CIOPT Tutorial

8/12/2007

by Benjamin G. Levine and Todd J. Martínez

1 Introduction

This document is intended to take the reader step by step through running the CIOPT software package. CIOPT is a freely downloadable program for the optimization of minimal energy conical intersections (MECIs). It uses the conical intersection optimization scheme described in B. G. Levine, J. D. Coe, and T. J. Martínez, submitted to J. Phys. Chem. A (referred to as LCM.) This method is unique in that it does not require the calculation of nonadiabatic coupling matrix elements. Only energy information is required. This makes this scheme more generally applicable than other conical intersection optimization schemes. Energy gradient information is not required, but if analytic gradients are available they significantly improve convergence.

CIOPT does not perform any electronic structure calculations of its own. Instead it can be interfaced with the electronic structure package of the user's choice via a template based scheme. When an electronic structure calculation is required CIOPT writes the appropriate input file, calls the electronic structure package, and writes the output.

CIOPT includes several advanced features, not available in other software packages. CIOPT contains an algorithm for the optimization of minimal energy intersections of three states (ME3CI). It also contains an algorithm for the optimization of the intersection closest to a reference geometry in weighted Cartesian coordinates (minimal distance conical intersections, or MDCIs). Like the MECI optimization scheme, these schemes do not require nonadiabatic coupling matrix elements. Energy gradient information speeds convergence but is not required.

This document is intended to introduce the user to the input structure, template scheme, and output structure of CIOpt. Instructions to compile CIOpt are in section E.2. The general structure of the input is briefly described in section E.3. Section E.4 illustrates the steps necessary to run a simple MECI optimization using Multistate Complete Active Space Second Order Perturbation Theory (MS-CASPT2) as implemented in the MolPro molecular electronic structure package. (Note that MolPro is not required to run CIOpt in general, but this tutorial uses it for examples.) Section E.5 describes a variety of sample jobs distributed with this code utilizing different optimization schemes and various electronic structure packages. Section E.6 give tips for dealing with troublesome optimizations.

This document is intend to help people get started using CIOpt as quickly as possible. It does not describe all options which can be used with CIOpt, only those which are immediately necessary. A more complete list of options can be found in the complementary document `ciopt.pdf`. If you have any comments on the CIOpt program or this documentation, or if you would like to contribute sample template files for programs/methods not included in this document feel free to contact us at ben@spawn.scs.uiuc.edu.

2 Installing CIOpt

Move the `ciopt.tgz` file to the desired directory. Change to that directory and run:

```
tar xvzf ciopt.tgz
cd CIOptRelease
```

```
make
```

Now, add the `CIOptRelease` directory to your PATH and CIOPT is ready to run.

3 General input structure

Each CIOPT job must be run in its own directory. The input to CIOPT is separated into several files. The `Control.dat` file contains options which define the optimization which CIOPT is to perform and the starting geometry of the molecule. Two or more `template.*` files instruct CIOPT how to run the electronic structure package. `template.write*` files allow CIOPT to write input decks, while `template.read*` files instruct CIOPT how to parse the output.

4 Setting up an MS-CASPT2 MECI optimization using MOLPRO

4.1 Sample input

Before setting up our own job, we will look at and run a sample job. This job is in `CIOptRelease/Tutorial/Ethylene-MolPro-MSPT2`. You will see 6 files. Here we will explain these files one at a time.

First look at the `Control.dat` file. You will see two sections in this file. In first section is the control block. This block must begin with the line **`&control`** and end with the line **`/`**. The control block contains all of the options which instruct CIOPT as to what kind of optimization you wish to perform. This job is set up for a typical conical intersection optimization taking advantage of analytic gradients. The options are described below.

`nopt=3`	**`nopt`** describes which optimization scheme CIOPT should use. **`nopt=3`** indicates that BFGS will be used. BFGS is the preferred optimization scheme if analytic gradients are available.
`natoms=6`	**`natoms`** indicates the number of atoms in the molecule. This job is to find an MECI in ethylene, which has 6 atoms.
`nstates=2`	**`nstates`** is set to the number of electronic states calculated. In this calculation we are only interested in S_1 and S_0, so **`nstates`**

	is set to 2.
`istate=2`	**`istate`** is index of the **higher** of the two electronic states between which we wish to find an intersection. States are numbered from the ground state, with the ground state being 1 (S_0 = 1, S_1 = 2, S_2 = 3, etc.) We wish to find an intersection between S_0 and S_1, thus we set **`istate=2`**, the index of S_1.
`nefunc=7`	**`nefunc`** tells CIOPT which energy function to optimize. **`nefunc=7`** indicates the two state energy functional in equation ??? of the LCM paper.
`crunstr=…`	**`crunstr`** is the command line instruction to run MOLPRO. **You will need to change this line prior to running this sample job.** CIOPT creates an input deck named "tmp.com" when it requires an electronic structure calculation. This string should be set to the command which will run this calculation on your machine.

The second section of the `Control.dat` file gives the initial geometry. The first column contains the symbols of the atoms, and the following three columns are the Cartesian coordinates of the atoms.

In addition to the `Control.dat` file, there are 5 template files. First we will consider the `template.write*` files. The `template.write*` files allow CIOPT to write input decks for the electronic structure code. CIOPT uses different `template.write*` files depending on what information it requires. Sometimes CIOPT requires energy and gradient information, in which case `template.writeg` is used. When CIOPT requires only energy information `template.write` is used instead.

Look at `template.writeg`. It is an input deck with CIOPT variables in place of a few numbers. In this case, the input deck is for MOLPRO. It performs a CASSCF calculation, followed by two MS-CASPT2 calculations. Each MS-CASPT2 calculation calculates the forces on a different state. Two gradients are required because two states are involved in the intersection. (The need to run two MS-CASPT2 calculations to get two gradients is inherent in MolPro, but may not be necessary in other electronic structure packages.)

Two different variables are present in the `template.writeg` file. The first are geometric variables, which take the form **`%%xxx`**, where **`xxx`** is the number of the geometric variable. Geometric variables are numbered as follows: atom 1 X coordinate is 001, atom 1 Y coordinate is 002, atom 1 Z coordinate is 003, atom 2 X coordinate is 004, etc.

The other variables are for state labels. These variables take the form **`%#ISTATE`** and **`%#JSTATE`**. These variables are used by CIOpt to ask for the gradient on specific states.

Now look at `template.write`, the template file for an energy calculation without gradients. You will notice that it is identical to `template.writeg`, except that it performs only one MS-CASPT2 calculation, and no calculations of forces. The **`%#xSTATE`** variables are never necessary in the `template.writeg`, because no gradient calculations are performed. It is very important when constructing `template.write` and `template.writeg` to make sure that they are consistent with one another and that the order of the atoms is consistent with that in `Control.dat`.

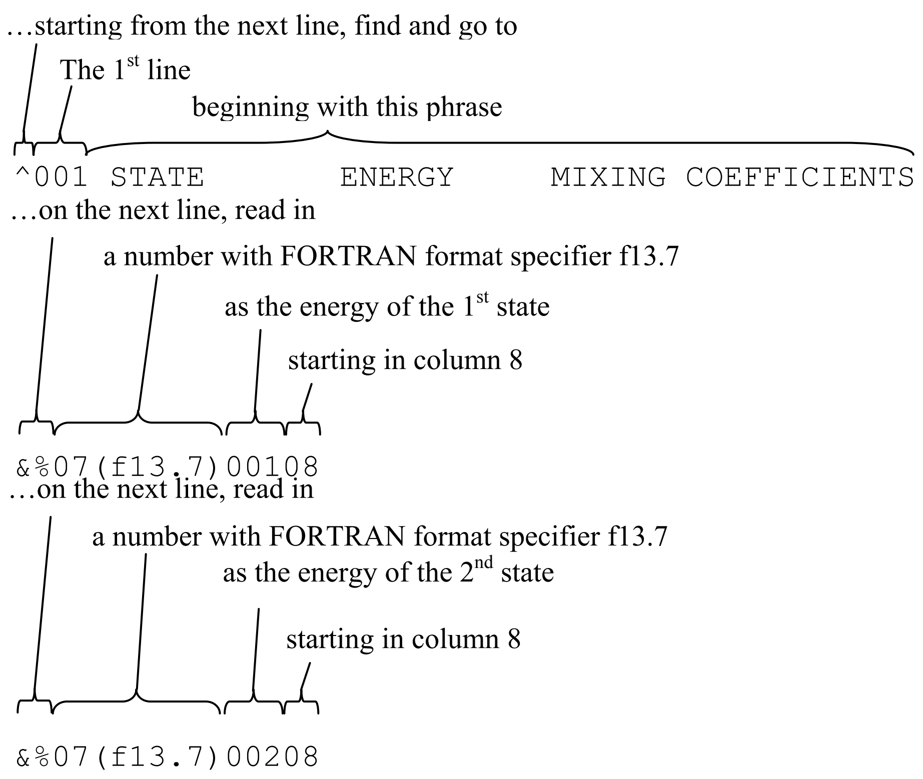
The `template.read*` files are a bit more complicated. Each `template.read*` file gives a set of instructions to CIOpt to parse and read the output of the electronic structure code. `template.read` reads the energy of the states, regardless of whether gradient information was asked for or not. `template.readg` reads the gradient information for ISTATE, and `template.readg2` reads the gradient information for JSTATE.

Each `template.read*` file is a set of instructions to CIOPT. They are read linearly from top to bottom like a computer program. Each instruction is performed from the line following the result of the previous instruction. The directives are listed in the table below.

`^iiistring`	Find iiith occurrence of string after start of line
`*iiistring`	Find iiith occurrence of string anywhere in a line Note that we only count one occurrence per line
`@iii`	Skip iii lines
`&%iiffffjjjkk%llggggmmmnn`	Using FORTRAN format string ffff of length ii, read par(j) starting at position kk, then do same for par(m) with format string gggg of length ll starting at position nn. In `template.read` par(j) is the energy of state j. In `template.readg` or `readg2` it is the jth element of the gradient.
`!string%iiffffjjjkk%llggggmmmnn`	Like above, but will search for line beginning with string **`^iiistring`**

Look at `template.read.` `The four directives are annotated here:`

$$\overbrace{\texttt{\^{}}}^{\text{Find and go to}}\overbrace{\texttt{002}}^{\text{The 2nd line}}\ \overbrace{\texttt{Heff Half Sum symmetrized}}^{\text{beginning with this phrase}}$$


…starting from the next line, find and go to
The 1st line
beginning with this phrase
…on the next line, read in
a number with FORTRAN format specifier f13.7
as the energy of the 1st state
starting in column 8
…on the next line, read in
a number with FORTRAN format specifier f13.7
as the energy of the 2nd state
starting in column 8

```
^001 STATE          ENERGY      MIXING COEFFICIENTS
&%07(f13.7)00108
&%07(f13.7)00208
```

CIOPT reads the energies from the output in a series for 4 steps:

Find the second instance of a line starting with " Heff Half Sum symmetrized"

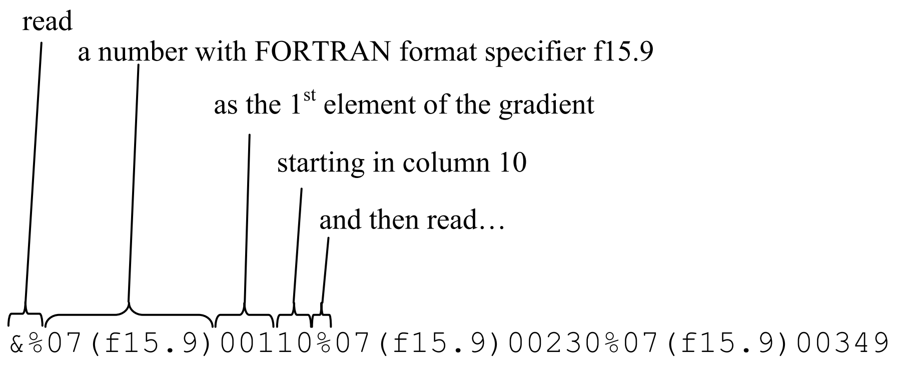
Starting from the next line, find the first instance of a line starting with " STATE…"

On the next line read the energy of state 1 (the ground state, **not** ISTATE) in FORTRAN format (f13.7) starting at position 8 (the either character of the line).

On the next line read the energy of state 2 in Fortran format (f13.7) starting at position 8.

If you are unfamiliar with FORTRAN format specification this website is useful: http://adc.gsfc.nasa.gov/adc/fortran.html

The `template.readg` and `template.readg2` files use the same language. The only difference is that par(j) now represents the jth element of the gradient, not the energy of state j. Look at `template.readg`. Notice that there are three `%` directives on each `&` line:


read
a number with FORTRAN format specifier f15.9
as the 1st element of the gradient
starting in column 10
and then read…

```
&%07(f15.9)00110%07(f15.9)00230%07(f15.9)00349
```

This allows CIOPT to read three elements of the gradient off of each line. `template.readg2` is essentially identical, except it finds the second gradient in the output file rather than the first.

4.2 Running CIOPT and reading the output

Now that we have looked at all of input it is time to run the job. This will take several minutes, as CIOPT must perform several MS-CASPT2 calculations. In the job directory run:

CIOpt.e&

This will run CIOPT in the background. This will probably take 5 or 10 minutes to run.

When CIOPT is finished there will be several output files. The most important are `CIOpt.out` and `iter.log`. `CIOpt.out` describes the progress of the optimization as it is occurring. The output is detailed, but most important is the output of the geometries for each iteration:

```
…
 Iteration      0 E=     -78.14395864 Cvg=       0.04284612
 Current Geometry
 C    0.0240000000    0.0800000000   -0.4500000000
 C   -0.3060000000   -0.1070000000    0.9580000000
 H    1.0830000000   -0.1580000000   -0.5830000000
 H   -0.1200000000    1.1060000000   -0.7890000000
 H   -0.5400000000   -0.5770000000   -1.1120000000
 H   -0.1410000000   -0.3430000000    1.9770000000
…
```

When optimization with a given λ is complete the final geometry will be output:

```
…
FINAL GEOMETRY, E=   -78.14668110, gap=     0.00375850
 C    0.0087809856    0.0813934024   -0.4349412192
 C   -0.3690106474   -0.0895968940    0.9492676647
 H    1.0758259009   -0.1661530726   -0.5325017617
 H   -0.1115097191    1.1077982567   -0.8005907895
 H   -0.5319271182   -0.5784320275   -1.1233130764
 H   -0.0721593998   -0.3540096675    1.9430791704
…
```

If the gap is sufficiently small to label the point an MECI (smaller than the **`cigap`** control block variable, .001 Hartree by default) the optimization is done. This is indicated by the line:

```
Sequential penalty converged, gap=…
```

Otherwise, λ is increased and another optimization is performed. This is indicated by the line:

```
Increasing penalty in seq penalty method…
```

`iter.log` allows the convergence of the simulation to be monitored. Look at `iter.log`. There is one row per iteration of the optimization. The number of the iteration is in column 1. Column 2 shows the current value of the objective function (LCM Eq. 3). This is followed by columns related to the convergence criteria: The decrease in the objective function relative to the previous iteration is in column 3 (see LCM Eq. 10). The total norm of the gradient of the objective function is in column 4. The norm of the component of the gradient in the gap direction is in column 5 (see LCM Eq. 11). The norm of the perpendicular component is in column 6 (see LCM eq. 12).

Monitoring columns 3, 5, and 6 lets you see how close to convergence you are. When λ is increased column 1 is reset to 0.

4.3 Modifying the input

Follow the following steps to run a conical intersection optimization for a different molecule: a propylene (methyl substituted ethylene) intersection.

1) Create a new directory in which to run the job and copy over the necessary input files (`Control.dat`, `template.*`). We will modify these files rather than starting from scratch.

2) Modify the `Control.dat` file. You will need to change the number of atoms, and the starting geometry. A decent starting geometry is obtained by removing the last hydrogen and add the following atoms:

```
C     -0.141           -0.343           2.300
H     -0.900           -0.400           2.700
H      0.200           -1.000           2.700
H      0.200            0.400           2.800
```

3) Modify the `template.write` and `template.writeg` files to reflect the new molecule. Modify the geometry section appropriately. Also, since there are now more electrons the occ, closed, and wf cards will need to be changed. Appropriate values are 13, 11, and 24 respectively. Don't forget to modify **all** such cards in **both** files, and to make sure the order of the atoms is the same as it is in the `Control.dat` file.

4) Since we are still only looking at 2 states, the `template.read` file needs no modification.

5) Modify the `template.readg` and `template.readg2` files. You will need to add additional **`&%`** lines to read in the elements of the gradient for the additional

atoms. These lines will be identical to the lines above, except for the values of jjj for **all three** elements per line.

6) Run the job as you did before. It will take longer than the ethylene job since the electronic structure calculations will take longer, and we are starting with an inferior initial guess of the geometry. (In general, it would probably be better to optimize with CASSCF first to refine our guess, before optimizing at the more expensive MS-CASPT2 level of theory. With larger molecules optimization with a poor guess would make the optimization unfeasible, but here it will just be slower than is necessary.) You can watch the progress of the job with:

   ```
   tail –f iter.log
   ```

   if you wish. This job may take an hour, but there is no need to run it to completion if the first several iterations finish without incident. (The result is interesting, though. Ground state propylene has C_s symmetry, but this intersection is of higher symmetry: C_{2v}!)

Now you know enough about CIOPT to set up your own MS-CASPT2 conical intersection optimizations in conjunction with MOLPRO, but CIOPT is much more general than this. Sample input for a variety of sample jobs (including MECI optimization jobs using CASSCF and MRCI in MOLPRO, and MS-CASPT2 in MOLCAS, as well a job using internal coordinates rather than Cartesian, an ME3CI optimization job, and an MDCI optimization job) are included in the `CIOptRelease/Tutorial` directory. The important differences between these jobs and the present job are described in section 5 of this document.

5 Sample jobs

5.1 CASSCF MECI optimization

As far as CIOPT is concerned, CASSCF optimizations and MS-CASPT2 optimizations performed with MOLPRO are identical. The only differences are in the `template.*` files.

5.2 MECI optimization without gradients (MRCI MECI optimization)

When analytic gradient calculations are not available at a particular level of theory the default optimizer (BFGS) can be run with numerical gradients, but this is not the optimum choice. The Powell conjugate direction scheme is more efficient, because it does not require gradient information. An example of such a calculation is given in the `Tutorial/Ethylene-Molpro-MRCI` directory. The Powell optimization scheme is chosen by setting **`nopt=1`** in the `Control.dat` file. It is also necessary to set **`zangrad=.false.`** to let CIOPT know that there are no analytic gradients available. When running without analytic gradients only `template.read` and `template.write` are needed. MOLPRO template files for MRCI calculations are given in the sample.

5.3 Optimization when each input deck can only calculate a single gradient

MOLPRO is capable of performing gradient calculations on multiple electronic states within a single input deck, but this is not the case in many other electronic structure packages. CIOPT is capable of dealing with this by calling the electronic structure code once for every state on which the gradient is needed. This is illustrated in `Tutorial/Ethylene-multigrad-false`. (MOLPRO is used in this example, even though it is never necessary to use **`multigrad=.false.`** with MOLPRO.) The need to run gradients on multiple states in separate input files is indicated by setting

`zmultigrad=.false.` in `Control.dat`. When this set, the `template.writeg` file should calculate only one gradient. The state index is replaced by the variable **`%$ISTATE`**. Only a `template.readg` file is used to read the gradient (not `template.readg2` file). Otherwise the input is identical to a typical MECI optimization.

5.4 MOLCAS MS-CASPT2 MECI optimization

An example calculation using MOLCAS is found in the `Tutorial/Ethylene-Molcas` directory. MOLCAS does not support MS-CASPT2 gradients, thus the Powell scheme described in the MRCI example is used (**`nopt=1`**). The only differences in the `Control.dat` file are in the **`crunstr`**, which now points to a script which runs MOLCAS, and in **`cinpdeck`** and **`coutfile`**, which rename the input deck and output file to be have extensions typical for MOLCAS input and output.

There are two additional file in the directory which are required by this calculation: `molcas2` and `JOBOLD`. `molcas2` is a script to run MOLCAS. You can replace this with your own script as long as you make the appropriate changes to the `Control.dat` file. `JOBOLD` is a MOLCAS restart file. It is necessary to make such a file before running CIOPT with MOLCAS if you wish to restart from the old wavefunction at each step.

`template.*` files are also modified to work with MOLCAS. These example MOLCAS template files may be useful to the MOLCAS users wishing to use CIOPT.

5.5 ME3CI optimization

A sample ME3CI optimization is in the `Tutorial/Butadiene-ME3CI` directory. ME3CI optimization is performed by optimizing the objective function in

LCM Eq. 14. In `Control.dat` this is called for by setting **`nefunc=10`**. The istate control block variable is set to the upper of the three states involved in the ME3CI. The `template.writeg` file now calculates three gradients. The index of the third state is represented by the **`%$KSTATE`** variable. A `template.readg3` file has been added which is responsible for reading the gradient of KSTATE.

5.6 MDCI optimization

A sample MDCI optimization is in the `Tutorial/Acrolene-MDCI` directory. MDCI optimizations are performed by optimizing the objective function in LCM eq. 15. In `Control.dat` this is called for by setting **`nefunc=11`**. A reference geometry is also set in the `Control.dat` file. It is input in the 3 * **`natoms`** dimensional vector **`rmsdgeo`**. The elements of the vector are in the following order: X component of atom 1, Y component of atom 1, Z component of atom 1, X component of atom 2, etc. The template files are identical to those used in a typical MECI optimization.

6 Troubleshooting

The algorithm implemented in CIOPT is generally quite stable, but still problems can arise when running CIOPT. If you have checked the template files and they are correct, read below for some tips on dealing with troublesome jobs.

The first question to ask when your job fails to converge is “why did my job fail?” In the majority of cases CIOPT optimizations fail because the electronic structure method is not stable in the region of the potential energy surface explored by CIOPT. The first thing to look at is your `tmp.out` file. This is the output of the last electronic structure calculation asked for by CIOPT. (This file may have a different name if you are running an electronic structure code other than MOLPRO., e.g. `tmp.log` in the MOLPRO example above.) Did this calculation converge?

If it did not, there are several possible reasons. Look at the geometry. Is it reasonable? (Look for bond lengths that are chemically appropriate.) If the geometry is unreasonable CIOPT took a step which was too large. If using BFGS (the default, **`nopt=3`**), this can be corrected by decreasing the **`hessinguess`** control block variable (this is the initial guess of the inverse Hessian diagonal elements for the BFGS optimizer; the default is .1d0).

If the geometry seems reasonable, then consider modifying the `template.write*` files to improve the convergence characteristics of the calculations. This could mean starting from a different initial guess, restarting from the previous wavefunction if you are not, or not restarting if you are. (Whether or not you are restarting from the previous wavefunction depends on how you design your template files.) You can also try using a different wavefunction optimization scheme (turning off the second order CASSCF optimization scheme implemented in MOLPRO can often help, though it does slow the calculations). If none of these options help consider that your choice of electronic structure method may be poor. If using a multi reference method, reconsider the active space. Not all active spaces can describe the entire PES accurately.

If convergence of the electronic structure method was not the problem, there are also ways to improve the results. If CIOPT seems not to be converging consider finding a better initial guess. Using an intersection optimized at a low level of theory as a starting guess is often fruitful.

If this does not work, or you are already using a good guess changing the parameters of the simulation can be beneficial. Again, **`hessinguess`** is a good place to start. If this does not improve the situation try changing **`alpha`** (the smoothing parameter in LCM Eq. 3) or **`dlambdagap`** (the initial value of the penalty multiplier in the same eq.).

set 10.0 will be the best choice