Carlier Group
Gaussian User Manual

Version 1.24
January 13, 2012

Paul Carlier, Nipa Deora, Polo Lam, Jason Harmon,
Larry Williams, and Neeraj Patwardhan

*Please send your comments and corrections to Paul Carlier (pcarlier@vt.edu).
Thank you!*

Please note—not every section of this manual has been updated to reflect changes in Gaussian '09 from Gaussian '03. Virginia Tech Users--please let me know if something does not work as expected.

This manual assumes that the reader has a good grasp of the basic concepts and terminology of computational chemistry. *Spartan '10* is a very friendly, windows-driven computational chemistry package and provides an excellent learning environment to start doing quantum chemical calculations. The *Spartan '10* manual is online only but within the program you will also find excellent tutorials and a link to a pdf version of "A Guide to Molecular Mechanics and Quantum Chemical Calculations." If you haven't worked through the tutorials, or seen the new on-line help, please do. In addition, the following books can be borrowed from Paul's library:

> *A Brief Guide to Molecular Mechanics and Quantum Chemical Calculations* (Hehre,Yu,Klunzinger,Lou): this book provides a good description of basis sets and computational methods.
>
> *Computational Chemistry: A Practical Guide for Applying Techniques to Real World Problems* (David Young): this book provides an introduction to several topics not covered in the above book (e.g. TS opt algorithms, Z-matrices, conformational search algorithms)
>
> *Exploring Chemistry with Electronic Structure Methods* (Foreman & Frisch): this book is designed to be used with Gaussian and gives worked out examples of how to do many types of calculations. The section on solvent effects (SCRF) is especially helpful.

Essentials of Computational Chemistry (C. Cramer): very nice description of different computational methods, discusses thermodynamics in great detail.

Introduction to Computational Chemistry (Jensen): At present this is our most detailed reference on the subject. Very thorough and has a lot of math, for those that like that sort of thing.

A Chemist's Guide to Density Functional Theory (Koch & Holthausen): A great book covering the basics of density functional theory. Quite theoretical and discusses the different types of density functionals.

Introduction to Quantum Mechanics in Chemistry (Ratner & Schatz): A great book detailing the basics and the theory behind quantum chemistry.

Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory (Szabo & Ostlund): An terrific book detailing the theory behind ab initio methods. Very good reference for Hartree-Fock theory (we do not have this book at present: it is available in the library ISBN 0029497108)

To use Gaussian successfully on Inferno2, you will need to learn the basics of the Unix operating system. In the group we have three books that may be helpful to you:

Unix-Visual Quickstart Guide (Jay & Jay): this is one of my favorite unix resources. Start here.

Teach Yourself Unix in 24 Hours 3rd Ed. (Taylor)—this is my other favorite unix book. Particularly good sections are those on permissions and filters and pipes, and handling the data they generate (section on .awk).

Unix for Dummies (Levine & Levine Young): I find the style of writing annoying, but the section on Text editors is pretty good—you will find a bunch of useful hints for vi here.

Nipa Deora also recommends *google* to find the answers to your questions. A brief summary of the basic commands and concepts you will need to master is given in Section 01. below.

01. Unix General

A. File management

You will need to master these basic commands. We recommend creating a 'run' directory in which you can store all your current calculations. As these finish, move them to a 'store' directory and delete the temporary files. Consult Section 09. 'Archiving Calculations' below for further tips.

Basic commands:

```
mkdir to create directory
rm to delete file
mv to move file
cp to copy file
```

Directory symbols:

```
a single period . means the current directory
two periods .. means the directory above,
```

The two period symbol .. is very useful for moving around and for copying/moving files with relative pathnames (see below).

B. Pathnames

Absolute pathnames start with / because this symbolizes the root directory
relative pathnames start without a / --that is the key difference

To move a file to a directory in another branch, you must pay attention to this distinction. If you want to use an absolute pathname you must start with /home. If you want to use relative pathnames, you can use the .. symbol to go up a level. For example

ex. file in run, want to move to store, both run and store directories are in pcarlier

from the run directory type

```
mv filename.gjf /home/pcarlier/store/
```

or you can use a relative pathname

```
mv filename.gjf ../store/
```

C. Text Editors

You will need to use a text editor to create and modify your g09 input files, and to search your output files. As far as I know, *vi* (or *vim*) is the only editor available on Inferno2. This is the least friendly, but most powerful editor in Unix. Consult our Unix books for how to use it. If you choose vim, here are some helpful hints:

1. Vim is a modal editor. On launch you are in “command mode.” To get to “insert mode,” type “i”. Now you can type or delete text as you would in a normal word processor. To get back to command mode, hit the “esc” key.

2. In command mode, a few helpful commands to get you started are

to save your work, type ":w"
to exit if no changes have been made since the last save, type "q"
to exit without saving your work, type ":q!"
to save your work and exit in one step, type ":ZZ" (note you have to hold down "shift"!)

3. If you don't know what mode you are in, hit "esc" two or three times—you will here a beep to let you know you are back in command mode.

Lots more vim commands will be found through this manual, but I also recommend p. 135-136 of *Unix for Dummies* and p.78-85 of *Unix-Visual Quickstart Guide*.

D. Helpful Unix commands

To view a file on screen

```
cat filename.out
```

To see a file screen by screen

```
more filename.out
```

To see the beginning screen of a file

```
head filename.out
```

To see the ending screen of a file

```
tail filename.out
```

This is useful when calculations fail, since you go right to the error message

To look at the end of the file as it is being generated:

```
tail -f filename.out
```

This command is useful if you want to look at the output file as it is being written, for example to be sure that your calculation gets started right (cntrl-C to get out).

To look at a file screen by screen

```
more filename.out
```

Just type 'q' to get out. Alternatively, the less command is also very useful:

```
less  filename.out
```

With this command one can scroll up and down instead of just down with the more command)
While using *more* or *less*, you can skip ahead to the search term you desire by typing

```
/search term
```

You can also feed the input of a certain command to more, such as

```
ls *.gjf | more
```

The vertical line is called a 'pipe' and it takes the output of the first command ls *.gjf and displays it screen by screen. In this case you will get a listing of all the .gjf files in the directory.

To look for a particular word or multi-word phrase in a file

```
egrep word filename.out
egrep 'multiword phrase' filename.out
```

These commands will look for the line containing the searchterm within a file and print it out. You need the single quoted only for the multiword phrase search. This is a fantastic command that you will use all the time. Note the egrep and grep are very similar and in most cases can be used interchangeably.

E. Permissions

Note: as described below in 1.I below, we have a new group directory at /home/PRC. I highly recommend use of this directory to share files within our group. Permissions are set to make this filesharing as seamless as possible. Nevertheless, if you are interested to learn about how to set permissions in your own directory, please read this section.

To gain a better understanding of this important subject I recommend that you refer to *Unix for Dummies* or *Unix-A Visual Quickstart Guide*.

If you want to allow everyone to view and copy files from a folder in your directory, go to your home directory and type:

```
chmod 755 .
```

The string 755 defines permissions and will be explained below. The period signifies the current directory and this will change the accessibility on your home directory to make the contents viewable, then, go to the public directory (e.g. pub)

```
cd pub
chmod 755 .
```

This will make the pub directory accessible. Then type

```
chmod 755 *.*
```

This makes all the files in the pub directory accessible to the world.

To make the pub directory and all the files within it accessible only to the owner and the group, repeat these steps with the 750 as the permission code.

By default the group classification will be the same as the owner. To change the group classification, the group must be created by the system administrator. Once that group membership is defined, you can change the group classification of your directory by

```
chgrp prc cyclopropyl
```

Now the directory cyclopropyl has a group classification of prc (defined below).

To see permissions and group classifications within a directory, use the ls –l command. Here is an example from my account.

```
pcarlier@inferno2:~> ls -l
total 9760
drwxr-x---  2 pcarlier pcarlier    4096 2005-04-04 10:44 amide
drwxr-x---  2 pcarlier pcarlier    8192 2005-06-08 20:30 bzd
drwxr-x---  2 pcarlier prc         4096 2005-03-10 09:43 cyclopropyl
drwxr-x---  2 pcarlier pcarlier   12288 2005-06-08 18:38 dimer
drwxr-x---  2 pcarlier prc         4096 2005-05-19 00:01 epoxide
drwxr-x---  2 pcarlier pcarlier      67 2005-04-27 09:54 mail
drwxr-x---  2 pcarlier pcarlier    4096 2005-06-21 17:34 misc
drwxr-xr-x  5 pcarlier pcarlier   12288 2006-09-07 08:05 molden4.4
-rw-r--r--  1 pcarlier prc      9861120 2006-07-07 13:56 molden4.4.tar
drwxr-x---  3 pcarlier pcarlier      18 2005-06-11 00:44 polo
drwxr-xr-x  2 pcarlier prc         4096 2006-10-24 15:23 pub
-rwxrw-r--  1 pcarlier pcarlier      95 2005-05-18 00:05 recent
-rwxrw-r--  1 pcarlier pcarlier     105 2005-05-18 00:05 recently
drwxr-x---  2 pcarlier pcarlier   12288 2007-06-24 22:05 run
drwxr-x---  5 pcarlier pcarlier    8192 2005-05-03 15:24 store
drwxr-x---  2 pcarlier pcarlier    8192 2005-05-13 07:07 tacrine
drwxrwxr-x  2 pcarlier pcarlier    4096 2005-06-21 17:29 test
-rw-r--r--  1 pcarlier pcarlier      25 2006-08-04 21:21 testfile
drwxr-x---  2 pcarlier prc          146 2005-05-23 22:04 tools
-rwxr-x---  1 pcarlier pcarlier       2 2005-05-02 21:26 #tryit#
```

Permissions are listed in the first column in a string of 10 characters.
a)The first character is d, if it is a directory—if it is a file, a hyphen will appear.
b)The remaining 9 characters describe the permissions for the file/directory in question: there are three user classifications (owner, group, world) and three types of permission (read, write, execute). Characters 2-4 pertain to the owner, 5-7 to the group, and 8-10 to the world. Each of these classifications are important. Note that as a default the

group membership is the same as the owner. For our convenience, we have defined the group "prc"—contact Paul if you are not yet a member of this group.

a)for the directory **run**
the owner (pcarlier) has read, write, and execute permissions;
the group (pcarlier) has read and execute permissions;
the world does not have permission to move to this directory (via a cd command).

b)for the directory **cyclopropyl**:
the owner (pcarlier) has read write and execute permissions;
the group (prc) has read and write permissions;
the world does not have permission to move to this directory (via a cd command).

c)in contrast, for the directory **pub**:
the owner (pcarlier) has read, write and execute permissions;
the group (pub) has read and execute permissions;
the world has read and execute permissions.

This means that the world can move to this directory (via a cd command) and view the contents. Note that individual permissions in this directory may be set in such a way to allow access to all, some, or none of the files within.

F. Communication

You will need an SSH client to communicate with Inferno2. Telnet is not secure and is not supported. On the PCs you should find a program called *SSH Shell Client*. On the Macintosh, you can use the *Terminal* or *X11* application. On the Mac to use Molden (an X-windows application) you must use *X11*.

As of April 2009, VT ARC currently has three SGI Enterprise ALTIX 3700 Superclusters which provide Virginia Tech researchers with access to high performance computing on SGI hardware (see http://www.arc.vt.edu/arc/sgi/index.php):

Hostname	CPUs-Speed	Memory	Operating System
inferno.cc.vt.edu	20-1.3 GHz	32 GB	SLES10-ProPack5
inferno2.cc.vt.edu	128-1.6 GHz	512 GB	SLES10-ProPack5
cauldron.arc.vt.edu	64-1.5 GHz	320 GB	SLES10-ProPack5

Inferno2 and Cauldron are accessed via a queuing system via the head nodes charon1.arc.vt.edu or charon2.arc.vt.edu, see the latest details see SGI Queuing System.

To connect from *Terminal*

```
ssh username@charon1.arc.vt.edu
```

To start an X-windows session using *X11*

```
ssh –X username@charon1.arc.vt.edu
```

To transfer files to and from inferno

```
sftp username@charon1.arc.vt.edu
```

G. Process management

Note that you can kill only those processes of which you are the owner! Why would you want to kill a process? If your monitoring of a calculation (see Section 07. below) shows it is not going in the right direction, then kill it for your and everyone else's benefit. It's a timesharing system! See Section 2.D. below for use of the qdel command.

H. Bash Shell settings

To allow easy use of Gaussian09 utilities, you will want to edit your .bashrc file to include some useful aliases. I have added aliases (in blue below) for two utilities I use often, newzmat and freqchk:

```
# Sample .bashrc for SuSE Linux
# Copyright (c) SuSE GmbH Nuernberg

# There are 3 different types of shells in bash: the login shell,
normal shell
# and interactive shell. Login shells read ~/.profile and interactive
shells
# read ~/.bashrc; in our setup, /etc/profile sources ~/.bashrc - thus
all
# settings made here will also take effect in a login shell.
#
# NOTE: It is recommended to make language settings in ~/.profile
rather than
# here, since multilingual X sessions would not work properly if LANG
is over-
# ridden in every subshell.

# Some applications read the EDITOR variable to determine your
favourite text
# editor. So uncomment the line below and enter the editor of your
choice :-)
#export EDITOR=/usr/bin/vim
#export EDITOR=/usr/bin/mcedit

# For some news readers it makes sense to specify the NEWSSERVER
variable here
#export NEWSSERVER=your.news.server
```

```
# If you want to use a Palm device with Linux, uncomment the two lines
below.
# For some (older) Palm Pilots, you might need to set a lower baud rate
# e.g. 57600 or 38400; lowest is 9600 (very slow!)
#
#export PILOTPORT=/dev/pilot
#export PILOTRATE=115200

test -s ~/.alias && . ~/.alias || true

alias newzmat='/apps/packages/gaussian09a02/g09/newzmat'
alias freqchk='/apps/packages/gaussian09a02/g09/freqchk'

umask 027
```

Basically the .bashrc file is hidden and to see it you will need to invoke ls –a. You can edit it with the vi text editor. Note—before modifying it would be good to copy the original file to another file name just in case you mess it up. Then, to make the change in the .bashrc file active, either log-off and log on again, or type !source and the modified .bashrc file will be read

```
pcarlier@charon1:~> !source
source .bashrc
```

Note that the alias gives the complete routing of the newzmat and freqchk commands. If you fail to set these aliases, and type “freqchk” you will get the response “command not found.” If you want to conveniently use other Gaussian09 utilities, you can create the corresponding aliases in your .bashrc file. Thanks to Ryan Fortenberry (Crawford Group) for helping me with this!

I. Group Shared Directory

We have a shared directory at /home/PRC, and current g09 users in our group should be members of the group prc and thus should have access to it. I have placed awk scripts and other useful files here. Worth copying to your run directory right away are:

```
header
testE.gjf
testE.sh
unixdosconvert.awk
enantiomer.awk
```

You can check membership of the group prc by the following command:

```
getent group prc
```

You can identify a person’s full name from their PID, e.g. “pcarlier” by the command by

```
getent passwd PID
```

The special usefulness of the /home/PRC directory is that you can put files there for me and other group members to access.

02. Gaussian General

A. Input Files

Note that the suffix can be anything--by convention unix input is .com, windows .gjf, but it really doesn't matter. Here is the basic structure of an input file, including our suggested header:

```
%mem=4GB
%scr=xxx
%rwf=xxx
%int=xxx
%d2e=xxx
%nosave
%chk=xxx
#route section here

Title card required (Please replace this text with a description of
your calculation, including the source of your starting geometry)

charge multiplicity
geometry information
```

Make sure there is an extra carriage return at the end of your input file! That means there must be a blank line with nothing in at the end of the input file.

Why is there so much information in the header? Let's explain. When g09 runs a job, it generates up to six types of scratch
(temporary) files:

```
filename.chk
filename.rwf
filename.scr
filename.int
filename.d2e
filename.inp
```

Unless otherwise specified, these will be stored in the /tmp directory and normal termination of a g09 job will cause most of them (.scr, .chk, .rwf, .int, .d2e) to be automatically deleted. However, if a job crashes, these files will remain in the /tmp directory. Since the .chk, .scr, and .rwf files can be quite huge, the accumulation of these files in the /tmp directory is a nuisance. Thus, to avoid this from happening, we recommend that before the route section of your g09 input file, you include the following:

```
%scr=filename
%rwf=filename
%int=filename
%d2e=filename
%nosave
%chk=filename
```

where filename matches the name of your input file. These commands will locate the .scr, .rwf, .int, .d2e, and .chk files in the directory from which you launch your g09 command. Upon successful termination of the g09 job, the first 4 temporary files will be deleted, but the .chk file will be retained. We recommend retaining the .chk file since it is a useful source of the geometry and wavefunction, as well as force constants and frequency information, if a frequency job was performed.

Of course, if the job does not terminate normally, all the temporary files will be retained in your directory; these should be then be deleted manually.

With these preliminaries now out of the way, here is a sample input file:

```
%mem=4GB
%scr=h0006a
%rwf=h0006a
%int=h0006a
%d2e=h0006a
%nosave
%chk=h0006a
# rb3lyp/6-31G(d) opt(TS,CalcFC,noeigentest) freq

this is the cyclopropylnitrile/LiNH2 deprot TS, starting with the
optimized Spartan geom

0 1
 C
 C   1        1.502018
 C   1        1.515748    2       60.308719
 H   1        1.087202    2      117.748539    3     -109.713774
 H   2        1.088607    1      118.741447    3     -105.687970
 H   2        1.087636    1      117.714428    3      109.702519
 H   3        1.359060    1      126.035436    2      114.866487
 C   3        1.407657    1      123.805164    2     -112.539484
 N   8        1.179146    3      162.424838    1     -142.724248
 Li  9        2.010977    8       92.742563    3       -0.161269
 N   10       1.900574    9      117.278823    8       -0.153917
 H   11       1.022058   10      121.108491    9      112.750388
 H   11       1.023026   10      120.698282    9     -112.443206
 H   1        1.088844    2      118.799820    3      105.619638
```

If you are getting your geometry from a checkpoint file, you obviously don’t need to include a geometry specification section.   But you still need to specify the charge and multiplicity:

```
%mem=4GB
%scr=h0092
%rwf=h0092
%int=h0092
%d2e=h0092
%nosave
%chk=h0092
# rb3lyp/6-31+G(d) opt(ReadFC) guess=read geom=checkpoint

got geometry from h0035.chk  This is the (S)  NiPr ala BZD eq. geom.

0  1
```

Notes on the route section: ReadFC reads the Force constants from the checkpoint, which can save you a lot of time, especially for transition state optimizations. In this case the route line should read

```
# rb3lyp/6-31+G(d) opt(TS,ReadFC,noeigentest) guess=read
geom=checkpoint
```

The option guess=read means that the wavefunction guess will be taken from the checkpoint. I am not certain how much time this option saves, and it has caused some problems for me doing mp2 optimizations. A safer option is guess=Tcheck; in this case if there is some problem reading the wavefunction information a new guess will be generated. In any event the guess option can certainly be omitted. The option geom=checkpoint means you are getting the geometry from the checkpoint; I find this extremely convenient.

Notes on the header:
Calculations are fastest with 1 processor, memory for each processor is 4GB, the rwf,scr, int, and d2e are temporary files that otherwise will be stored in the temp directory of system. Upon normal completion of a calculation these files are apparently automatically deleted, courtesy of the %nosave command. However, if the calculation does not complete normally these files are not deleted. Since they can be quite large, this can present a problem. That is why we recommend you store them in your directory, so that you are responsible for deleting them. Finally, one last comment on the %nosave command: note that it precedes the %chk designation line. Checkpoint files are extremely useful, and I prefer not to delete them automatically. Since, %nosave precedes %chk, the checkpoint files are retrieved. Be aware however that these checkpoint files are extremely large.

B. Getting geometry information for your input file

Historically we have used *Spartan* to build structures, and then moved them to Gaussian. This is not a bad approach, because the *Spartan* builder is so easy to use. We find it especially useful for adding pre-optimized fragments, such as $Li(OMe_2)_3$. Note that *Spartan '10* includes the option to save a structure in cartesian coordinates

The discussion below assumes you know the difference between internal and cartesian coordinates. If you are not clear on this distinction, go to Section 03. below.

i. Starting with *Spartan '10* Input
Take your *Spartan* structure and save as cartesian coordinates (.xyz).

a)On Macintosh: Open the .xyz file with TextEdit, select and copy the text; using Terminal, create a file with the vim editor, and paste into the file and save. In this way you will not transfer any unwanted special characters. Note this is also a good way to get coordinates into a unix file from published coordinates. All you need are single space between the element, X, Y, and Z coordinates.

b)On PC: Open the .xyz file with WordPad, select and copy the text, and paste it into a unix text editor.

Alternatively, you can use the following .awk script (in /home/PRC) to remove invisible characters from the Cartesian coordinate file

```
awk –f unixdosconvert.awk filename > filenameU
```

(The script must create a new file—I typically add a U or u to the end of the filename to signify it is now unix compatible).

You will now need to add the header information described above to the cartesian coordinates in order to get a proper g09 input file.

I have developed a simple way to do this. First copy the header file from my /home/PRC

Copy your filename.xyz to filename.gjf. Open .gjf file in vim. Scroll to the top of the file. Hit escape to be sure you are in command mode. Then type the following to insert the header file at the beginning of the cartesian coordinate file.

```
:r header
```

hit return, and voilà. Then go in and change all the xxx file addresses, add a description of your calculations, add/remove carriage returns as necessary, and then add two carriage returns at the end of the file.

ii. Getting geometry info from a previous Gaussian calculation.
a) You can get the geometry from a previous checkpoint file by copying that file to a new checkpoint file (specified in your input file), and then use the command geom=checkpoint. You will need to specify the charge and multiplicity, as usual, after the route section.

b) An alternative is to use newzmat -ichk -oxyz. You will generate an .xyz file, to which you can add info to make a Gaussian info file. Note that you to use this command you will have to define an alias in your .bashrc file (see Section 01.H above). The actual coordinates that come out from this procedure and c) below are different, but this seems to stem from a different locus in space. The energies and geometries are the same. Note that you may have to type the following command once per login session before you use the newzmat command:

```
ulimit -s unlimited
```

c) To get cartesian coordinates from the .out file, do the following:

Open the .out file in *vim*, and search for the term Optimized,

```
:/Optimized
```

then scroll to the cartesian coordinates copy and paste into another file. Massage it to make it look like .xyz input: you basically have to remove columns 1 and 3, and change column 2 to the atom symbol (from atomic number). Then put the charge and multiplicity in front, as always, and give your typical header and calc. instructions.

When would you want to do this? If you need to get individual geometries out of a multiple optimization, say a dynamically constrained optimization. Alternatively you could do this to get unoptimized geometries to figure out what is going wrong with a calculation.

Intermediate geometries from a constrained optimization can be viewed and saved individually as .gjf files using *Gaussview4*. In case of unoptimized geometries intermediate geometries can be obtained using *Gaussview4* by clicking on ‘Read Intermediate Geometries (Optimization only)’ when opening the output file. You can view each of the intermediate geometries, and can save any of them as a .gjf file

d) If you are restarting, include the restart keyword and the calculation will get the starting geometry from the .chk file and will ignore any geometry info in the input file, but presumably will read the charge and multiplicity info.

There are many alternatives. The Collum Group uses *MOPAC*, the Troya Group uses *Molden*, and *Chem3D* can also generate Gaussian Input (apparently).
iii. Starting with pdb input (including Hydrogens; not from experimental protein x-ray .pdb, that have no H)

To convert a Spartan-generated pdb file (this includes hydrogens) to cartesian coordinates we use another awk script: entitled Spartan3678.awk

```
awk –f Spartan3678.awk filename.pdb > filename.xyz
```

The name refers to the fact that the awk script extracts columns 3,6,7, and 8 from the pdb file to create the cartesian coordinate file. Note that not all pdb files have this formatting, and the Spartan3678.awk script will work only pdb files that have Spartan-type formatting. Then complete the input file as described in i) above.

iv. Using *GaussView 4*
Gaussview 4 can be easily used to build geometries. It works really well for single molecules. For complexes or transition state structures involving more than one molecule, *Spartan* is definitely preferable. The greatest advantage is that one can directly generate a .gjf file which can then easily be converted to the unix compatible format. Overall though, *Spartan* is definitely much better for building geometries.

C. Special note on MP2 and CCSD Calculations

MP2 and other higher order correlated methods such as CCSD can consume huge quantities of memory. If you want to avoid getting nasty messages from the System Operator, when you perform single points or optimizations using these methods you should include a maxdisk statement in your route section, such as

```
# rmp2/6-31+G(d) opt(ReadFC) geom=checkpoint maxdisk=1GB
```

This will prevent you from writing more than 1GB of disk any point during the calc. and changes the way the calculation is done to make this happen. If we don't specify this maxdisk limit the program will assume we have inexhaustible disk resources. You can see the g09 for more explanation.

D. Starting Calculations:

Submission of Gaussian09 jobs can only be done via the queuing system SGI Queuing System. Therefore, in addition to creating an input file, you will need to create a script that embeds commands for starting the calculation and storing the results. One then submits the script file to the queue.

To submit your job to the queuing system use the command qsub:

```
qsub JobScript.sh
```

This will return your job name of the form xxxxx.queue.tcf-int.vt.edu. The number before the .queue.tcf-int.vt.edu is your job_number. **You must make a note of this number!**

If you need to remove your job from the queue, use qdel:

```
qdel <job_number>.
```

To see status information about your job, you can use:

```
showstart <job_number>  will tell you expected start and
                                finish times.

qstat -f <job_number>   general information about the job.
```

When your job has finished running any outputs to stdout or stderr will be placed in the files .o<job_number> and .e<job_number>. These 2 files will be in the directory that you submitted the job from.

To find information about your queued or running jobs you can use the command "istat" or "showq -p INFERNO2" to see the queue and free CPUs on inferno2

If you would like detailed information on your job, use qstat -f <job_number> or checkjob -v <job_number>.

If you have a job sitting in the queue that you think should be able to run, use the command checkjob -v <job_number> to see the reason the job is not running, as shown at the bottom of the output.

Now, what is the content of your jobscript .sh file? That is the critical issue. Here is the required information. Note that the text in blue receives specific comment below:

```
#!/bin/bash

# NOTE: Edit the number of cpus defined by "#PBS -lncpus" below to
match the
# number of cpus defined by "%nprocshared" in your Gaussian input file.

#PBS -lwalltime=01:00:00
#PBS -lncpus=1
#PBS -W group_list=sgiusers
#PBS -q inferno2_q

cd $PBS_O_WORKDIR

source $g09root/g09/bsd/g09.profile

g09 testE.gjf testE.out

exit;
```

You can copy this file (testE.sh) from the shared /home/PRC directory. Comments on the important blue text follow below:

1. Estimating the wall time has important consequences. If the processors are all occupied, jobs with a short estimated wall time will start sooner than those with a long estimated wall time. But if your estimated wall time is less than needed to finish the calculations, the job will terminate. This is not a huge problem for optimizations, since

these can be restarted (for example see sections 2.B., 4.A., and 5.B.). However, frequency calculations cannot be restarted—they start from the beginning. Experience will prove helpful here in setting the estimated wall time.

2. The default in this testE.sh file is to use one processor. Only use more than one if you know you need it!

You will note that the core of the .sh file embeds the launch command

```
g09 testE.gjf testE.out
```

Make sure you edit your .sh file to include the name of the input (.gjf) and output (.out) files that you want to use and write!

03. Z-Matrices

A. Internal vs Cartesian coordinates

Traditionally Z-matrices have implied the use of internal coordinates. But Gaussian seems to use this term more broadly sometimes to refer to cartesian coordinates. *GaussView* generates input files based on internal coordinates. What follows below is a description of internal coordinates.

How does it work? Everything is defined in terms of R, A, and D from other atoms

R is distance, A is angle, D is dihedral.
The first atom is at the origin. The second atom is defined only with a distance, the third with a distance and an angle, then the rest need three parameters. In this way one can understand the 3N-6 degrees of freedom of a molecule.

R, A, and D are defined with respect to atoms--note that the atoms are numbered sequentially (numbers not shown). In the example below C2 is 1.492004 Å away from C1. The distance of C3 from C2 is given, and the C3C2C1 angle is 60.314221 degrees. Finally, the Distance of H4 from C1 is given, as is the 412 angle, and the 4123 dihedral angle is -110.121478 degrees.

```
C
C       1       1.492004
C       2       1.504535    1       60.314221
H       1       1.077448    2      117.368114    3     -110.121478
```

Note that a common *GaussView* and *Molden* format for Z-matrices lists variables R1, R2, in the matrix itself, and then specifies the value of each variable below the matrix. It's the same thing.

Proper cartesian coordinate formatting (for another structure) in Gaussian looks like this:

```
C        -2.049483396021     -0.217543797504     -0.746273232373
C        -2.047273654168     -0.218752008125      0.751987769846
C        -0.722795562791     -0.264606217493      0.000864720920
H        -2.310203827396      0.704488520113     -1.257647361157
H        -2.322062833602     -1.127764765367      1.278962587352
H        -2.306482533438      0.702453630005      1.265615166991
H        -0.128590176337     -1.180975127251     -0.000751044777
C         0.154818794643      0.869471269358      0.000484656195
N         0.925807117143      1.744760715210      0.000052956257
Li        2.587989711588      0.515810028825     -0.003417932413
N         2.522448698964     -1.285377371677     -0.004796115153
H         2.738695938552     -1.872851709713     -0.813052056868
H         2.741114258643     -1.874173432797      0.801845842204
H        -2.325827763896     -1.125706083617     -1.273900711753
```

Note that these are simply the X, Y, and Z coordinates of the individual atoms.

B. Converting Z-matrices or Cartesian Coordinates to Enantiomeric Structures

To make the enantiomer: change the sign of the dihedral in the Z-matrix, or for cartesian coordinates, change the sign of one of the coordinates for each atom (i.e. all the x coord). Note that we have a new awk script to do this latter transformation for a geometry in cartesian coordinates:

```
awk –f enantiomer.awk file1.xyz>file1e.xyz
```

This awk script can be found in /home/PRC

C. Editing/Creating Z-matrices in *Molden*

Molden has a fantastic Z-matrix editor. It has the ability to create Z-matrices from cartesian coordinates, and redefinition of the Z-matrix is very easy---you just click on the atoms in the order that you wish the Z-matrix to be defined.

04. Geometry Optimization

A. Basic Commands

Since geometry optimizations are the basis for everything we do, please master the contents of this section.

i) In the absence of the opt keyword, only a single point energy will be determined.

Thus

```
#rb3lyp/6-31G(d)
```

and

```
#rb3lyp/6-31g(d) opt
```

are not the same.

ii) If you run out of optimization cycles, you can restart with the restart key word. This will ignore any geometry info in the input file and get the starting geometry from the specified checkpoint file.

```
#rb3lyp/6-31g(d) opt(restart,maxcycle=N)
```

Note that you can also increase the number of optimization steps by using an option.

```
#rb3lyp/6-31g(d) opt(maxcycle=N)
```

iii) The freq key word is used to generate frequencies (to verify a stationary point)

```
#rb3lyp/6-31g(d) opt freq
```

Frequency calculations are extremely time-consuming and memory-intensive. In certain projects we are approaching the limit of molecular size—sometimes frequency calculations just fail, even with 4GB of dedicated memory. We need to figure out how to get these to go. The utility freqmem can be used to estimate the memory requirements of a particular frequency calculation.

iv) Note that constrained optimizations are performed using the opt=modredundant keyword; this is described in Section 6 below

B. Options

Note that since geometry optimizations can be so time-consuming it is worthwhile to use the following time-saving techniques

i) The gdiis option is good for the flat potential surface of DFT. I recommend this if you have trouble converging to a stationary point:

```
#rb3lyp/6-31g(d) opt=gdiis freq
```

ii) Calculations that include diffuse functions can be very time-consuming. You can significantly speed up these jobs if you have force constant and wavefunction information from a previous lower level calculation. The typical situation we would use this technique is in going from 6-31G(d) to 6-31+G(d).

First, copy the 6-31G(d) .chk file to a new file name, then use that .chk file to provide your initial geometry, force constants, and wavefunction guess. Then specify the following in the route section

```
#rb3lyp/6-31+G(d) opt(ReadFC) guess=read geom=checkpoint
```

Note that for ReadFC to work, you must have previously run a frequency calculation— that is the source of the force constant information in the .chk file. Again, the guess=Tcheck option can replace the guess=read option. Don't forget to specify the charge and multiplicity!

The biggest time savings comes from having the force constants fed in (ReadFC), and in this regard correct force constants from a previous freq job are apparently greatly superior to the approximate force constants from a previous optimization.

iii) When having convergence problems,

```
opt=tight or opt=vtight
```

Polo also recommends

```
SCF=verytight int=ultrafine
```

We have not had much trouble in SCF convergence for single point calculations, except when using the MIDI basis sets. I think we had to loosen the SCF convergence criteria in these cases.

Note on Single Point Calculations
If a single point calculations are performed at B3LYP/6-31+G* on a B3LYP/6-31G* optimized geometries, it is better to specify the command SCF=tight especially for bigger systems to avoid errors. If the option of SCF=tight is needed, you will see the following output on using the command - egrep SCF filename.out

```
Warning!  SCF SP cutoffs with diffuse functions may be unreliable.
Consider SCF=Tight
SCF Done:  E(RB+HF-LYP) =  -1438.04848732     A.U. after   10 cycles
           Population analysis using the SCF density.
```

05. TS Geometry Optimization

A. Basics

The most important concept in TS Geometry optimization is the good initial guess. Basically, finding a saddle point is more difficult than finding a local minimum, and unless you start very close to the TS geometry, the optimization will have difficulty knowing which way to go. *Spartan '10* has a TS geometry library and an intuitive TS drawing program. In conjunction with a semi-empirical method, you may be able to quickly generate a good guess. Alternatively, you can do a dynamically constrained optimization in *Spartan '10* to find the approximate TS geometry. We have used this method with great success for bond-breaking transition structures. Finally, *g09* has function opt=QST2 which when given two equilibrium geometries (with identical atom numbering) will attempt to generate a good TS guess in between the structures and then start a TS opt. I have not had success with this method yet. Up to now my favorite method for a previously unexplored transition structure is to quickly find an AM1 or PM3 transition structure, and then use this as a guess for DFT TS opt. If you have already found a transition structure at DFT, and want to look for analogous ones, your best bet is to take this structure into *GaussView* or *Spartan '10* and make the required edits.

Once you've got your good guess initial geometry loaded into a *g09* input file, the basic command you will use is

```
#rb3lyp/6-31G(d) opt(TS,CalcFC,noeigentest) freq
```

In some cases you will have already calculated the force constants at a lower level of theory, say HF. In these cases you can use the ReadFC keyword, and get the geometry and the force constant information from the checkpoint file.

```
#rb3lyp/6-31+G(d) opt(TS,ReadFC)
```

Do you need the 'noeigentest' option here? I don't know. Calculating the Force Constants can be a very time consuming process. I have one example where the first SCF calc took about 5 hrs clock time, then the FC calc took about 24 hrs clock time. Then the SCF calc finally started to roll in. I think that if you had an approximate hessian you would significantly shorten the time between the 1st and 2nd SCF calc. So with ReadFC we get the force constants from a checkpoint file--this is specified in the header.

As mentioned above, this technique would be very useful if you were going to rb3lyp/6-31+g(d) from a rb3lyp/6-31g(d) transition structure and frequency job.

B. Restarting TS optimizations

To restart a TS calc, use the restart keyword

```
opt(restart,TS,CalcFC,noeigentest)
```

Note that I wonder whether the ReadFC would be superior here, since you are getting the geometry info from the checkpoint. Polo has noted that here the CalcFC command seems to be ignored, as the FC come from the checkpoint file.

See Sections 07. and 08. below on Monitoring Calculations and Visualizing Gaussian Output to determine the number of imaginary frequencies and to see the molecular motion corresponding to the imaginary frequency.

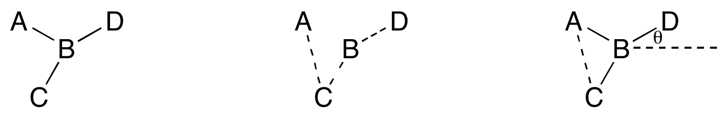
06. Constrained Optimizations

A. Setup

To perform constrained optimizations, the key command is

```
#rb3lyp/6-31g(d) opt=modredundant
```

You can use internal or cartesian coordinates. As usual, I prefer cartesian coordinates. The key is to know the atom numbers for your constraint. You can get these by looking at the structure in *Molden* (*GaussView* should also work). One atom constrains an atom to not move in 3D space. Two atoms define a distance, three an angle, and four contiguous atoms define a dihedral angle. Four non-contiguous atoms constitute an improper torsion which is a useful way to define or control the orientation of a group (D) relative to a plane (ACB).



the relationship of D to the plane defined by ABC can be measured in terms of improper torsion or deformation angle

Note that the improper torsion angle has the benefit of being signed, whereas a more conventional deformation angle is not signed.

After the geometry specification, include a carriage return, and then specify the distance, angle or dihedral you want to constrain, using the appropriate atom numbers; following this you can several different things

Examples of dihedral angle/improper torsion constraints

```
27 25 12 11 F
```

this would fix the 27 25 12 11 dihedral to what it is in the initial geometry: F stands for fixed.

```
27 25 12 11 10.0 F
```

this would fix the 27 25 12 11 dihedral to 10.0 degrees (note floating point, dihedral is different from initial geometry)

```
27 25 12 11 S 2 10.0
```

this would perform a series of constrained optimizations, where the 27 25 12 11 dihedral is first constrained to its original value, and then incremented twice by 10.0 degrees. Here S stands for 'step.' Note the use of the floating point number for the increment.

```
27 25 12 11 30.0 S 8 10.0
```

this would perform a series of constrained optimizations, where the 27 25 12 11 dihedral is first constrained to 30 degrees (different from its original value), and then incremented eight times by 10.0 degrees. Note the use of the floating point number for the increment.

B. Multiple Constraints

I have had occasion to perform optimizations with one fixed dihedral constraint and one dynamic dihedral constraint. As far as I can remember these have all failed. No atoms were shared in the constraints, and these calculations were also problematic in *Spartan*. What I have been able to do is perform optimizations with two fixed dihedral constraints

C. Monitoring constrained optimizations

To check conveniently--find out the internal coordinate name for your constraint--this will be R# for a bond length, A# for an angle, D# for a dihedral

```
egrep D37 h0053.out
```

this will show you the final value for your constraint

when you have a series of calculations in the same job, and want to extract the energies,

```
vim filename.out
:/Summary
```

this will take you to a list of eigenvalues left to right, under columns representing each optimized step in the calculation. The eigenvalue is the energy. Note that semiempirical energies are given in hartrees and are very small (since they reflect heat of formation, not electronic energy).

If you want to get the geometries and energies out

```
vim filename.out
:/Optimized
```

then scroll up to SCF done--your energy for the first optimization is here.

Confirm the value of your constraint by searching for it

```
:/D50
```

(an example where D50 is the constraint you are interested). Do this one more time and you will see the value of the constraint in the optimized point.

```
:/Standard

or

:/Orientation
```

This takes you to the standard coordinates for the optimized point. Copy and paste into a blank file and edit to adopt .xyz format.

07. Monitoring Calculations

Molden appears to be by far the best way to visually monitor your calculations in real time. Please see a group member about getting it installed on your Inferno2 account, and if you have a PC, that you have the *cygwin* application installed to provide a Linux-like environment. You will need this to emulate X-windows. On Mac, the X11 application is an optional but easy install in Mac OS X and unlike *cygwin* is a true X-windows application. *Gaussview4 also provides excellent way to monitory the progress of your calculation, by downloading the .out file and examining it offline (see below)*

But monitoring your job from the command line is also extremely useful, and this is where we will start.

A. Monitoring your jobs from the command line

i. To look at calculated energies

```
egrep SCF filename.out
```

If you are doing MP2 calculations:

```
egrep MP2 filename.out
```

For CCSD, you would egrep CCSD. Look out for energy loops—if they appear, kill the calculation and restart using opt=gdiis, or start from another geometry. The loop is not going to go away on its own. The best way to see such oscillation is to use *Molden* and click on "Geometric Convergence."

ii. To see if you are getting close to convergence

```
egrep 'YES| NO ' filename.out
```

this will give you info on

Max Force
RMS Force
Max Displacement
RMS Displacement

for DFT calc, forces often converge before Displacement. Under most circumstances, YES is needed for all four before the optimization will finish. Note that this command does not work with grep; you must use egrep. An example showing progression towards successful convergence (beginning omitted):

```
 Maximum Force            0.000066     0.000450     YES
 RMS     Force            0.000014     0.000300     YES
 Maximum Displacement     0.009215     0.001800     NO
 RMS     Displacement     0.001611     0.001200     NO
 Maximum Force            0.000037     0.000450     YES
 RMS     Force            0.000008     0.000300     YES
 Maximum Displacement     0.003695     0.001800     NO
 RMS     Displacement     0.000553     0.001200     YES
 Maximum Force            0.000022     0.000450     YES
 RMS     Force            0.000004     0.000300     YES
 Maximum Displacement     0.002317     0.001800     NO
 RMS     Displacement     0.000374     0.001200     YES
 Maximum Force            0.000012     0.000450     YES
 RMS     Force            0.000002     0.000300     YES
 Maximum Displacement     0.000918     0.001800     YES
 RMS     Displacement     0.000150     0.001200     YES
```

iii. To see if the Opt is done on a Opt+Freq job, type

```
egrep Job filename.out
```

If you have only one Job line, then the Opt is done and the Freq job is going. When the Freq. job is done you will get two Job lines

iv. To check the progress of your frequency job,

```
egrep vectors filename.out
```

Which will give you output like

```
135 vectors were produced by pass  0.
135 vectors were produced by pass  1.
135 vectors were produced by pass  2.
```

```
135 vectors were produced by pass  3.
135 vectors were produced by pass  4.
100 vectors were produced by pass  5.
18 vectors were produced by pass  6.
3 vectors were produced by pass  7.
```

Note that frequency jobs can take an awfully long time. As the calculation progresses you will see a decreasing number of vectors. When does it stop? Usually you get down to "1 vectors were produced..." but in the current example apparently after 3 vectors are found the calculation quickly comes to a successful termination.

v. To see how many imaginary frequencies there are, type

```
egrep imaginary filename.out
```

or

```
egrep Frequencies filename.out
```

which will give you a list of all the frequencies. Any imaginary frequencies will appear at the head of the list as negative frequencies. A convenient way to see this is

```
egrep Frequencies filename.out|more
```

The pipe to more (i.e. “lmore”)allows you to see the output screen by screen.

B. Following your optimizations visually with *GaussView, GABEDIT* and *Molden*.

To make the best use of your time, you need to know that the calculation is going in the right direction, and the best way to assess this is **visual inspection**: for this the most convenient method thus far is to use *GaussView, Gabedit,* or *Molden*. At the present time we do not have GaussView, and so PC users should use *Gabedit*, and Mac Users can use *Molden*. See Section 08 below.

08. Visualizing Gaussian Output

Gaussview4 and Molden will provide the easiest way for you to visualize your output, which will make much of the text below obsolete. *Molden* also has the ability to make .gif files that look pretty good. For publication purposes I still think *Molecule and PyMOL* give the best looking figures (see D below).

A. Directly from *GaussView4*

Reading your output file with *GaussView4* is a very convenient way to visualize the structures, animate frequencies, and generate pretty good looking pictures for your presentations. The output may not meet our standards for journal publications: for this purpose we regularly use *Molecule* and *PyMOL.*

With Gaussview (and Molden) there is not need of file conversion to any other format as it will directly read the .out file from Gaussian. The other good feature of Gaussview is that it usually gives an error message if the file has not terminated properly (This is really helpful if you have restarted a calculation, and you open up the first file with the intermediate geometry by mistake!).

Note: as of 1/11/12 we do not have a working copy of GaussView.

B. Installation and Use of *Gabedit* (PC)

Gabedit is a free graphical user interface program for computational chemistry-available for download from the link http://sites.google.com/site/allouchear/Home/gabedit after registration. Registration is not required but advisable to do since the developers provide info on email about latest upgrades. The installation is generally smooth, but pop-up questions come up during the installation which needs to be answered. (Typical for installing any windows based program). The latest available version (as of 01/12/12) is 2.4.0.

Visualization of Gaussian output files - Following steps need to be followed for visualization of your *.out files

1. First using the file transfer utility of your SSH client, download the *.out file to your hard drive – works better and faster than accessing a file at a remote location!
2. Launch Gabedit on your PC
3. In the main toolbar, go to Geometry -> Draw – A separate window will open up
4. In this new window, on the top left hand corner, you will find a button with the letter M on it – click this button – then move cursor to the Gaussian tab in the drop down list, another dropdown list appears – choose view last geometry from a Gaussian output file
5. Select the saved *.out file from your hard drive – your structure should show up.
6. You can convert the model type to 'ball and stick' using the icons on the left pane
7. To view the intermediate geometries, again click on the 'M' button and go to Read>Geometries Convergence > from a Gaussian output file - select the output file on your hard drive - A graph will show up with points corresponding to energies on it – click on the point and you can see the corresponding geometry in the Geometry window.
8. To visualize IR vibrations - click on the Display Geometry/Orbitals/ Density/Vibrations button (red and blue) on the main menu –A separate window will pop open– again select the 'M' button on the top left corner and point the cursor on the 'Animation'>'Vibration' option. Again a new window will open. In this new window-Go to Read>Read a Gaussian output file - and select the output file on your hard drive. Now you will see list of frequencies in this window – click on the frequency and hit 'Play' – you will see the animation in the earlier window. (Note: Just like Gaussview or Spartan you will see the imaginary frequencies at the top of the list)

C. Installation and Use of *Molden (Mac)*

Note: You will need to have X11 installed on your computer—It should be in your Utilities folder, in Applications. If it isn't, get your OSX installation disk and pop it in your drive. This program should be available as an **optional** install. Do NOT reinstall OSX from scratch.

1. Download the latest version of Molden executable for MacOSX according to the instructions

http://www.cmbi.ru.nl/molden/molden.html

http://www.cmbi.ru.nl/molden/howtoget.html

As of 6/14/10, this was molden4.8.macosX (same for gmolden—openGL version, good for showing orbitals??)

Put it in the directory you want (Applications?)

2. Open a terminal window in X11, the X Windows application on Macintosh.

[Remember that in X11, to interact with a window, you must click on it to select it. At that point the mouse or the keyboard will control that window]

Go to the directory where you have located Molden

To use this application you will have to do two things

a)change the downloaded file to an executable:

```
chmod ugo+x molden4.8.macosX
```

[Note—this is what happens—it now becomes an executable for you: you are changing permissions for user (u), group (g), and other (o) to add "x" or execute"]

```
CarlierMacBookPro09-2:unix paulcarlier$ ls -l gmolden*
-rwx------   1 paulcarlier   staff   3445780 Feb 14 21:16 gmolden4.8.macosX

CarlierMacBookPro09-2:unix paulcarlier$ chmod ugo+x gmolden4.8.macosX

CarlierMacBookPro09-2:unix paulcarlier$ ls -l gmolden*
-rwx--x--x   1 paulcarlier   staff   3445780 Feb 14 21:16 gmolden4.8.macosX
```

b)change the suffix to get rid of the periods (you can name it anything unique!)

```
mv molden4.8.macosX molden4_8_osX
```

3. Now from the terminal window type

```
open molden4_8_osX
```

This will launch Molden. In addition to the xterm window, you will get a MOLDEN window (blank) and a Molden Control window.

4. To open a g09 output file it must be on your computer (i.e. downloaded from Inferno): In the Molden Control window, under Miscellaneous, click read. This will open the Molden File Select window.

You will need to navigate to the directory that has your file. You should probably be starting from your home directory, then type /directory/subdirectory etc to get to the directory you want. Then select the file from the window, scrolling if necessary. Your structure should appear in the MOLDEN window.

Note that *Molden* reads all the intermediate geometries, **and by default, shows the starting geometry**. To see the final geometry, click the "Movie" button and you will see all the structures move past. Alternatively you can use the "First" and "Next" buttons. Particularly useful is the graphical presentation of Geometric convergence. Click on the "Geom. Conv." button and you will get three graphs. Click on any point on the Energy vs. point graph and that will take you to the corresponding structure.

Note that Molden provides a convenient method to measure distances, angles, and dihedrals. You click on the button, then once in the structure window, and then click on the atoms you are interested in.

You may wish to de-select the Shade box under Draw Mode in the Molden Control window. With your cursor in the MOLDEN window you can now rotate your structure to the desired view.

5. To see the course of the optimization on the energies, click Geom. conv. under Convergence (Molden Control). You will get a new window called Geom. Convergence. This is an interactive plot-click anywhere and you will go to that structure.

6. To see the Frequencies, click the Norm. Mode box under Frequencies in the Molden Control window. This will give you two new windows, Spectrum and Molden Frequency Select.

To animate a particular frequency, click it in the Molden Frequency Select window. To stop the animation, deselect the Norm. Mode box in the Molden Control window.

7. To change to another structure, simply click the appropriate file name and the new structure will appear.

8. To exit Molden, simply click the Skull icon in the Molden Control window.

D. Export as .xyz and view in Spartan

Use newzmat to generate a pdb file from your checkpoint file:

```
newzmat –ichk –opdb filename.chk filename.pdb
```

You can then view this using *Spartan '10*.

D. Export as .xyz and Open in *Molecule*

I believe that *Molecule* makes the best B/W high resolution output for publication; we only have it on the Mac platform (current version we have is 2.006, from 7/16/08, which is Beta). Use newzmat to create a cartesian coordinate file:

```
newzmat –ichk –oxyz filename.chk filename.xyz
```

Then on the Mac launch *Molecule* and open the file (make sure you select "Enable: Readable Files"). You will get an error message "Error in input"; ignore this by clicking OK, and select "XYZ with tabs/spaces" when you are asked to specify the input type. *Molecule* is not the most intuitive software but it makes nice figures.

Note: as of 1/12/11 there is no version of Molecule that is compatible with OS 10.7. Paul may have a working copy of Molecule on an old Mac Notebook computer.

E. Export as .xyz or .pdb and Open in *PyMOL*

PyMOL makes visually stunning figures in color, with shadows. There is so much you can do with this program! See Section 17. below for how to get started.

F. Visualizing Imaginary Frequencies

To do a quick check that your imaginary frequency corresponds to the correct molecular motion, *Molden* or *GaussView4* are the easiest option. In *Gaussview4* these can be easily visualized by going to Results and then selecting 'Vibrations'.

GaussView4 now has an easy option for making movies as well. Imaginary frequencies associated with transition state calculations can directly be saved in the .gif movie format. Any vibration can be saved as a movie by clicking on the option of – 'Save movie' and selecting 'Save movie file'. These files can easily be incorporated into a powerpoint slide, and are animated in the presentation mode.

For another option on making movies, see the gOpenMol description in Section 10. below.

G. Measuring bond lengths, angles, dihedrals

Please note that since coordinates in .pdb files are truncated to 4 decimal places, all measurements should be made on the original .out files in *GaussView4*, or on the .xyz files in *Spartan '10*, *PyMOL*, or *Molecule*.

09. Archiving Calculations

First, remember that we use a shared resource. So get rid of all your temporary files: .int, .rwf, .d2e, .scr: if you formatted your input file correctly, these should be deleted in all cases except where your calculation crashed. Get rid of your .sh files.

Second, we suggest periodically moving completed calculations out of the run directory to project-specific storage folders in your home directory.

Third, when a project is finished, copy the files in each project directory to an external drive (we have a few in the lab), and erase them from your directory on the shared file system. This last step is crucial for the other users of the shared file system.

10. Implicit Solvation using SCRF

There are many theoretical models for putting a molecule in a field of virtual solvent. Note this is different than explicitly including solvent molecules as ligands for Li. Note that Wiberg has shown that SCRF calculations with solvents as nonpolar as cyclohexane can result in significant stabilizations. However, the effect of implicit solvation with non-polar solvents on energy of reactions is often small (in our experience). Larger effects might be expected for non-polar solvents. One important exception we have seen is in ion pair separation of organolithiums. Here the separated ion pair is more stabilized than the stabilizations enjoyed by the contact ion pair and THF.

Two different solvation models are available in Gaussian03 – Polarised Continuum Model (PCM) and the Onsager solvation model. Both these methods calculate the energy of a molecule by placing the solute in a cavity, and applying a dielectric through it. Along with the treatment applied by these different solvation models, the main distinguishing feature between these solvation models is the type of cavity that is used for these calculations.

Onsager Model:

For the Onsager model, a radius for a spherical cavity has to be specified. One of the problems associated with this model is that for a non-spherical molecules, a spherical cavity might not be an appropriate fit. Our experience showed us that the input radius has significant effects on the energy obtained, varying by even up to 4 kcal/mol per angstrom. These radii values are obtained by running a 'Volume' calculation on the molecule. A significant problem that arises from this 'Volume' calculation is that the values obtained are not uniquely defined by the input geometry, because Gaussian03 invokes a Monte Carlo integration to determine the volume. Due to the sensitivity of Onsager energies to the radius employed (a_0), uncertainty in radius propagates to uncertainty in energy.

If you need to calculate an Onsager energy, we recommend the following.

i) Perform the 'Volume' calculation on the vacuum optimized structure which you plan to use as your input geometry multiple times (10x) with the option 'tight' and take the average value as your input radius. As these calculations finish in a matter of minutes for big molecules, time should not be a constraint. The standard deviation of the radius could then be used with the mean to calculate single point energies at the mean a0, mean a0 + s, and mean a0 - s. In this way, the mean energy and standard deviation can be determined

or

ii) Perform the Volume calculation only once, but calculate energies at $0.9*a_0$ and $1.1*a_0$ to determine the mean energy and standard deviation. This 10% variation in radius approach was taken in our 2010 J. Org. Chem. paper (Deora & Carlier).

A sample route line for a volume calculation:

```
# rb3lyp/6-31g(d) Volume=tight
```

The recommended radius can be easily located in the output file by using the command.

```
egrep SCRF filename.out
```

The output obtained looks like the following:

```
Recommended a0 for SCRF calculation = 3.91 angstrom ( 7.40 bohr)
```

Note that the recommended radius a_0 obtained from the Volume calculations output is 0.5 Å larger than the radius that would be extracted from assumption of a spherical be in angstroms.

In order to perform a single point Onsager calculation the following should be your route line:

```
# rb3lyp/6-31g(d)  SCRF=(Dipole, Dielectric=7.58, a0=6.63, Solvent=thf)
```

where SCRF (Self-Consistent Reaction Field) for Onsager calculations is Dipole, the dielectric constant has to be specified (Gaussian has values for a number of solvents in their user manual), and the solvent (in this case - THF) is also specified. The average radius a_0 can be obtained by the method described above.

The energy output looks like the output obtained from an energy calculation in vacuum

and can be obtained using the following command:

```
egrep SCF filename.out
```

Onsager optimizations are performed the same way with the addition of opt and freq options in the route line as follows:

```
# rb3lyp/6-31g(d)  SCRF=(Dipole, Dielectric=7.58, a0=6.63, Solvent=thf)
opt freq
```

Onsager calculations can be performed with DFT and HF methods. However, in Gaussian 03, they could not be performed using the MP2 method.

Polarized Continuum Model (PCM):

As mentioned earlier, the other solvation model that is frequently used is the PCM. Unlike in the case of the Onsager model, the PCM comes with the option of predesigned cavities, owing to which there are no fluctuations associated with the energies obtained. There are several different flavors of PCM (i.e. CPCM, SCI-PCM); but in Gaussian03 with our typically large molecules, we have had luck only with PCM itself.

UA0: This is the default radius in Gaussian03. In this model, the hydrogens are incorporated in the spherical cavities of the heavy atom they are bonded to.

Bondi: Uses the Bondi atomic radii for all atoms and treats hydrogens explicitly.

Pauling: Uses the Pauling atomic radii for all atoms and treats hydrogens explicitly. Is recommended for ionic molecules.

Note: The new default for Gaussian 09 is UFF (Universal Force Field), and has explicit hydrogens unlike UA0. We have not used this yet.

We have used the default settings, as well Bondi and Pauling radii with great success. In some cases where Bondi has failed in generation of the cavity, Pauling has been applied successfully.

The route line for default PCM single point calculations is as follows:

```
#rb3lyp/6-31G(D) scrf=(PCM,solvent=thf)
```

If you want to specify additional options, put "read" in the route section

```
#rb3lyp/6-31G(D) scrf=(PCM,solvent=thf,read)
```

at the end of the file you specify your options, such as

```
(empty line)
radii=bondi
surface=SES
(empty line)
```

where Surface=SES is the default for Gaussian 03. The default for Gaussian09 is Van Der Waals surface. The only other option for surface is SAS, though this seems less refined than the others.

So far, we have used only the option Surface=SES, and gotten good results.

PCM calculations can be performed using HF, DFT as well as MP2 methods with diffuse functions without any problems for single point calculations.

We have been able to perform Geometry Optimization on fairly big systems with the Onsager model only. Significant improvements have been observed in geometries of ionic compounds on incorporation of Onsager solvation during optimization. Geometry optimizations under PCM with *Gaussian03* have been possible in cases with up to 16 heavy atoms (work of Larry Williams). Perhaps the performance of PCM optimizations is improved in *Gaussian09.*

.

12. Thermodynamic Corrections

A. Introduction

In using computational chemistry to describe a chemical process, there are several ways to express energetic changes. Raw output from an energy optimization is the electronic energy. Comparing reactants vs. products shows a change in electronic energy, not ΔH or ΔG. These thermodynamic values must be calculated separately.

Thermodynamic parameters are determined through the use of a frequency calculation in Gaussian '09. There is a terrific document entitled "Thermochemistry in Gaussian" available at http://www.gaussian.com/g_whitepap/thermo.htm . Please read through this manual for a detailed explanation of the sources of thermodynamic quantities, thermochemical output, and some worked-out examples.

B. Setting up a calculation to obtain thermodynamic parameters

Adding the 'freq' keyword to the route section prompts Gaussian to start calculating frequencies and thermodynamic parameters once the geometry is optimized. Note that the geometry of the structure must be optimized to obtain reliable thermodynamic data.

Example:

```
# rb3lyp/6-31g(d) opt freq
```

It is important to note how these values are actually calculated. Free energy (G), enthalpy (H), entropy (S), and zero-point corrections are all derived from the vibrational frequencies of the optimized structure. These values are calculated as correction factors to be added to the original electronic energy. The example below shows how enthalpy (H) would be determined for an optimized structure:

```
Example [thf: calculated at B3LYP/6-31G*]:

A) Electronic energy=-233.449443 (hartrees)
B) Thermal correction to enthalpy=0.123272 (hartrees)
C) Therefore H = Electronic energy + enthalpic correction = -233.326171
(hartrees)

Multiplying this result by 627.51 will convert units to kcal/mol

-233.326171 (hartrees) * 627.51 kcal/mol/hartree = -145786.9955
kcal/mol
```

This procedure must be applied to each molecular species in a thermodynamic cycle in order to obtain valid thermodynamic parameters.

C. Locating thermodynamic data in the output file

i. To get the thermal and free energy corrections to enthalpy

```
pcarlier@inferno2:~/run> egrep correction h0498.out
 Zero-point correction=                           0.147025
(Hartree/Particle)
 Thermal correction to Energy=                    0.152503
 Thermal correction to Enthalpy=                  0.153447
 Thermal correction to Gibbs Free Energy=         0.119061
```

Now you can take these corrections and apply them the electronic energy. This is especially convenient when you plan to apply the free energy correction at a lower basis (e.g. B3LYP/6-31G*) to a higher basis single point energy, or a PCM-corrected electronic energy, a BSSE-corrected electronic energy, or an energy that includes all three!

ii. But if you prefer to get the sum of corrections and electronic energy:

```
pcarlier@inferno2:~/run> egrep Sum h0498.out
 Sum of Mulliken charges=  -0.00000
 Sum of Mulliken charges=  -0.00000
 Sum of Mulliken charges=  -0.00000
 Sum of Mulliken charges=  -0.00000
 Sum of Mulliken charges=  -0.00000
 Sum of Mulliken charges=  -0.00000
 Sum of APT charges=  -0.00000
 Sum of APT charges=  -0.00000
 Sum of electronic and zero-point Energies=           -234.501264
 Sum of electronic and thermal Energies=              -234.495786
 Sum of electronic and thermal Enthalpies=            -234.494842
 Sum of electronic and thermal Free Energies=         -234.529228
```

iii. To see detailed entropy information, type

```
more h0498.out
```

then within the more program, type

```
/Thermal
```

and you will get the following output (truncated at the bottom:

```
/Thermal
...skipping

 Zero-point correction=                           0.147025
(Hartree/Particle)
 Thermal correction to Energy=                    0.152503
 Thermal correction to Enthalpy=                  0.153447
 Thermal correction to Gibbs Free Energy=         0.119061
 Sum of electronic and zero-point Energies=           -234.501264
 Sum of electronic and thermal Energies=              -234.495786
 Sum of electronic and thermal Enthalpies=            -234.494842
 Sum of electronic and thermal Free Energies=         -234.529228

                    E (Thermal)             CV                S
                     KCal/Mol        Cal/Mol-Kelvin    Cal/Mol-Kelvin
 Total                 95.697             21.296             72.372
 Electronic             0.000              0.000              0.000
 Translational          0.889              2.981             39.129
 Rotational             0.889              2.981             24.809
 Vibrational           93.919             15.335              8.433
```

D. Obtaining thermodynamic parameters at any temperature

Gaussian '09 calculates thermodynamic parameters at STP (Standard Temperature and Pressure) which means 298.150 K and 1atm. Often, we desire these values at much lower temperatures. There is a utility built into Gaussian '09 called freqchk. This utility pulls thermodynamic values from filename.chk, and recalculates them for whatever temperatures and/or pressures you specify. This is an easy method for changing temperature and pressure after the calculation is completed

Example: (**user input** in bold)

```
 pcarlier@charon1:~/sgi/run> freqchk
 Checkpoint file? j0007.chk
 Write Hyperchem files? n
 Temperature (K)? [0=>298.15] 193
 Pressure (Atm)? [0=>1 atm] 0
 Scale factor for frequencies during thermochemistry? [0=>1/1.12]
0.9804
 Do you want to use the principal isotope masses? [Y]: y
 ---------------------------------------------------------------------
 Center     Atomic                   Coordinates (Angstroms)
 Number     Number                      X           Y           Z
 ---------------------------------------------------------------------
      1        12                  -0.286520   -0.578356   -1.611669
      2        17                  -0.830292   -1.383436    0.657515
      3         8                  -2.141684    0.267642   -2.142086
(lots of data after this, then)

Project out gradient direction? [N] n
```

Notes on using this utility:

Write Hyperchem files? I enter 'no'.

Temperature (K) [0=>298.15]: Entering 0 means the system will automatically assume you want to use the standard temperature of 298.150K (default temp). Any other temperature will be read as is (remember—use degrees Kelvin).

Pressure (Atm)> [0=>1 atm]: As above, entering 0 will signify 1 atm (default pressure).

Scaling factor [0=>1/1.12]: This value is specific to the method/basis set of the calculation. For each level of calculation there is an appropriate scaling factor to be applied. Entering 1, signifies no scaling.

B3LYP/6-31G* = 0.9804

Principal isotope masses? Select yes for this question. Otherwise it will ask you what mass you want to use for each atom.

While this utility is running, a large amount of data starts scrolling by. Fear not. The thermodynamic correction data you want is only a quick scroll up from the bottom.

13. Charge and Orbital Analyses of Calculated Structures

A. Mulliken Charges

Mulliken charges are calculated automatically during geometry optimizations, and can be recovered from the output file. They are calculated twice during the optimization: the first set is for the starting geometry and the second is for the optimized geometry.

Mulliken charges can be extracted using the 'more' command.

```
more *.out
```

Since the output file contains two sets of mulliken charges, we need to get to the optimized section of the output file and for that we need to use the 'optimized' keyword while in the more command mode.

```
/Optimized
```

Once this is done, then type mulliken keyword, to get the charges of the optimized structure

```
/Mulliken
```

Sample mulliken charge output for a cation looks as follows:

```
Mulliken atomic charges:

     1  C   -0.598270
     2  H    0.241501
     3  H    0.241487
     4  C    1.073842
     5  C   -0.816830
     6  H    0.218026
     7  H    0.239357
     8  H    0.240034
     9  C   -0.644768
    10  H    0.277717
    11  H    0.223774
    12  H    0.277934
    13  O   -0.279529
    14  H    0.305725
 Sum of Mulliken charges=   1.00000
```

In this case, the molecule is positively charged and therefore the sum of all charges equals one. Mulliken charges are also available with hydrogens summed into heavy atoms. These are found immediately after the Mulliken atomic charges.

B. Natural Bond Order (NBO)

NBO analysis gives the occupancy of the bonds. It treats a bond as a Lewis type 2 electron bonds. The orbital is said to be occupied, if the number of electrons in the orbital is greater than 1.95.

The NBO analysis can be done as a single point calculation with the route section as follows:

```
%mem=4GB
%scr=*
%rwf=*
%int=*
%d2e=*
%nosave
%chk=*
# B3LYP/6-311++G(d,p) geom=checkpoint Pop=NBORead

Protonated epoxide 5H+. Starting geometry B3LYP/6-311+G**

1 1



$nbo bndidx $end
```

The end line “$nbo bndidx $end” is required to calculate different types of Bond orders including NBOs, Wiberg bond index, AO and MO bond orders.

NBO output can be extracted from the output file using the command

more *.out

followed by

```
/Natural
```

If the analysis is done as a part of a geometry optimization, then the NBO analysis is done twice, once at the beginning of the optimization and once at the end. In this case, the ‘more’ command should be followed by

```
/Optimized
/Natural
```

C. NBO OUTPUT:

NBO population analysis divides all the orbitals in four categories:

BD: Bonding orbital
CR: Core orbital
LP: Lone Pair orbital
RY: Rydberg orbital (almost empty high energy to account for basis functions that include polarization)

An NBO output looks like the following:

```
(Occupancy)   Bond orbital/ Coefficients/ Hybrids
---------------------------------------------------------------------------------
    1. (1.98970) BD (1) C   1 - H   2
        ( 61.97%)   0.7872* C   1 s( 30.83%)p 2.24( 69.11%)d 0.00(  0.06%)
                                          -0.0001  0.5549  0.0183 -0.0017  0.0001
                                          -0.4149  0.0145  0.0016 -0.0018 -0.0627
                                           0.0117  0.0009 -0.0009  0.7170  0.0225
                                           0.0021 -0.0020 -0.0028 -0.0151  0.0052
                                           0.0047  0.0190

        ( 38.03%)   0.6167* H   2 s( 99.95%)p 0.00(  0.05%)
                                           0.9998 -0.0006 -0.0000 -0.0000  0.0120
                                           0.0042 -0.0181
```

This is the C-H bond. This bond is between C1 and H2.

The values (61.97%) and (38.03%) give the percent of electron density on Carbon and Hydrogen respectively, thus showing the polarization of the bond (in this case, the bond is polarized towards carbon).

The s(30.83%) for Carbon and s(99.95%) for Hydrogen, show the extent of s character. In case of hydrogen, the 's' character is 99.95% and the 's' character for the carbon is 30.83% which shows 'sp^3' hybridisation.

D. Wiberg NAO Bond Index (WBI)

Wiberg bond index is used to measure the degree of bonding. WBI is calculated during a normal NBO analysis calculation.

WBI values can be extracted using the 'more' command followed by

```
/Wiberg
```

Even in this case, if the WBI values are being extracted after an optimization instead of a Single point calculation don't forget to use the command

```
/Optimized
```

```
/Wiberg
```

14. Basis Set Superposition Error: Counterpoise Correction

A. Introduction

Basis Set Superposition Error (BSSE) is an error incurred during the calculation of a super-molecule (i.e., dimer). Suppose two molecules, A and B join together to form AB dimer, as shown below.

$$A + B \rightarrow AB$$

The “dimerization energy” is not simply $\Delta E = E_{AB} - (E_A + E_B)$. The term ΔE we would obtain in this way would be artificially low due to this BSSE. In this example, suppose that each fragment is calculated using 12 basis function orbitals, and the dimer AB, being composed of one A and one B fragment is calculated using 24 basis function orbitals. When the electronic energy of AB is calculated, all 24 basis function orbitals are available to each A or B fragment, which means that electrons from one A fragment can use the orbitals of the B fragment to stabilize it, and vice versa. The direct result of this is that the energy of the dimer AB will be too low.

B. Running a Counterpoise calculation

In order to correct for the BSSE imposed when calculating clusters of molecular fragments (e.g. dimers, solvated monomers, etc.), a **counterpoise** correction can be implemented on an optimized geometry. The subject of how best to compensate for BSSE is quite complex and counterpoise is just one possible way to estimate the BSSE. But the counterpoise correction is defined as follows:

$$\Delta E^{CP} = E_{AB} - (E_{A\{AB\}} + E_{B\{AB\}})$$

Where $E_{A\{AB\}}$ signifies the energy of fragment A calculated using all the basis function orbitals available to dimer AB.

You will need to get the optimized geometry from your previous geometry optimization and put this in your input file. But you will need to be absolutely certain of your atom # assignments to properly create the input file.

Every atom of the structure in the input file must be correctly assigned to its proper molecular fragment. Use *Molden* or *GaussView* or *Spartan* and examine each atom to ensure each atoms numeric assignment (e.g. O is atom #1 above, Li is atom #2, a C is atom #3). **Be very, very careful. A misassignment will destroy a BSSE calculation. Make careful notes as you do this. Some people find it useful to**

write out the structure on paper and put the atom numbers on accordingly so you can assign each fragment correctly.

Nipa prefers to open the structure in *Gaussview4* with the ‘labels’ option on. If the atom numberings are the same in both cases (which it absolutely is if you use *Gaussview4* to generate the .gjf file), then the technique works really well. The other advantage is that you can open up the XYZ coordinates in *Gaussview4* while doing this, this way you even have the Cartesian coordinates in the written form to check against your .gjf files (this works only if you have your .gjf file originating from *Gaussview4*, as using the XYZ coordinates usually has a different locus point , and hence different coordinates).

When editing the Z-Matrix, the format for internal coordinates is as follows: after the last number in the matrix line, add 0,Fragment #. The 0 must be present after the dihedral angle value. Note that this does not apply to atoms 1-3, as there is no dihedral angle that directly defines these three atoms. *For cartesian coordinates, simply place a comma and the fragment number after the Z coordinate value for each atom*.

Sample Counterpoise input format for internal coordinates:

```
# rb3lyp/6-31g(d) Counterpoise=2 geom=connectivity    [Note: Counterpoise=# of fragments]

THF monosolvate of C-Lithiated Monomer

0   1 0 1 0 1        [Note: format is overall charge, multiplicity, charge of fragment 1, multiplicity, etc.]
 O,2     [2 denotes fragment assignment]
 Li                 1       1.873552,1
 C                  2       2.169765     1      156.968899,1
 C                  3       1.447149     2       75.359781     1      -158.376879,0,1 [0 follows
dihedral, 1 is frag. #]
 C                  4       1.423789     3      120.986626     2       119.496245,0,1
 C                  5        1.385636     4      121.493836     3     -174.151467,0,1
 C                  6        1.405027     5      121.306720     4         0.398801,0,1
 C                  7        1.390937     6      118.439055     5        -1.054454,0,1
 C                  8        1.401740     7      121.107820     6         0.785437,0,1
 H                  9        1.089389     8      119.199427     7     -173.764885,0,1
 H                  8        1.088691     7      120.099408     6     -177.706375,0,1
 H                  7        1.085794     6      120.650531     5     -179.916333,0,1
 H                  6        1.087815     5      119.028538     4     -179.659010,0,1
 H                  5        1.088013     4      118.658403     3         6.029556,0,1
 C                  3        1.394473     2       70.395488     1        71.801006,0,1
 N                 15        1.182453     3      167.932274     2        12.919999,0,1
 H                  3        1.083311     2      129.699417     1       -41.007575,0,1
 C                  1        1.464954     2      116.328255    15         7.715954,0,2
 C                 18        1.538538     1      106.043581     2     -140.248791,0,2
 C                 19        1.540089    18      103.580800     1        17.428509,0,2
 C                  1        1.446787     2      124.067007    15     -133.878184,0,2
 H                 21        1.093723     1      107.508995     2        -6.756916,0,2
 H                 21        1.098053     1      109.147348     2     -124.893305,0,2
 H                 20        1.093445    19      113.026823    18     -154.568022,0,2
 H                 20        1.096149    19      110.759422    18        83.513792,0,2
 H                 19        1.092574    18      111.785554     1       138.973865,0,2
```

```
H                19        1.094544    18        110.667870     1       -100.723686,0,2
H                18        1.094182     1        106.606333     2        -19.185987,0,2
H                18        1.094745     1        108.093137     2         98.488797,0,2
```

Sample Counterpoise input for Cartesian coordinates

```
# rb3lyp/6-31+G(d) counterpoise=2

DMA enolate and methyl chloride alkylation TS from h0113.xyz

-1 1 -1 1 0 1
 C         1.343477879814     -0.772380105194      0.533385825331,1
 H         1.315014867780     -0.727099181959      1.617767168140,1
 H         2.330957531784     -0.765202145223      0.080848817173,1
 C         0.332689899132     -1.411446133045     -0.226379130666,1
 O         0.449943439420     -1.746161800948     -1.435523596347,1
 N        -0.960643624364     -1.608148801034      0.377766890013,1
 C        -1.127166224106     -1.497677636275      1.810404888802,1
 C        -1.820022974087     -2.595581430403     -0.244312502250,1
 H        -0.701876036017     -0.559643625034      2.177377628416,1
 H        -0.657585551543     -2.330861089859      2.372451972720,1
 H        -2.199307822346     -1.494426170911      2.049886933846,1
 H        -2.875123811001     -2.347995784228     -0.057258866794,1
 H        -1.638146760964     -3.620351564286      0.141609482528,1
 H        -1.625139972009     -2.595687849551     -1.317176083054,1
 H        -0.088587554714      1.452050478918      0.441362318283,2
 C         0.850854464496      1.657072996735     -0.046822504536,2
 H         0.990444933971      1.244764082150     -1.034215443681,2
 H         1.721934290714      1.780891235888      0.575942276256,2
 Cl        0.533727968708      3.700586557530     -0.539202066073,2
```

Remember. Be very careful in your atom assignment and in your assignment of charge and multiplicity to the fragments.

C. How the Counterpoise calculation actually works

During the counterpoise calculation, Gaussian '03 performs 5 separate single point energy (SPE) calculations at the level specified in the route section.

Step 1: SPE calculation of the dimer

Step 2: SPE calculation of fragment 1 using all available orbitals. This is called a DCBS calculation. Think D for dimer.

Step 3: SPE calculation of fragment 2 using all available orbitals. $DCBS_2$.

Step 4: SPE calculation of fragment 1 using only orbitals associated with that fragment. This is called an MCBS calculation. Think M for monomer.

Step 5: SPE calculation of fragment 2 using only its own orbitals. $MCBS_2$.

Once these 5 electronic energies are determined, a Counterpoise correction is calculated as such:

counterpoise correction = ($MCBS_1$ + $MCBS_2$) – ($DCBS_1$ + $DCBS_2$)

Note that by this definition the counterpoise correction is always positive.

Counterpoise corrected energy is simply the sum of the original electronic energy (calculated in Step1) and the BSSE correction.

C. Deciphering Counterpoise Calculation Output

Three kinds of data can be found in the output for calculations of this type.

To obtain the Counterpoise corrected energy and the Counterpoise correction (called Counterpoise BSSE energy):

```
$ egrep Counterpoise h0124.out
```

which will give you the following output:

```
 Counterpoise: doing DCBS calculation for fragment   1 NewBq=T
 Counterpoise: doing DCBS calculation for fragment   2 NewBq=T
 Counterpoise: doing MCBS calculation for fragment   1
 Counterpoise: doing MCBS calculation for fragment   2
 Counterpoise: corrected energy =    -787.356695644622
 Counterpoise: BSSE energy =       0.000470907464
```

If you want to examine the SCF energy for each of the steps 1-5 listed above, then

```
$ egrep SCF h0124.out
SCF Done:  E(RB+HF-LYP) =  -787.357166552     A.U. after    9 cycles
            Population analysis using the SCF density.
 SCF Done:  E(RB+HF-LYP) =  -287.229868396     A.U. after    8 cycles
            Population analysis using the SCF density.
 SCF Done:  E(RB+HF-LYP) =  -500.111250669     A.U. after    5 cycles
            Population analysis using the SCF density.
 SCF Done:  E(RB+HF-LYP) =  -287.229604580     A.U. after    8 cycles
            Population analysis using the SCF density.
 SCF Done:  E(RB+HF-LYP) =  -500.111043578     A.U. after    5 cycles
            Population analysis using the SCF density.
```

If you have any concerns about your counterpoise correction you can record each value. In the case of a simple 2A -> A2 dimerization, the two DCBS terms should be very similar, as should be the MCBS terms. If they are not similar, you may have missassigned the atoms to the respective fragments. For whatever reason, this missassignment can wreak havok with the calculated counterpoise correction.

D. An example of BSSE estimation using the counterpoise method:

Consider the reaction: 2PhCHCN-Li ➔ $[PhCHCN\text{-}Li]_2$ (dimerization)

In calculating the dimerization energy, it may seem intuitive:

$$\Delta E_{dimerization} = E_{dimer} - 2*E_{monomer}$$

Calculated data at B3LYP/6-31G*:

$E_{monomer}$ =	-370.7372461 hartrees
E_{dimer} =	-741.5601649 hartrees

$\Delta E_{dimerization}$ = -53.76 kcal/mol

Because the right side of the reaction incorporates two molecular fragments, the energy of the dimer will be artificially lowered by the BSSE. To correct for this error, we perform a counterpoise calculation.

counterpoise data:

Step 1:	-741.560091
Step 2:	-370.7338629
Step 3:	-370.7338738
Step 4:	-370.7309419
Step 5:	-370.7309527

Counterpoise correction: 0.005842195
Counterpoise corrected energy: -741.5542488

Now, with application of counterpoise corrected energy:

$$\Delta E_{dimerization}\text{ (corr.)}=E_{BSSEdimer} - 2*E_{monomer} = -50.09\text{ kcal/mol}$$

In this counterpoise calculation, the BSSE is estimated to be 3.67 kcal/mol.

Note that as the basis set increases in size, the basis set superposition error should decrease (by definition). By extension, the magnitude of the counterpoise correction should also decrease. This can be seen from a study of alkylation of dimethylacetamide enolate anion by methyl chloride.

Table 1. Effect of basis set on B3LYP relative electronic energies for the reaction of dimethylacetamide enolate anion and Methyl Chloride[a]

	6-31G(d)	6-31+G(d)	6-311+G(d,p) //6-31+G(d)	aug-cc-pVDZ //6-31+G(d)
reactant complex	-13.6 (-10.4)	-10.0 (-9.7)	-10.1 (-9.8)	-10.2 (-9.8)
alkylation TS	-11.3 (-6.7)	-5.9 (-5.3)	-6.1 (-5.4)	-6.3 (-5.4)
product complex	-74.8 (-71.9)	-68.5 (-68.4)	-67.5 (-67.1)	-68.2 (-67.8)
basis functions	*146*	*178*	*250*	*287*

[a]Energies in kcal/mol relative to infinitely separated reactants; counterpoise corrected energies in parenthesis.

15. IRC Calculations

note: this section substantially revised June 25, 2007

IRC calculations allow you to follow a reaction from the transition structure to the reactant (or reactant complex), and from the transition structure to the product (or product complex). Thus to map out the entire reaction coordinate you will need to run two IRC calculations (arbitrarily designated as forward and reverse), starting from the same transition structure.

Thus the first job is to get the transition structure. Once this is obtained it is easy to construct the input file.

```
%mem=4GB
%scr=h0462
%rwf=h0462
%int=h0462
%d2e=h0462
%nosave
%chk=h0462
#b3lyp/6-31G(d) irc=(RCFC,reverse,maxpoints=20,stepsize=10)
geom=checkpoint

IRC calc based on h0441 MeBr alkylation TS of concave enolate
(simplified).  geom from checkpoint

-1 1
```

So in this example I have taken an enolate alkylation transition structure (h0441), copied its checkpoint file to h0462.chk, and used h0462.chk as the source of the initial geometry and the force constants (note RCFC option in route section). Similarly, the forward IRC input file was constructed in the same way:

```
%mem=4GB
%scr=h0461
%rwf=h0461
%int=h0461
%d2e=h0461
%nosave
%chk=h0461
#b3lyp/6-31G(d) irc=(RCFC,forward,maxpoints=20,stepsize=10)
geom=checkpoint

IRC calc based on h0441 MeBr alkylation TS of concave enolate
(simplified).  geom from checkpoint

-1 1
```

Note that a priori one does not know which "way" a "forward" IRC calculation will proceed from the transition structure. However, in this case, the "forward" IRC calculation did go to the intended products.

In my experience the most critical factor for success in IRC calculations is setting the step size. You have to play with this. If the step size is too big, I have seen at least two unwanted things happen

1) you won't seem to progress along the reaction coordinate—this is confusing, because you will think you need to further increase the step size—which is tragically deceptive.
2) when you further increase the step size, crazy stuff starts happening.

For example, in the present case with a stepsize of 10, this IRC runs smoothly in the forward and reverse directions. With a stepsize of 50, this IRC runs only in the forward direction and the first point is UP in energy from the transition structure. It will not go anywhere productive in the backward direction with this large stepsize, even if the phase is specified (bond axis for movement). With a step size of 25, it runs in the forward direction, but again the energy of the first step is UP (not as much as when it was 50). In both the cases of 50 and 25, the forward IRC gets close to the product, but a vibrating behavior is seen and a stationary point is never reached. An attempt at running the backward IRC with a step size of 100 leads to a vibration, and an explosion. However, in the case of stepsize of 10, the forward and backward IRC go smoothly 20 points towards the product and SM respectively, but never give the message "Minimum found." Nevertheless, both give a successful termination message.

Thus although you may be successful at getting the IRC calclulations to proceed *towards* reactant and product, you may still not be successful at reaching your final destinations ("Minimum found"). Do not dismay. You must first assess how much progress you have made energetically, and to do this the easiest procedure is to take your points of furthest progress, save these geometries, and initiate geometry optimizations. These equilibrium geometries and associated energies will tell you how far away your IRC final geometries are from the goal. At this point you may need to restart the IRC calculation.

To restart, you rename the .gjf file and .chk file (for safety), and increase the number of maxpoints, e.g.

```
%mem=4GB
%scr=h0461r
%rwf=h0461r
%int=h0461r
%d2e=h0461r
%nosave
%chk=h0461r
#b3lyp/6-31G(d) irc=(restart,RCFC,forward,maxpoints=40,stepsize=10)
geom=checkpo
int

continue from h0461; IRC calc based on h0441 MeBr alkylation TS of
concave enolate (simplified).  geom from checkpoint

-1 1
```

In this job the first point will be 21 (since we ended at 20 in the original calculation), and will terminate at point 40. In this system I ended up restarting once for the reverse IRC calculation and twice for the forward IRC calculation. Note that even then, you may not get to a perfect equilibrium geometry; in my case an oscillating behavior ensued for both forward and reverse IRC calculations. However, by paying attention to the energies, it is clear that the reverse IRC calculation was very successful at bringing us to the reactant complex.

File	Calculation	Energy (au)	Rel energy (kcal/mol)
h0477	enolate	-456.8687758	
h0476	CH3Br	-2611.616651	
	total sep. reactants	-3068.485427	0.0
H0441	A\alkylation transition state	-3068.491983	-4.1
h0466	reactant complex	-3068.500544	-9.5
h0462	Reverse IRC	-3068.497843	-7.8
h0462r	Restart of h0462 (lowers slightly, then oscillates)	-3068.498643	-8.2
h0461	Forward IRC	-3068.509155	-14.9
h0461r	Restart of h0461	-3068.541975	-35.5
h0461r2	Restart of h461r (lowers slightly, then oscillates)	-3068.544692	-37.2
h0481	Product complex	-3068.581991	-60.6

As the table shows, the relative energy at the end of h0462r is -8.2 kcal/mol, very close to that of the reactant complex (-9.5 kcal/mol). What is wrong with the forward IRC calculation, where the furthest point is only -37.2 kcal/mol, where as the product complex is -60.6 kcal/mol? Inspection of the structure shows the C-C bond is fully formed and the product is in a good conformation. However, in this case the product is the alkylated enolate and a bromide ion, and the position of the bromide ion plays a huge role in the energy of the product complex. This is shown by deleting the bromide from both h0461r2 and h0481, and taking single point energies. In this case we showed that the organic fragment of h0461r2 was only 2.9 kcal/mol higher in energy than the organic fragment of the product complex h0481. Thus apparently, movement of the bromide to it's favored position is not driven by the same forces that move the IRC from the transition structure towards C-C bond formation.

16. Monitoring IRC Calculations

You must follow what is happening, because failure is common and you don't want to waste your own (and everyone else's) time. Again, *Molden* or *GaussView3* will provide the easiest way to monitor these calculations. Visual inspection of geometries and energies is the key. Note that in *Molden* all one can assess is how many points have produced. Points are not the same as steps; a step moves you on the reaction coordinate a certain amount (governed by stepsize). However after each step is taken a constrained optimization follows, and this may take or or more points. In *Molden* if one looks at geometric convergence you can often see the energy go down in increments, and the bigger increments most often correspond to these steps. Note that the first step of an IRC calculation may actually be slightly higher in energy than the TS. Don't abort the IRC calculation too quickly—give it a chance to see if the energy starts going down.

Once the number of steps reaches the set maximum the calculation will quit. To see what step you are on from the command line interface, type

```
      egrep Step filename.out
```

geometries on the reaction coordinate you have found

```
      egrep Optimized filename.out
```

In the case of h0461, the IRC calculation terminated normally at the 20th step, even though a minimum was not found. Using the egrep commands listed above on h0461.out you would find 20 steps and 20 optimized geometries.

To see whether the energy is decreasing as desired, the easiest way is to use *Molden* or *GaussView3*. But if you want, you can generate an .xmol file using ircextract2.awk, and then

```
      egrep Point filename.xmol
```

See Section 17 below for a description of ircextract2.awk. This will give you the energies at each optimized point, which allows you to see if the energies are decreasing as desired.

To learn whether the calculation has found a stationary point, you need to look for " Minimum found on this side of the potential", so

```
      egrep 'Minimum found'
```

Note that if the final step is calculated, you will get the nice quotation even if the last point is not a stationary point. As mentioned above, you can restart the calculation if you ran out of points, and possibly increase maxpoints. But even then, as described above, you may not find the stationary point.

17. Exporting IRC Data

The IRC output will be in one or more .out files. Note that the first structure will NOT be the TS. You need to get the TS from the original optimization. Again, I think that *Molden* will sometimes be the best way to export structures from an IRC calculation. This can be done by selecting each point of interest, and using the "write" functionality. Nevertheless, there is a command line way to do this, and it is described below. The command line method is at present the best way to get the energies and structures out for the purpose of making a movie. I am hoping that *GaussView4* will make this process easier.

A. Use a script to extract coordinates and energies from these files and store them in an .xmol file.

There are two scripts that will be useful (get them from Paul): ircextract.awk and ircextract2.awk. The second is to be used when you have edited gaussian's z-matrix. The syntax is as follows:

```
      awk -f ircextract2.awk h0005a1.out >h0005a1.xmol
```

This takes the data from the output file and stores just what you need in an xmol file.

Note: to look at the unoptimized geometries Polo created another script to do the same thing--it is optextract.awk

It may be useful to use this script to look at the unoptimized structures when the irc calc is going awry.

B. Then use WordPad to create individual .txt files (step #, energy, coordinates).

It may be easier to store these files in the PDB folder of the *PyMOL* folder for later use, but this is not crucial. Open two windows and copy each step into a new .txt file. In this case I took data from the .xmol file and made files h0005a1.txt to h0005a10.txt In future I recommend use of f and r for the forward and reverse IRC calc., e.g. h0007f.out, h0007r.out. The .xmol files would have the same name, and then, the .txt files could be named h0007f1.txt, h0007f2.txt etc.

C. Manually change suffix of all these files to .xyz.

This can be done beautifully if you know DOS and use the Command Prompt

ls is now dir, cd is the same, what you can do is to go to the command prompt (programs -> accessories -> command prompt)

You need to go to the directory of where the files are located, then use the rename command, in this case:

```
rename h*.xyz h*.txt
```

I have been looking for good renaming programs for Mac. So far I have tried *Skooby Renamer* and *R-Name*. Still learning these.

18. Using *PyMOL* to Make Movies

Note this section not revised in a long time. PyMOL can now read .xyz files.

A. Use *PyMOL* to load all the .pdb files (*PyMOL* can't read .xyz).

This is done at the command line, specifying the path

```
load pdb\h0005a1\h0005a1.pdb
```

Load the reverse IRC structures in reverse order
Load the TS structure
Load the forward IRC structures in forward order

Once you've loaded them all, use 'Save Session As' to save the collection of molecules for future use.

Once all of them are loaded, you can turn individual ones off by clicking the button, or type 'disable all' to turn them all off

B. Reorient selected structures, if necessary

Sometimes the coordinate system appears to have changed and the molecules don't overlay correctly. You can then select the first structure and the second structure. Go to Wizard/Pair Fitting and then use CNTRL-Middle Click to select atom pairs--in each case select the one you want to move, and then the one you want to move it to. Select three pairs in all, and then click the 'Fit 3 pairs' button. If it looks good, click done. Then turn off the first structure, turn on the third, and fit the third to the second. Be sure to use the same anchor atoms in each case.

When you are done, 'Save Session'

C. Select your display options

You will want to make all the carbons the same color and set your other display options. Type in at the command line:

```
show sticks, all
set stick_radius, 0.1
set stick_ball, 1
set stick_ball_ratio, 3
color carbon, symbol c
```

If you have a bond-breaking TS, at the TS the non-bonded atom may appear as a cross. To fix this problem with non-bonded atoms

```
show nb_spheres
```

D. Decide on resolution (normally 1024x768)

Polo recommends the following

```
viewport 1024,768
set orthoscopic
```

(this command makes the display and ray tracing programs adopt the same perspective)

E. Decide on which views you want to start with and move to

For example v1 is far away, v2 is rotated far away and v3 is close up

As you get each desired view, type

```
view v1, store  (then continue for v2, v3, as needed).
```

F. Then use scripts to generate an animation and change viewing options.

This is best done in a text editor. The basic command is mvCmd, followed by a frame number, a command, and a filename. Here is the script for my LiNH2 cyclopropyl nitrile deprotonation movie:

```
mvCmd 1, disable h0007a13
mvCmd 1, enable h0007r22
mvSinViewTravel 1-30, v1, v2
mvSinViewTravel 31-60, v2, v3
mvCmd 61, disable h0007r22
mvCmd 61, enable h0007r21
mvCmd 62, disable h0007r21
mvCmd 62, enable h0007r20
mvCmd 63, disable h0007r20
mvCmd 63, enable h0007r19
mvCmd 64, disable h0007r19
mvCmd 64, enable h0007r18
mvCmd 65, disable h0007r18
mvCmd 65, enable h0007r17
mvCmd 66, disable h0007r17
mvCmd 66, enable h0007r16
mvCmd 67, disable h0007r16
mvCmd 67, enable h0007r15
mvCmd 68, disable h0007r15
mvCmd 68, enable h0007r14
mvCmd 69, disable h0007r14
mvCmd 69, enable h0007r13
mvCmd 70, disable h0007r13
mvCmd 70, enable h0007r12
mvCmd 71, disable h0007r12
mvCmd 71, enable h0007r11
mvCmd 72, disable h0007r11
mvCmd 72, enable h0007r10
mvCmd 73, disable h0007r10
mvCmd 73, enable h0007r9
mvCmd 74, disable h0007r9
mvCmd 74, enable h0007r8
mvCmd 75, disable h0007r8
mvCmd 75, enable h0007r7
mvCmd 76, disable h0007r7
mvCmd 76, enable h0007r6
mvCmd 77, disable h0007r6
mvCmd 77, enable h0007r5
mvCmd 78, disable h0007r5
mvCmd 78, enable h0007r4
mvCmd 79, disable h0007r4
mvCmd 79, enable h0007r3
mvCmd 80, disable h0007r3
mvCmd 80, enable h0007r2
mvCmd 81, disable h0007r2
mvCmd 81, enable h0007r1
mvCmd 82, disable h0007r1
mvCmd 82, enable h0006a
mvCmd 83, disable h0006a
mvCmd 83, enable h0007a1
mvCmd 84, disable h0007a1
mvCmd 84, enable h0007a2
mvCmd 85, disable h0007a2
mvCmd 85, enable h0007a3
mvCmd 86, disable h0007a3
mvCmd 86, enable h0007a4
mvCmd 87, disable h0007a4
mvCmd 87, enable h0007a5
mvCmd 88, disable h0007a5
mvCmd 88, enable h0007a6
mvCmd 89, disable h0007a6
mvCmd 89, enable h0007a7
mvCmd 90, disable h0007a7
mvCmd 90, enable h0007a8
mvCmd 91, disable h0007a8
mvCmd 91, enable h0007a9
mvCmd 92, disable h0007a9
mvCmd 92, enable h0007a10
mvCmd 93, disable h0007a10
mvCmd 93, enable h0007a11
mvCmd 94, disable h0007a11
mvCmd 94, enable h0007a12
mvCmd 95, disable h0007a12
mvCmd 95, enable h0007a13
```

The key thing to remember is that you need to enable and disable individual views, The enabling and disabling is best done in a single frame. In frame 1 we are disabling the last view (so that playback is not compromised), then showing the first view, and beginning a rotation.

The movement from view 1 to view 2 takes place over 30 frames (will be 2 seconds), then the movement from view 2 to view 3 takes place over 30 frames.

G. Preview the movie

Copying the text in this script and pasting it into the command line. Hit return and give it a second. Then type movie in the command line, and hit the play triangle on the lower left. Note that you want to clear all extraneous views before doing this.

If you like it, great. If you want to make changes, go ahead. But before playing the new script you need to type

```
mvClear
mClear
mset
```

H. Make your movie frames.

Once you think you've got it, make sure you are at the beginning view of the movie, deselect any extraneous views, and type

```
set ray_trace_frames=1
set cache_frames=0
```

```
mclear
mpng mov
```

This will generate the png files in the *PyMOL* directory.

I. Assemble the movie frames into a movie file

We have not done this in a long time. As of 1/12 I don’t know what the best PC and Mac programs are to accomplish this.

19. Acknowledgements

We thank Professors T. Daniel Crawford and Diego Troya for many helpful discussions and suggestions, but we take responsibility for any and all errors, typographical or conceptual. This manual was compiled in the course of research supported by the National Science Foundation (NSF-0213525 and NSF-0750006).

Appendix 1: Why your calculation failed (not yet updated for g09)

1) If Calculation does not start on giving the qsub command:

- because of a wrong pathway specified in the .sh file.
- Because of the presence of ampersand at the end of nohup command
- If you see your job queued up: You have specified more memory or processors than currently available, or you have surpassed the number of jobs allowed.

If the calculation failed after starting (you will have the *.rwf, *.scr and *.out files in your directory) Commonly occurring errors and causes:

2) Charge and multiplicity card seems defective: Charge is bogus.
Reason: There is an extra space between your title line and your description

3) End of file in ZSymb.
Error termination via Lnk1e in /apps/packages/gaussian03c02/g03/l101.exe
Reason: This means that there is no space after the last geometry line. You need to have at least 1 space after geometry specification.

4) The combination of multiplicity 1 and 39 electrons is impossible.
Reason: check your charge and multiplicity – one of them is wrong!

5) Convergence criterion not met or Convergence failure -- run terminated.
Reason: There is a problem with convergence during geometry optimization. Try opt=tight or opt=vtight

6) Error termination request processed by link 9999. Error termination via Lnk1e in /apps/packages/gaussian03c02/g03/l9999.exe at Sat Nov 15 10:43:07 2008.
Reason: This is also a convergence issue, though I have usually seen this occur after a number of cycles (usually more than a day of computational time) and could also be due to lack of sufficient resources. Try using MaxCycle=N to increase the number of cycles.

7) No Z-matrix found on checkpoint file.
Cartesian coordinates read from the checkpoint file: thfer.chk
FileIO operation on non-existent file.
Reason: This happens if you specify geom=checkpoint but forget to make the .chk file.

8) Symbolic Z-matrix:
Charge = 0 Multiplicity = 1
There are no atoms!
Reasons: 1. You forgot to paste your geometry or specify geom=checkpoint
2. You have specified geometry but have forgotten to put a description.

3. There is an extra line between your command line and description.

Appendix 2: Useful awk scripts

A. unixdosconvert.awk

```
#converts dos formatted files to Unix format
#use format $awk -f unixdosconvert.awk dosformat.file
>unixformatted.file
#the command sub is a simple substitution with the format sub(/what to
look for/what to replace it with)
#\r means carriage return, the $ means every instance of that carriage
return
#print simply reprints the line after alteration
{sub(/\r$/,"");print}
```

B. enantiomer.awk

```
#takes a cartesian coordinate file and inverts it by multiplying x
coord. by -1;thank you Daniel Crawford for writing this code!
{ new = $2 * -1.0;  printf("%s          %15.12f      %15.12f
%15.12f\n", $1, new, $3, $4); }
```

C. Spartan3678.awk

```
#takes a spartan pdb file and generates cartesian coord. from columns 3
6 7 8
/HET/{ print $3 " " $6 " " $7 " " $8}
```

D. ircextract2.awk

```
BEGIN{j=0
at_symbol[1]="H"
at_symbol[2]="He"
at_symbol[3]="Li"
at_symbol[4]="Be"
at_symbol[5]="B"
at_symbol[6]="C"
at_symbol[7]="N"
at_symbol[8]="O"
at_symbol[9]="F"
at_symbol[10]="Ne"
at_symbol[11]="Na"
at_symbol[12]="Mg"
at_symbol[13]="Al"
at_symbol[14]="Si"
at_symbol[15]="P"
at_symbol[16]="S"
```

```
at_symbol[17]="Cl"
at_symbol[18]="Ar"
at_symbol[19]="K"
at_symbol[20]="Ca"
at_symbol[21]="Sc"
at_symbol[22]="Ti"
at_symbol[23]="V"
at_symbol[24]="Cr"
at_symbol[25]="Mn"
at_symbol[26]="Fe"
at_symbol[27]="Co"
at_symbol[28]="Ni"
at_symbol[29]="Cu"
at_symbol[30]="Zn"
at_symbol[31]="Ga"
at_symbol[32]="Ge"
at_symbol[33]="As"
at_symbol[34]="Se"
at_symbol[35]="Br"
at_symbol[36]="Kr"
at_symbol[46]="Pd"
at_symbol[47]="Ag"
at_symbol[48]="Cd"
at_symbol[50]="Sn"
at_symbol[51]="Sb"
at_symbol[53]="I"
at_symbol[54]="Xe"
at_symbol[64]="Pt"
at_symbol[65]="Au"
at_symbol[66]="Hg"
at_symbol[67]="Tl"
at_symbol[68]="Pb"
at_symbol[69]="Bi"
}
{
 if ($1=="Standard" && $2=="orientation:") {
   getline
   getline
   getline
   getline
   getline
   i=0
   while (NF!=1) {
      i++
      at[i]=$2
      if (NF==5) {
        x[i]=$3
        y[i]=$4
        z[i]=$5
        }
      else {
        x[i]=$4
        y[i]=$5
        z[i]=$6
        }
      getline
      }
   }
```

```
  if ($1=="SCF" && $2=="Done:") {
    energy=$5
    }
  if ($3=="Threshold") {
    getline
    if ($5=="YES") uno=1
      else uno=0
    getline
    if ($5=="YES") dos=1
      else dos=0
    getline
    if ($5=="YES") tres=1
      else tres=0
    getline
    if ($5=="YES") cuatro=1
      else cuatro=0
    if (uno==1 && dos==1 && tres==1 && cuatro==1) {
      j++
      print i
      print "Point ",j, " Energy= ",energy
      for (k=1;k<=i;k++) {
        print at_symbol[at[k]], x[k], y[k], z[k]
        }
      }
    }
}
```

E. optextract.awk

```
BEGIN{j=0
at_symbol[1]="H"
at_symbol[2]="He"
at_symbol[3]="Li"
at_symbol[4]="Be"
at_symbol[5]="B"
at_symbol[6]="C"
at_symbol[7]="N"
at_symbol[8]="O"
at_symbol[9]="F"
at_symbol[10]="Ne"
at_symbol[11]="Na"
at_symbol[12]="Mg"
at_symbol[13]="Al"
at_symbol[14]="Si"
at_symbol[15]="P"
at_symbol[16]="S"
at_symbol[17]="Cl"
at_symbol[18]="Ar"
at_symbol[19]="K"
at_symbol[20]="Ca"
at_symbol[21]="Sc"
at_symbol[22]="Ti"
at_symbol[23]="V"
at_symbol[24]="Cr"
at_symbol[25]="Mn"
at_symbol[26]="Fe"
at_symbol[27]="Co"
at_symbol[28]="Ni"
at_symbol[29]="Cu"
at_symbol[30]="Zn"
at_symbol[31]="Ga"
at_symbol[32]="Ge"
at_symbol[33]="As"
at_symbol[34]="Se"
at_symbol[35]="Br"
at_symbol[36]="Kr"
at_symbol[46]="Pd"
at_symbol[47]="Ag"
at_symbol[48]="Cd"
at_symbol[50]="Sn"
at_symbol[51]="Sb"
at_symbol[53]="I"
at_symbol[54]="Xe"
at_symbol[64]="Pt"
at_symbol[65]="Au"
at_symbol[66]="Hg"
at_symbol[67]="Tl"
at_symbol[68]="Pb"
at_symbol[69]="Bi"
}
{
 if ($1=="Standard" && $2=="orientation:") {
   getline
   getline
   getline
   getline
   getline
   i=0
   while (NF!=1) {
      i++
      at[i]=$2
      if (NF==5) {
        x[i]=$3
        y[i]=$4
        z[i]=$5
        }
      else {
        x[i]=$4
        y[i]=$5
        z[i]=$6
        }
      getline
      }
   }
  if ($1=="SCF" && $2=="Done:") {
    energy=$5
    j++
    print i
    print "Point ",j, " Energy= ",energy
    for (k=1;k<=i;k++) {
      print at_symbol[at[k]], x[k], y[k], z[k]
      }
    }
}
```