A.V.E.D.A.

Automated Variable Electric Field DFT Application

User Manual

Contents

Technical Details **1**

1.1 Program Dependencies 1

1.2 General 1

Application Workflow **1**

2.1 Start.sh 1

2.2 Alignment 2

2.3 CartesianSP 3

2.4 ZMatrixSP 3

2.5 DipoleCalculation 4

2.6 ElectricFieldOpt 5

2.7 Results 5

Technical Details

1.1 Program Dependencies

— A.V.E.D.A. is only compatible with a Slurm compute cluster organization as written

— Gaussian 16 accessed through BlueHive/ Slurm

— Python 3.6 is used for text formating as well as vector algebra

— Python Packages:

- PyMol, Math, Sys, Time, Os, Numpy, Argparse, Scipy, Csv, Matplotlib

1.2 General

— Avogadro 1.2.0 used for structure generation and manipulation

— GaussView 6.0.16 used result analysis

— Structures visualized and presented with Chimera 1.15 and CYLview20

Application Workflow

2.1 Start.sh

— Prior to a computation, both intermediate and transition state .xyz files as well as the *start.sh* and Program folder must all be contained in the same folder. A user must cd into the specified directory prior to running the computation.

— Atoms in the intermediate and transition state .xyz files must be ordered consistently such that corresponding atoms in each structure are in the same order in the geometry file.

— With each new A.V.E.D.A. submission, a copy of all program files will be made specific to the job. These files are held in a folder titled by the *ComputationName* provided by the user. All Gaussian computations and script calculations are performed in this new folder.

— Running:

```
   sh ./start.sh ts_filename.xyz int_filename.xyz charge multiplicity
      functional basis_set [atom_reordering_method (0, 1, or 2)]
      computation_name number_of_processors SLURM_partition_name
```

begins an instance of A.V.E.D.A. and is the only user interaction required

— The *start.sh* script creates a directory to hold user specification as well as passes the input xyz files to the new job specific folder. Submission scripts are generated by calling *SubScripter.sh* which makes transition state and intermediate optimization scripts. These are then submitted through *batchSubmit.sh*. Individual Slurm submission scripts are generated by *templateGaussian.sh* and submitted to gaussian by *submitGaussian.sh*.

2.2 Alignment

— After each successful geometry optimizations the outputs are sent to the *2_alignment* directory and *2_alignment_boss.sh* is called. Only if two .out files are in the directory does alignment proceed, otherwise there is no action from the script.

— The PyMol library is loaded and *alignPyMol.py* is called to align the intermediate to transition state in space. After alignment with PyMol, this script reorders atoms for optimal field optimization in accordance with the user specified choice 0, 1 or 2 as described below.

0. *Method 0* may be selected to preserve the ordering they input affected only by Gaussian's *newzmat* utility's construction of the Z-matrix.
1. *Method 1* was developed to maximize the stability of the orientation atoms in an electric field by calculating the unweighted Cartesian center of the transition state and moving the nearest three atoms to the orientation atom position. It is predicted these core atoms will have the least freedom for low energy free rotation during optimization.

2. *Method 2* reorders atoms to minimize the geometry changes' influence on dipole-field orientation by choosing the atom furthest from the site of transformation where it is predicted the most dramatic changes will take place. Rather than moving individual atoms, this remote atom and all subsequent atoms are moved in a block to the top of the input file so that connectivity represented by atom order is preserved as much as possible.

— The new TS and Int ordered .xyz files are saved. Copies of both .xyz files are put into the folders *3_cartesianSP* and *4_zMatrixSP*.

— After alignment is done, *3_cartesianSP_boss.sh* and *4_zMatrixSP_boss.sh* are called.

2.3 CartesianSP

— Submission scripts for single point energy calculations are generated from the reordered .xyz files in cartesian coordinates by *SpSubmissionScripter.sh*. These single point calculations will also calculate the molecular dipole moment in the cartesian coordinate system.

— Both submission files are submitted with *batch4Sp.sh*, *submit4Sp.sh*, and *template4Sp.sh*

— The outputs are copied into the directory *5_dipoleCalculation* and *5_dipoleCalculation_boss.sh* is called.

2.4 ZMatrixSP

— First, both Cartesian geometries are converted to a z-matrix orientation for field optimization with the *newzmat* the Gaussian utility. This software is accessed and the output is formated with *newzmatRunner.sh*. Atom 1 is placed on the origin, Atom 2 on z-axis, Atom 3 (w.r.t. to DOF) is placed on the y-axis. This locked orientation ensures minimal movement w.r.t. the E-Field during optimization. The software may reorder atoms in the z-matrix slightly as some orderings are not compatible with a z-matrix.

— The z-matrix structures are copied to *6_electricFieldOpt*

— *ZMatReorientedSpSubmissionScripter.sh* is used to make a z-matrix single point energy calculation for both intermediate and ts.

— Both submission files are submitted with *batch4SpZmat.sh*, *submit4SpZmat.sh*, and *template4SpZmat.sh*

— The outputs are copied into the directory *5_dipoleCalculation* and *5_dipoleCalculation_boss.sh* is called.

2.5 DipoleCalculation

— *5_dipoleCalculation_boss.sh* waits until all 4 single point outputs are in *5_dipoleCalculation*, before continuing.

— First, the calculated dipole moments of both intermediate and transition state in cartesian and z-matrix coordinates are extracted from the output files. These results are passed as arguments to the python script *dipoleOrienter.py*.

— The logic of *dipoleOrienter.py* is as follows:

  - The goal of A.V.E.D.A. is to apply an electric field along the direction with the greatest change of stabilizing the transformation from (or difference between) intermediate to transition state.
  - This direction is along the negative difference (TS-INT) of the two structures dipole moments when aligned to each other.[1]
  - Intermediate and TS are aligned in Cartesian coordinates but not necessarily in the z-Matrix coordinate system.
  - Electric Fields in Gaussian may only be reasonably applied in the z-matrix system to avoid structure reorientation during optimization.
  - Therefore, a rotation matrix is calculated from the ts_{cart} to ts_{zmat} and int_{cart} to int_{zmat} to quantify the rotation between the two coordinate systems.
  - The $Diff_{cart} = TS_{cart}$ - Int_{cart} dipole difference vector is calculated in Cartesian coordinates.

  - The $Diff_{cart}$ vector is multiplied by each respective rotation matrix to project it into the z-matrix coordinate system yielding $Diff_{ts,zmat}$ and $Diff_{int,zmat}$
  - An electric field may be applied along these individual vectors while maintaining a physical relation to the transformation of interest.

— The normalized difference vector is scaled by $(-25,\ -50,\ -75,\ -100) \cdot 10^{-4}$ $a.u.$ to be applied in succession to the intermediate and transition state. These 3D vectors are exported for the Int and TS.

— The dipole text files are copied to the *6_electricFieldOpt* directory and *6_electricFieldOpt_boss.sh* is called.

2.6 ElectricFieldOpt

— *eOptsubmissioncripter.sh* uses the z-matrix orientation of both ts and intermediate to generate an optimization submission with a field strength of $-25 \cdot 10^{-4}$ $a.u.$ these submission files are submitted with *batch4eOpt.sh*, *submit4eOpt.sh*, and *template4eOpt.sh*.

— Next submissions are generated to start from a checkpoint file for field strengths $(-50,\ -75,\ -100) \cdot 10^{-4}$ $a.u.$ with *checkSubScripter.sh*. Folders are made for these submissions with *makeCheckDirs.sh* and *template4eOpt.sh* but jobs are not submitted.

— *analyzeGaussian.sh* and *analyzeInitalOpt.sh* are moved into all submission folders.

— Once optimization is complete at each field strength, the checkpoint file is moved into the next strongest field submission folder for Int and TS folders respectfully and the jobs are submitted using the scripts generated with *template4eOpt.sh*.

— After all optimizations the .out files are moved into *7_results* and *7_Results_boss.sh* is called.

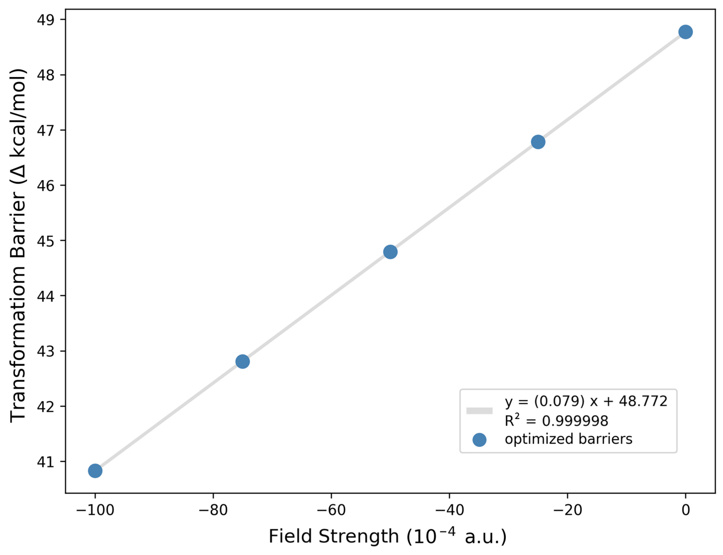
2.7 Results

— When all 6 .out files from optimizations at field strengths of $(0,\ -50,\ -75,\ -100) \cdot 10^{-4}$ $a.u.$ are in *7_results*, *7_Results_boss.sh* runs.

— The final energy of each structure is copied to its own text file. These values are passed as arguments to *resultsFormater.py*.

— The python script first converts all energies from Hartree to kcal/mol. Barriers are calculated by subtracting $\Delta E = E_{TS} - E_{Int}$ at each field strength. The net benefit by each variable field strength is calculated by comparing $\Delta\Delta E = \Delta E_{field} - \Delta E_{no\,field}$.

— These data are formatted into a .csv and export plot with matplotlib for user analysis.



ts dipole moment (xyz)

[0.388, 1.242, 0.696]

||ts dipole|| = 1.475

int dipole moment (xyz)

[0.236, -1.892, -0.077]

||int dipole|| = 1.908

ts-int dipole moment (xyz)

[0.152, 3.134, 0.773]

||dipole difference|| = 3.231

results summary

field strength (10^{-4} a.u.)	0	-25	-50	-75	-100
int energy (a.u.)	-210.2534	-210.2518	-210.2506	-210.2497	-210.2491
ts energy (a.u.)	-210.1757	-210.1773	-210.1792	-210.1814	-210.1841
barrier ΔE (kcal/mol)	48.78	46.78	44.79	42.81	40.83
ΔΔE (kcal/mol)	0.00	-1.99	-3.98	-5.97	-7.95

References

[1]https://doi.org/10.1021/jacs.8b08233