well as atoms defined by the 'cart' array(see below) are describe in CARTESIAN, whereas the others are described in Zmat - Baker: use of Baker coordinates, also called delocalized internal coordinates - Hybrid: geometries are described in zmat, but the gradient are used in cartesian

Step_method: method to compute the step for optimizing a geometry; choices are: - RFO: Rational function optimization - GDIIS: Geometry optimization using direct inversion in the iterative supspace

HesUpd: method to update the hessian. By default, it is Murtagh-Sargent Except for geometry optimization where it is BFGS.

MaxCyc: maximum number of iterations for the path optimization

Smax: Maximum length of a step during path optimization

SThresh: Step Threshold to consider that the path is stationary

GThresh: Gradient Threshold to consider that the path is stationary, only orthogonal part is taken

FTan: We moving the path, this gives the proportion of the displacement tangent to the path that is kept. FTan=1. corresponds to the full displacement, whereas FTan=0. gives a displacement orthogonal to the path.

IReparam: The path is not reparameterised at each iteration. This gives the frequency of reparameterization.

ISpline: By default, a linear interpolation is used to generate the path. This option indicates the first step where spline interpolation is used.

Arrays:

Rcov: Array containing the covalent radii of the first 80 elements. You can modify it using, `rcov(6)=0.8`.

Mass: Array containing the atomic mass of the first 80 elements.

AtTypes: Name of the different atoms used in a VASP calculations. If not given, Path will read the POTCAR file.

Flags:

MW: Flag. True if one wants to work in Mass Weighted coordinates. Default=.TRUE.

Renum: Flag. True if one wants to reoder the atoms in the initial order. default is .TRUE. most of the time.

OptProd: True if one wants to optimize the geometry of the products.

OptReac: True if one wants to optimize the geometry of the reactants.

PathOnly:TRUE if one wants to generate the initial path, and stops.

Hinv: if True, then Hessian inversed is used.

IniHup: if True, then Hessian inverse is extrapolated using the initial path calculations.

HupNeighbour: if True, then Hessian inverse is extrapolated using the neighbouring points of the path.

FFrozen: True if one wants to freeze the positions of some atoms. If True, a `&frozenlist` namelist containing the list of frozen atoms must be given. If VASP is used, and frozen is not given here, the program will use the F flags of the input geometry

FCart: True if one wants to describe some atoms using cartesian coordinates. *** Only used in 'mixed' calculations. *** If True, a `&cartlist` namelist containing the list of cart atoms must be given. By default, only frozen atoms are described in cartesian coordinates.

Autocart: True if you want to let the program choosing the cartesian atoms.

VMD: TRUE if you want to use VMD to look at the Path. Used only for VASP for now

More to come... have a look at the files provided in the `examples` directory. In particular, you will find there three directories containing easy-to-use examples. That is, each directory contains all the files you need to launch the application. For now, you have working examples for:

1. `Gaussian`: Examples using sequential call to Gaussian to calculate the energies and forces along the path.

2. `VASP`: Examples using to VASP to calculate the energies and forces along the path. As VASP can perform NEB calculations, Path can use it in two ways: Serial or Parallel.

   Serial In the Serial mode, Path uses Vasp to compute the energy and forces of each image separately (as it does for Gaussian for example). Therefore, the INCAR file should not contain any references to the number of images (no IMAGES command!).

   Parallel In the Parallel mode, Path uses VASP to compute at once the energies and forces for all the images (as VASP does for a NEB calculation). Therefore, the INCAR file MUST contain the IMAGES command. More, the directories 00, 01, ... should exist.

   Home or Scratch ? On our local cluster, it is adviced to perform the calculations on the /scratch directory that is local rather than on the /home that is mounted via NFS and is thus slow. However, when doing so, we sometimes had troubles with VASP, and discovered that all files should be copied on the /scratch of all machines. Therefore, the scripts are a bit tricky. In case your computer center is similar to ours, we provide 2 SGE scripts:

   run_Path_SGE_home performs the VASP calculations in the /home directory

   run_Path_SGE_scratch creates the directories, copies the files and performs the VASP calculations in the local /scratch directories.

3. `Mopac`: Examples using sequential call to MOPAC2009 to calculate the energies and forces along the path.

4. `Test`: Examples using the analytical HCN potential energy surface to calculate the energies and forces along the path. As this is fast, we also provide other Analysis tools. See README files in the Cart and Zmat subdirectories.

4.2 Path analysis

In order to analyse the path evolution, we provide some utilities, in the `utils` directory. `xyz2scan` and `xyz2path` have to be compiled. For this, go to either the `src` or the `utils` directory and type `make utils`. Here is a brief description of these utilities.

- `AnaPath` and/or `AnaPathref`
  These scripts analyse a calculated path. They use `xyz2path` to convert the cartesian coordinates saved by `OpenPath` into a data file (called PathName.datl) that be plotted using the gnuplot files that are also created. They all plot the path energy but in different ways:

  PathName_l.gplot plots the energy of the first iteration (the initial path) together with the energy of the following iterations, but one at a time.

  PathName_l2.gplot plots the energy of the path for all iterations, with respect to the curvilinear distance along the path.

  PathName_l3.gplot plots the energy of the path for all iterations, with respect to the index of the images.

- `xyz2path` and `xyz2scan` convert a bunch of cartesian coordinates into a data file. They are very similar: the only difference is the way they print their results. For `xyz2scan` the analyses are indiced using the geometry number whereas `xyz2path` computes the mass weighted distance between two geometries and uses this distance to index the results.
  They both use a file called 'list' to perform the analysis, which has the following structure: each line contains the type of the value you want to follow, it can be:

  b for a Bond distance

  a for an angle

d for a dihedral

c to create a center of mass

This descriptor is followed by the number of the atoms involved. The exception, is the `c` command that is followed by the number of atoms used to define this center of mass, and then the index of the atoms. A typical file can be:

```
b  1  2
b 2  3
a 1 2 3
c  2 1 2   <- create a new atom located at the middle of the 1-2 bond.
```

The cartesian geometries have to follow the XMol format. If the comment line contains `E=` then `xyz2scan` and `xyz2path` will read the following number and take it as the image energy.