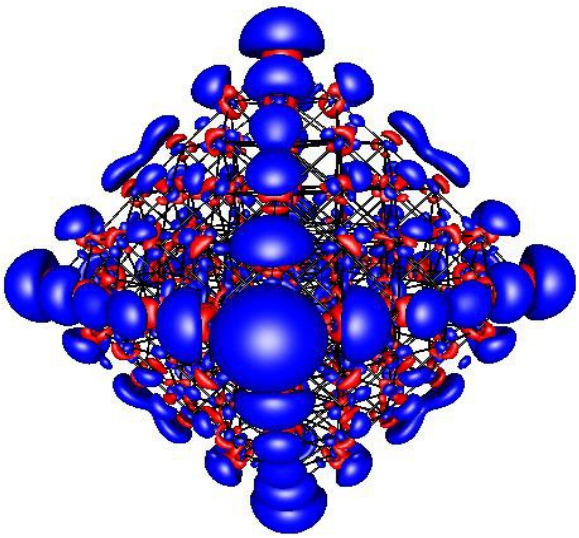
TURBOMOLE



Tutorial V7.0

MAY 2015

Table of Contents

1. TURBOMOLE Usage Philosophy

The usage of TURBOMOLE has been adapted to the way a UNIX user is working:

- Command line driven,
- Many different programs, each one specialized for methods and/or properties,
- Scripts are used to combine the functionalities and manage workflows,
- Input can be changed by text editors,
- Output processed by standard UNIX tools (editors, grep, awk, ...).

Hence: some UNIX skills are required!

The functionalities are not crammed into a single program, user interface, or tool, but they are separated in modules. The user has to know the most important modules and the way they interact via the input and output files and this is not the only difference to most other quantum chemistry programs:

Most other programs:	TURBOMOLE:
• a single input **file** per job like *job1.inp*	• one **directory** per job
• containing: – coordinates – basis set definition or basis sets – method – kind of job (energy, structure, properties,...)	• containing: *control* file with references to external files like: – coordinates – basis sets – molecular orbitals – old results and data – ... • **not** containing: – method – kind of job

• start job by calling **one program**:
`qcprog job1.inp > job1.out`
	• start job by calling one or a **sequence of programs**:
	`dscf > dscf.out`
	`aoforce > aoforce.out`

• all calculations and prerequisites are done in **one step**	• just **one type of calculation per call**, no automatic run of energy calculation or geometry optimization or any other kind of pre-step will be performed
• output contains the results	• control file and output contain the results
• **input** file remains **unchanged**	• **input** files are **overwritten**, i.e. coordinates are changed, mos are changed,...

2. How to create the input

If TURBOMOLE is located at /my_disk/my_name/TURBOMOLE, you can get access to all scripts and programs by executing the following three commands in a (bash-like) shell:

```
export TURBODIR=/my_disk/my_name/TURBOMOLE
export PATH=$TURBODIR/scripts:$PATH
export PATH=$TURBODIR/bin/`sysname`:$PATH
```

2.1. Coordinates

First of all, you need a 3D structure of your molecule:

1. Build

   The TURBOMOLE module that interactively generates an input (details are described below), **`define`**, has some limited features to build a structure from scratch, but that is neither convenient nor intuitive, except for some small cases, especially if you apply symmetry. The graphical user interface TmoleX contains a molecular builder and is available for free from the COSMO*logic* web site.

   1. Use TmoleX to import structures, modify them or build from scratch, define a complete input for TURBOMOLE, submit jobs to the local machine or to remote systems, get back the results and analyze the data.

   2. Use an external builder (molden, ECCE, Hyperchem, Insight, Arguslab, MAPS, Accelrys DS Visualizer,... there are many!) and convert the structure to xyz format.

   3. Get the structure from a database or the internet and modify it by **`define`** or the builder of your choice.

2. Convert

   Use the TURBOMOLE script **`x2t`** to convert xyz files to TURBOMOLE coordinates:

   ```
   x2t struct.xyz > coord
   ```

The script `calculate` can read in xyz, sdf, ml2, car, and cosmo files, while TmoleX is able to import a wide range of coordinate types.

2.2. From the structure to the calculation

There are several possibilities to obtain a complete `control` file and run a calculation. These are the most widely used:

1. **`define`**
   a) run **`define`**
   b) run any TURBOMOLE module (program or script).
   You have to start with a ground state energy calculation to get converged orbitals with **`dscf`** or **`ridft`** (exception: **`jobex`** performs an intial energy calculation by default).

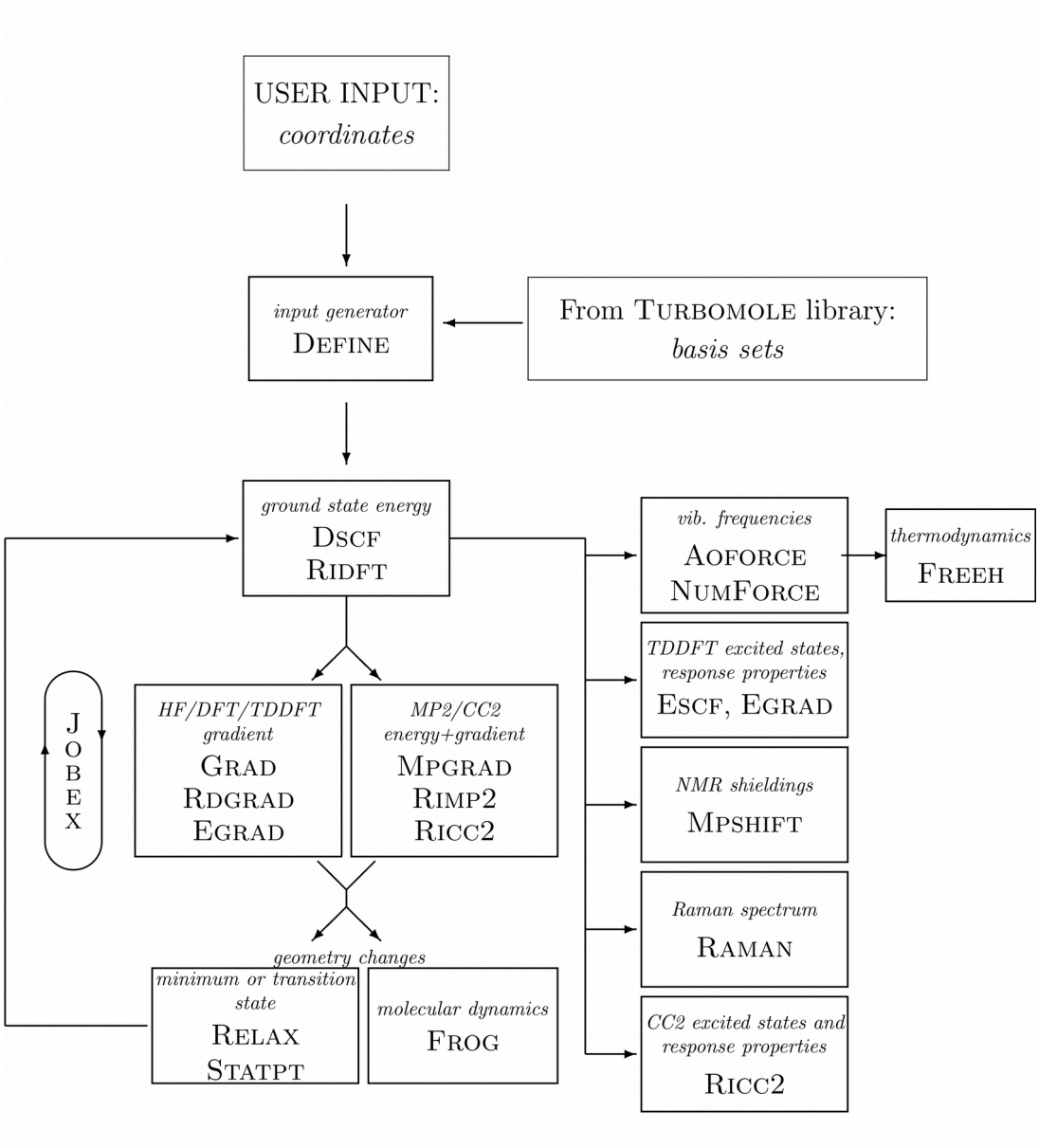
2. **`calculate`**
   generates *.cosmo files for COSMO*therm* input.
   a) make a list of molecules
   b) run COSMO calculations as a batch over all given molecules by using the script `calculate`

3. **`TmoleX`**
   TmoleX is a free add-on to TURBOMOLE by COSMO*logic*. A version for Linux/PC, MacOSX, and Windows is available from the ftp server where the usual TURBOMOLE distribution is located (ask your TURBOMOLE administrator). There is also a free client version which can be used to build molecules, generate input files and submit TURBOMOLE jobs to remote machines. Check The COSMO*logic* home page for details about the client version.

   TmoleX can import coordinates, change structures, generate input files and run TURBOMOLE jobs. Results are shown in a panel within TmoleX. Structures, geometry optimizations and vibrational frequencies can be displayed and animated. See the web site: TmoleX, the Turbomole GUI

In this tutorial we are using **`define`** and the command line to be able to exploit *all* features of TURBOMOLE – all other types of input generation offer only a limited functionality.

3. The TURBOMOLE modules



See chapter 1.4 'Modules and Their Functionality' of the TURBOMOLE documentation for a description of each module and script. Please note that not all available modules are displayed in this picture.

4. The quick tour: How to run a calculation

Creating an input: The shortest way through define

As we have seen in the previous chapters, there is a special program that creates a complete TURBOMOLE input: **`define`**

Most of the functionalities of **`define`** are not needed in your day-to-day work. In general, hitting `<ENTER>` will print out the current menu and `*` will proceed to the next step. And most of the time you may just accept all defaults.

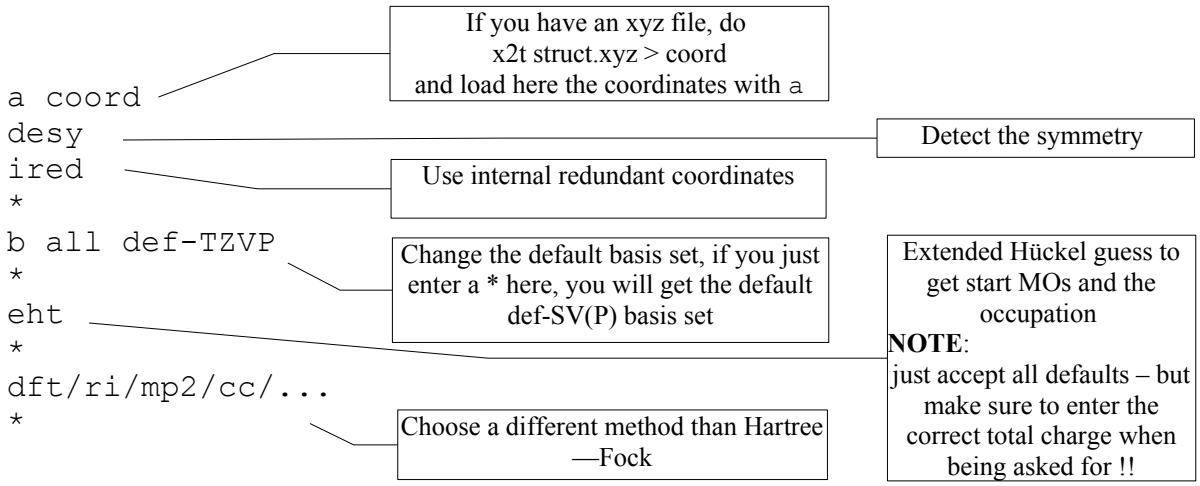
In the first three menus, the geometry is loaded, basis sets are assigned and the start MOs together with the occupation is created. If you do not change anything in the last menu, you will get a nice Hartree-Fock input. Entering `dft`, `ri`, `mp2`, `cc`, ... in the last menu will switch `on` or `off` other methods.

So, the shortest way to an input is to get a structure, either in TURBOMOLE coordinates (file *`coord`*), or in an xyz file (*`struct.xyz`*) and then let the **`x2t`** script of Turbomole convert it to a *`coord`* file:

```
x2t struct.xyz > coord
```

Then create a new **empty** directory and copy the *`coord`* file in there. Change to that directory and call **`define`**. First, you will be asked for a name of the input file, just hit `<Enter>` here, and then you can enter a title. The title can be just an empty line (it might be a good idea to give a title here, sometimes the fastest way to see what the input is about).

The most important commands in **`define`** are explained here. The `*` indicates where each one of the four main menus does end and the next one begins, since `*` or `q` are the commands to end a menu and proceed to the next (see below).



After that, you have a nice Turbomole input and you can run any of the Turbomole modules or scripts – depending on which property you want to calculate.

Creating the input and run a calculation

First exercise!

Instead of starting from an xyz file, we simply take one of the structures from the TURBOMOLE structure library, that is part of every TURBOMOLE distribution. We will do that for almost all other examples that will follow later...

1. Create the input with **`define`**:

- start from a new or empty directory,
  call **`define`**
  and enter the following things:

```
<Enter>
This is the first exercise, where we do a HF calculation on benzene
a ! benzene
y
desy
ired
*
*
eht
<Enter>
<Enter>
<Enter>
*
```

All commands given above will be explained later in this tutorial. The (almost) same exercise is done in chapter 6.1 - with a detailed description of the in- and the output. If you have difficulties with this example, please try chapter 6.1 !!

Some notes:

You can skip the lengthy title, of course, or just hit `<Enter>` for not providing a title at all.

Molecules from the structure library are loaded like any other coordinate files, the only difference is that there is an exclamation mark '`!`' between the command `a` and the name of the molecule.

We have let **`define`** determine the symmetry with `desy`, and **`define`** found D_{6h} point group.

Internal redundant coordinates are determined automatically with `ired` – this would not have been necessary here, but if `ired` is not called, you will be asked if you are sure not to use internal coordinates when exiting the geometry menu (the first `*`).

We have accepted the default basis set def-SV(P) – this is one of the TURBOMOLE default (def-) basis sets, SV(P) is a double zeta valence basis set with polarization at all non-Hydrogen atoms.

`eht` runs an extended Hückel calculation to create start orbitals and to determine the occupation/multiplicity, the rest of the `<Enter>` keys belong to questions that Hückel asks you. Note that one of those questions is the total charge of the molecule (default is 0 – neutral), and another question is if you accept the found occupation (closed shell system, trivial in this case).

Finally, the last `*` exits **`define`**, changing nothing at the last menu – hence we have a standard Hartree-Fock input.

2. Run a Hartree-Fock energy calculation:

   ```
   dscf > dscf.out
   ```

   After the calculation has finished (takes about 3 seconds), please look at the output file *dscf.out*:

   You will see that 9 iterations have been done, the total energy is -230.51807365370 Hartree (convergence is 10^{-7}, so depending on your system/hardware, your result might differ from the one given here).

That's it.

See chapter 6 for details about single point calculations.

Second exercise!

Let us do the same thing as before – but this time we will do a geometry optimization with RI-DFT.

Now we have two different possibilities:

1. Start from scratch, i.e. from an empty directory, and go through **define** as before. But this time, do not enter the last `*`. So you will stay in the last, the general menu:

   ```
    GENERAL MENU : SELECT YOUR TOPIC
    scf    : SELECT NON-DEFAULT SCF PARAMETER
    mp2/cc2: OPTIONS AND DATA GROUPS FOR MP2, CC2, ETC.
    ex     : EXCITED STATE AND RESPONSE OPTIONS
    prop   : SELECT TOOLS FOR SCF-ORBITAL ANALYSIS
   ...
   ```

   Enter `dft` – and you will get in the DFT menu. Just enter `on` here and leave with `<Enter>` or `*`.
   Enter `ri` – and you will get in the RI-J menu. Like in the last menu, enter `on` here and leave again.
   Now enter `*` or `q` to leave **define**.

2. Rerun **define** in the same directory as before, keeping all files in there (including the output files like *dscf.out*, *energy* and the changed input files like *mos*).
   define will tell you that it found an input file which will be read in.

   Now you will be asked if you want to enter each of the three basic menus for geometry, atomic and molecular attributes. Just hit `<Enter>` all the time, since we do not want to change anything except the method.

   define will automatically stop in the last menu – no matter how often you are hitting `<Enter>` now...

   Here enter the same commands as given above (`dft` to `on` and `ri` to `on`) and leave define with `*` or `q`.

We have an input for benzene, DFT (standard) B-P86 functional, using RI-*J* and D_{6h} symmetry. You can have a look which density functional will be used with the **sdg** tool, that prints data groups of the *control* file.

```
sdg dft
```

gives:

```
$dft
```

```
functional b-p
gridsize   m3
```

A single point calculation could be done by starting **ridft**, but we want a geometry optimization, so we enter:

```
jobex
```

jobex will do 4 geometry cycles until convergence. To learn more about **jobex** and structure optimizations please refer to chapter 7.

There are several ways to check the results, but the most common ones are:

1. look at the file *job.last* which contains the output of the last energy and gradient step
2. look at the file *GEO_OPT_CONVERGED* which contains some collected information about the run
3. look at the file gradient: it contains all geometries and gradients at each step of the optimization. A common shortcut is to do

   ```
   grep cycle gradient
   ```

   to check the convergence behavior during the optimization and to get the final result (last energy and gradient norm).
4. call

   ```
   t2x > final_structure.xyz
   ```

   and load the file *final_structure.xyz* in the molecular viewer of your choice (it is a multiple-xyz file)
5. or call **tm2molden** to get a **molden** input file that includes also the molecular orbitals.
6. at the command line, use `dist`, `bend`, or `tors` to get bond lengths, angles, dihedral angles, etc.

   ```
   dist
   ```

   gives:

   ```
   dist  1  c --  7  h =  2.0845 au = 110.30 pm
   dist  2  c --  8  h =  2.0845 au = 110.30 pm
   dist  3  c --  9  h =  2.0845 au = 110.30 pm
   dist  4  c -- 10  h =  2.0845 au = 110.30 pm
   dist  5  c -- 11  h =  2.0845 au = 110.30 pm
   dist  6  c -- 12  h =  2.0845 au = 110.30 pm
   dist  1  c --  2  c =  2.6570 au = 140.60 pm
   dist  1  c --  6  c =  2.6570 au = 140.60 pm
   dist  2  c --  3  c =  2.6570 au = 140.60 pm
   dist  3  c --  4  c =  2.6570 au = 140.60 pm
   dist  4  c --  5  c =  2.6570 au = 140.60 pm
   dist  5  c --  6  c =  2.6570 au = 140.60 pm
   etc.
   ```

   Call

   ```
   dist -help  ,
   bend -help  , or
   ```

```
tors -help
```

to see the available options and syntax.

7. To see how a bond, a bending angle or a torsion changed during the geometry optimization, use the script evalgrad. Call evalgrad with two atom numbers for a bond length, three atom numbers for an angle and four atom numbers to see how a torsion has changed during optimization. The output shows three colums: the optimization step, the energy at this geometry and the value of the bond length, angle or torsion:

```
> evalgrad 1 2
 1 -232.0725369223  1.3892
 2 -232.0764471780  1.4034
 3 -232.0765625911  1.4060
 4 -232.0765631365  1.4061
```

5. DEFINE

define is an *interactive* input generator which creates the input file *control*:

1. Supports most basis sets in use, especially the only fully atom optimized consistent basis sets of SVP, TZV, and QZV quality available for the atoms H–Rn (including lanthanides and actinides up to Lawrencium for SVP and TZV),
2. Determines the molecular symmetry,
3. Determines the internal coordinates, allowing for efficient geometry optimization,
4. Allows to perform a geometry optimization at a force field level to pre-optimize the geometry and to calculate a Cartesian Hessian matrix,
5. Sets the keywords necessary for single point calculations and geometry optimizations for a variety of methods,
6. Manipulate geometries of molecules.

Just try and see how it works...

Notes about how to proceed in this tutorial:

The following chapter explains the usage of **define** to get a proper input for a calculation. Since **define** is not just a simple input preparation program but is also able to analyze, change, build, and compute properties of your input, the chapter is quite long – and you will not do a single real calculation until you have reached the near end of this chapter...

For impatient users or users who want to know how to perform a certain calculation:

> Please read chapters 5.1 and 5.2 and then go directly to chapter 6: Single point calculations or to one of the other chapters ...

5.1. The path through define

There are four major menus in define:

1. Molecular Geometry Menu
2. Atomic Attribute Definition Menu
3. Occupation Number & Molecular Orbital Definition Menu
4. General Menu

But before you get to the first one, you will be asked for:

1. The name of the input file to read in and/or to be generated. The default is *control*

   Always accept the default here!

2. A title for the job you want to create an input for:

   That can be an arbitrary string, but it will be truncated to 80 characters automatically.

5.2. Navigation within define

Some general remarks:

1. You will always get the current menu printed on screen (enlarge your screen if you can not see everything).
2. The first line contains the most important data of your input for the current menu.
3. Most of the commands lead to sub-menus.
4. To finish the menu and proceed to the next one, enter:

   ```
   *
   ```

5. To go back to the previous menu, enter:

   ```
   &
   ```

   (but please do not do that too often...)

6. If you are in one of the four main menus, hitting <Enter> will just print out the menu you are in.
7. To quit **define** immediately (and prematurely), enter:

   ```
   qq
   ```

   which only gives proper input files if you have been reading in an existing input.

5.3. Making symmetric molecules with define

Skip this chapter if you are not going to build molecules with the `define` module of TURBOMOLE.

`define` is probably not the most convenient tool for coordinate inputs. For highly symmetric cases, however, it provides some nice features. Here are some examples:

- Cyclopentadienyl anion (Cp^-, point group D_{5h}):

In the geometry menu of define, we start with defining the point group. Enter

```
sy d5h
```

Maybe you have to use the scroll bar of your window to see the following lines:

```
symmetry group of the molecule :   d5h

the group has the following generators :
  c5(z)
  c2(x)
  mirror plane sigma(xy)

  20 symmetry operations found
```

This tells us, that the molecule must lie in the xy-plane and the symmetry distinct atoms of Cp^- should be on the x axis. To define the molecule, we enter the following commands

```
ai
c
1.2 0 0 A   # Angstrom units!
*
h
2.3 0 0 A
*
*
```

Now we can have a look at the results with

- `disc`: displays the Cartesian coordinates
- `disb 1,6`: displays the bond distances for the two symmetry distinct atoms (1 c and 6 h)
- `disg`: calls a visualizer, needs some customizing by your system administrator; does probably not work properly if define runs on a remote machine.
- `disi`: displays internal coordinates (not yet defined, see next exercise).

We can save the current status of our project coordinates by typing

```
w coord.cp
```

(or any other file name, of course). If we don't like the results, we just delete everything by entering

```
del all
```

and confirming with y. The Cp^- example continues in the next exercise.

- P_4 (point group T_d): Proceed as described above
- PF_5 (D_{3h})
- $B_{12}H_{12}^{2-}$ (I_h)
- C_8H_8, Cubane (O_h)

... Think of other examples.

Hint: Be always aware how the symmetry generators of the chosen point group are defined.

5.4. Internal coordinates

The definition of internal coordinates is extremely important for efficient structure optimizations.

If you are not interested in any property related to internal coordinates, it is usually safe to define the internal redundant coordinates automatically by `ired` and then proceed to the next menu with `*`. In that case, just keep in mind to call the `ired` option in the first **`define`** menu and go to the next chapter.

However, if you want to do constrained searches, or if you are looking for transition states, you need to know how to deal with internal non-redundant coordinates as well:

- Non-redundant internal coordinates (manually):

  Assume, we want to continue with our Cp^- example – if you did skip the last menu, never mind... you can use the coordinates from the structure library.

  Call **`define`** in an empty directory and then type (ignore the comments behind and including the #):

  ```
  <Enter>        # two or more times - until you get to the geometry menu
  a ! cp-        # load the coordinates from the structure library
  ```

  restore the point group information by using the command

  ```
  desy
  ```

  We can now define the coordinates manually by issuing the following commands (ignore the comments):

  ```
  i              # you can skip this, it only serves
                 # for displaying the internal coord. menu
  idef
  k stre 1 2     # c-c distance
  k stre 1 6     # c-h distance
  ```

  After another two strokes on the return key, we are back in the internal coordinate menu. We can enter `imet` to find out whether the coordinates are complete and have a decent Wilson B-matrix. Now, also the `disi` command is working. We can play a little with internal coordinates and reshape the molecule (often the distances are not satisfyingly after input of guessed Cartesian coordinates). Use the `iman` command to change the bond lengths (but remember to reset them to chemically reasonable values before going to the next item of this exercise). If you are doing the „quick tour“ through the tutorial go to exercise 5.6.

- Non-redundant internal coordinates (automatically):

  Except for small cases, manual input of internal coordinates is a rather joyless task. Think of the internal coordinates necessary for Cp^- in C_1 symmetry. Delete the internal coordinates of the previous part of the exercise by typing

  ```
  irem k
  ```

  and reset the symmetry by

  ```
  sy c1.
  ```

  Now we can try an automatic assignment with

```
iaut
```

This will bring us to the „BOND ANALYSIS MAIN MENU“.
Hitting the return key will display the automatically detected bonding information. Hit the return key another time and enter a

```
*
```

to get back to them geometry main menu.
Use `disi` to view the result. After that we finish our **`define`** session prematurely by typing

```
*  # after this command we should be in the attributes menu
qq
```

Now have a look at the *`control`* and *`coord`* files (with `more`, `less`, `vim`, etc.).

- Redundant internal coordinates (automatically):

  The

  ```
  iaut
  ```

  command fails for more complicated cases (cage molecules or sandwich complexes).
  Let us create a new directory and copy there the *`coord`* file containing the Cp^- anion. We run `define`, load the molecule, detect the symmetry using `desy` and lower the symmetry to C_{5v} via the command `susy`.
  After that we add an aluminum atom:

  ```
  ai
  al
  0 0 2 A
  *
  *
  ```

  If we try to use `iaut` on this molecule, **`define`** will fail (you will see this, if you try `imet`). In cases like these we can use

  ```
  ired
  ```

  which produces redundant internal coordinates. You can see the result by typing `red_info` (and using the scroll bar of you window). For the current molecule, due to its high symmetry, there still is a good solution using non-redundant internal coordinates (can you think of one?). However, if symmetry is lowered and/or the molecule becomes more complex, the Wilson B-matrix will become ill-defined for non-redundant internal coordinates.

Lets quit that project for a moment. Enter

```
w coord
qq
```

You can now have a look at the *`control`* file. It contains the redundant internal coordinates in the data group `$redundant`.

Note: you can delete files like *`tmp.input`*. They may remain after premature exits from **`define`**.

5.5. Detecting (unknown) symmetry

Crystal data or final structures from molecular dynamics runs (and even results of geometry optimizations) sometimes are only slightly distorted and will show high symmetry, if further optimized. **define** can help you to detect these „hidden“ symmetries. You have to use the

```
desy
```

option of the geometry main menu, together with a larger threshold:

```
desy 0.5
```

This threshold gives the size of a sphere around each atom (in Bohr units). If spheres overlap after a symmetry transformation, the two atoms are considered as symmetry equivalent. After a higher symmetry has been detected, the molecular framework is symmetrized.

There are three examples in the structure library.

1. Call **define** from an empty directory, and
2. hit <Enter> two times to get to the SPECIFICATION OF MOLECULAR GEOMETRY menu.
3. Add one of the example input coordinates with
   `a ! name`
   where *name* is one of `c4h6.c2h` (try to find C_{2h}), `ptcl4` (almost D_{4h}), or `mg17` (search for D_{4d}),

Use `desy` to find the molecules secrets.

Hints: If the radius of the spheres was chosen too small such that only a subgroup of the full symmetry group was identified, the symmetrization may prevent **define** from finding the full symmetry. Sometimes, however, the reverse is true. Use the `w <filename>`, `del all` and `a <filename>` commands to save intermediate results, delete all atoms and to reload the coordinate files.
Of course, a too large value for the sphere is not very helpful, either. Best results are typically obtained with `desy 1.0`.

5.6. Assigning basis sets

Lets resume with the Cp^- example (exercises 5.3, 5.4). Go to the previous directory and start **define**. If you left it in the status after the last exercise you might, after the title section, get a message like

```
SYMMETRY c1   AND CARTESIAN COORDINATES FOR  10 ATOMS
HAVE BEEN READ FROM THE DEFAULT INPUT FILE control .
DEFINITIONS OF   24 INTERNAL COORDINATES HAVE BEEN READ.
SPECIFICATION OF BOND TOPOLOGY HAS BEEN READ.
DO YOU WANT TO CHANGE THE GEOMETRY DATA ?  DEFAULT=n  GOBACK=&
```

If it says `SYMMETRY d5h` everything is OK and we answer `n`.
Else we answer `y` and restore the symmetry with `desy`. After leaving that menu with `*` we find ourselves in the basis set menu. Automatically, **define** will assign the `def-SV(P)` basis sets. If we want something else, we may enter e.g.

```
b all def2-TZVP
```

All atoms now are assigned the `def2-TZVP` basis. Use the `bl` command to look at the current status. If we feel that we should rather use the `def-SVP` basis for the hydrogens, we enter

```
b "h" def-SVP
```

Use `bl` to check that this works.

Symmetry is maintained in this section of **`define`**, so

```
bb 7 def2-TZVP
```

has the effect of assigning the `def2-TZVP` basis to *all* symmetry equivalent hydrogen atoms.

At this point we leave this menu by entering `*`. To immediately quit define, enter `qq`.
If you want to continue with exercising, please go to chapter 5.11. To learn more about basis sets and basis set assignments, simply continue.

If you are interested in some more advanced features like basis set modifications, entering `h` will provide some help.

5.7. Which basis set should I take?

The question which basis set is the best for your purposes depends on which method you apply and what you are going to do. First of all, we recommend to use the Karlsruhe (Ahlrichs) basis sets – not just for TURBOMOLE. There is a complete set of highly optimized basis sets at double zeta (split valance – SV), triple zeta (TZV), and quadruple zeta (QZV) quality available.

The default basis set which define sets automatically is def-SV(P), i.e. def-SVP for all non-hydrogen elements and def-SV for the H's (no polarization function P there). This basis set is sufficient for ground state DFT structures and some properties, but should be set to a higher level (def2-TZVP) for DFT single point energy calculations or Hartree—Fock and MP2/CC2 calculations.

A short note about the names and the quality of each basis set for the methods implemented in TURBOMOLE is given, when you call the basis set information sub-menu (`bi`) in the `ATOMIC ATTRIBUTE DEFINITION MENU`.
Enter `bi` and you will get this message on screen:

```
bi

 ================================================================================

 Information about improved TURBOMOLE basis sets:

 Balanced basis sets for H-Rn of types SV, TZV and QZV (plus polarization sets)
 were developed in 2005, partly by modifying previous bases.
 The resulting bases, def2-SV(P)/SVP/TZVP/TZVPP/QZVP/QZVPP,
 were tested at more than 300 representatively chosen systems.
 We thus may give some rough recommendations:

 QUALITY:  exploratory-----qualitative-----quantitative-------limit
 DFT                        def2-SV(P)    def2-TZVP       def2-QZVP
 HF                         def2-SVP      def2-TZVPP      def2-QZVPP
 MP2          def2-SVP      def2-TZVPP    def2-QZVPP

 For further details see: F. Weigend and R. Ahlrichs,
  PCCP, 7, 3297-3305 (2005).

 ================================================================================
```

Note that the former standard basis sets for TURBOMOLE are still in the basis set library, just replace def2- by def-, e.g. def-SVP instead of def2-SVP.

Another set of basis sets, typically used for correlated methods like Coupled Cluster calculations, are Dunning's basis sets. Those are included in the standard Turbomole basis set library, denoted as usual:
cc-pVDZ, cc-pVTZ, cc-pVQZ, cc-pV5Z, cc-pV6Z. And the augmented ones: aug-cc-pVDZ, aug-cc-pVTZ, etc.

To get an overview about the quality of each basis function compared to the other ones, choose an element and look at the file $TURBODIR/basen/<element name>. At the beginning of the file, the names of the basis sets are given together with the total energy of an unrestricted Hartree-Fock calculation. The lower the energy, the better the basis set.

5.8. How to get a list of available basis sets

There is (quite hidden) an option in **define** that prints out all available basis sets, letting you choose the one you like. It was thought as a help, so you get there only if you do the assignment step by step.

- In the basis set menu enter

```
b all
```

- then you will be asked for a nickname. If you have no idea what to take, enter something like (it does not matter):

```
help
```

- **define** will now search for the string 'help' and gives an error output:

```
 THERE ARE NO DATA SETS CATALOGUED IN FILE
 /usr/local/TURBOMOLE/basen/c
 CORRESPONDING TO NICKNAME c help

 USE ONE OF THE FOLLOWING OPTIONS :
 nick    - REPEAT NICKNAME INPUT
...
```

- Enter `nick` to get to the nick name section

- **define** now asks for a nick name:

```
INPUT NICKNAME
```

- Enter a question mark here:

```
?
```

- you will first get the list of available basis functions (the header of the files in the basis set library), and then **define** goes through all basis set nicknames, asking if you want the current entry or not.

- If you just remember the beginning of the name, or if you want to use one of the default basis sets, you can also enter as nick name:

```
def-?
```

or

```
def2-?
```

And you will get just the basis sets from the library that start with def- or def2-, resp.

Unfortunately, **define** will take the first name of a basis set if several names are given, which might prevent the automatic assignment of auxiliary basis sets for RI later on. But there, you will be able to go through the available auxiliary basis sets either.

5.9. A basis set library of your own

In your home directory, you can create a *.definerc* file (the leading dot is important!). The contents of *.definerc* may look like that:

```
basis=/home/me/PROG/TURBOMOLE/basen
basis=/home/me/BASISSETS
```

Each line defines an alternative basis set library. The user defined basis set library should look like the TURBOMOLE library (have a short look at *$TURBODIR/basen*): The directory must contain files having the name of the element, which in their turn contain the various basis sets. Create such a directory in your home directory. Load one or two basis sets of your choice for carbon and hydrogen from the EMSL basis set library

```
http://bse.pnl.gov/bse/portal
```

and copy the basis set information into files named `C` and `H`. Make sure that you selected the TURBOMOLE format. There should be only one `$basis` at the beginning and one `$end` at the end of each library file. Then create a *.definerc* in your home directory and enter the full path of your new basis set library in the format shown above.

To check whether it is working, go to the directory where you last edited the Cp^- anion and run define. In the basis set menu, enter `lib` and choose your new library. Now you can specify the basis sets that you have just downloaded.

5.10. Geometry manipulation with define

If you feel that your molecule builder is much better, you can simply skip this exercise.

Get a copy of the Cp^- ring, which you hopefully succeeded to create in the previous exercise and turn it into Cp^{*-} (i.e. Substitute the hydrogens by methyl groups). This can be accomplished as follows:

- Create a new directory and call **define**. Enter two times `<Enter>` until you reach the Molecular Geometry Menu.
- Enter `a !cp-` and then `desy`
- If the displayed symmetry group is not D_{5h} use `desy 0.5`. Then use the `susy` command to lower the symmetry group to C_5 (you have to go to C_{5v} first).
- Still in the geometry menu, enter `sub`. Then you have to specify the atom index of one of the hydrogens. Use the `disc` command if you do not remember. The program will ask you whether you want to replace all symmetry equivalent atoms (provided the symmetry was set correctly).
  Answer `y`.
- Now we have to define a molecular fragment that should replace hydrogen. We use the little structure library

of TURBOMOLE again and enter
`!ch4`. We accept the first offered structured and choose one arbitrary hydrogen of methane to be taken away, thus defining the way the fragments are linked together. For the following question, you can accept the defaults.

- After entering the basis set menu, we can simply leave define by entering `qq`.
  Use the `t2x` or the `tm2molden` tool to get the input for a visualizer. Copy this file via FTP to your local machine and hope, that there is something installed to view the result.

5.11. Providing an initial guess for molecular orbitals

This menu serves for the definition of occupation numbers (and thereby the charge of the molecule) and the start orbitals for the SCF calculation. For well-behaved molecules (like our Cp^- anion), the procedure is very easy. Just choose the option

```
eht
```

to perform an extended Hückel calculation. Accept the 'DEFAULT PARAMETERS' and enter the charge (-1 for Cp^-, of course). The program will propose occupation numbers which for most cases are correct. If the HOMO/LUMO separation is small, you will get a warning. For Cp^-, we can accept the assignment and go on to chapter 5.13 .

If you don't like the assignment, entering

```
n
```

brings you to a menu for manual assignment of orbitals. You will get this menu automatically, if **define** cannot establish occupation numbers because of degeneracies of the frontier orbitals. The most important options are:

- `l` to list the MOs (ordering according to their EHT energies).
- `t` to choose a UHF triplet occupation
- `u` to choose a UHF multiplet with `n` unpaired electrons
  e.g. `u 3` for a (high-spin) quartet.
- `c` to declare a list of closed shells,
  e.g. `c 1-20,22`.
- `a` and `b` to declare a list of occupied alpha and beta orbitals (for UHF).
- `o` declares open shells for ROHF (!) runs.
  Do not use this option if you actually want to declare the occupation for a UHF run.

5.12. Reusing old mos

All TURBOMOLE jobs are restart jobs as default. So you will *always* reuse the old molecular orbitals that are present in your working directory. In some cases those are not valid any more and **define** provides the `use` option to recycle MOs when

- you have changed the symmetry (without changing the structure!),
- you have changed the basis set.

The `use` command will need a unchanged copy of the *control*, *basis* and *mos* files; therefore make a copy of the contents of your working directory (e.g. in the subdirectory *save*: `mkdir save ; cp * save`).

- Run **define** in the working directory and change the symmetry to C_2 (use `susy`).
  Leave the geometry menu and skip the attributes menu; you will automatically end up in the molecular orbital section of **define**.
  Enter the command
  ```
  use save/control
  ```
  (you have to modify the path name, of course) to obtain transformed orbitals.

  Quit **define** and use the **eiger** command to view the result.

- Restore the old *control* and *coord* files (symmetry D_{5d}) by copying them back from the subdirectory and rerun **define**.
  This time you change the basis set to TZVPP.
  The command
  ```
  use save/control
  ```
  will generate projected orbitals.

  Run **ridft** and compare with a run which starts from EHT Mos.

5.13. The general menu of *define*

Here, you can choose additional parameters for your calculation. Depending on the type of calculation you wish to start you have to visit some sub menus:

- Hartree-Fock SCF:
  This is the default with a convergence threshold for the energy of 10^{-7} Hartree.
  If you want to change some options, go to the scf sub menu. Most options concern convergence acceleration. They are normally OK for most cases.
- MP2/RI-MP2/RI-CC2/PNO-CC:
  In this menu you can set all RI-MP2 and RI-CC2 related settings – i.e. assigning auxiliary basis sets with cbas, freezing orbitals with freeze, setting the memory that should be used for the calculation with memory, and denconv to get accurate Hartree-Fock densities as input for MP2 and CC2.
  In general, you can use the **ricc2** program for both MP2 and CC2 calculations. Call cc2 instead of mp2 in the general menu and choose the method by entering ricc2. With list models you get a list of available wave function models.
- DFT:
  Enter the dft sub menu, switch it on and select a functional and a grid (if the default is not satisfactory).
  If you want to switch on dispersion correction for DFT, use the dsp submenu to switch it on or off.
- RI-DFT:
  Enter the dft sub menu and make your changes as described above. Subsequently, go to the ri sub menu and switch on this option. You should also increase the core memory; the default is 200 MB. In general, you can use up to about 80-90% of the net memory available per processor (ask your system administrator and check the run time behavior of the program with top).
  To switch on the multipole accelerated version of RI-J, just got to the marij menu (see chapter 6.7 for details about MARI-J).

Some other options will be described later in the tutorial.

Coming back to our Cp^- example: Try to set up the input the input for the following calculations and run a single point energy calculation.

1.	Hartree-Fock SCF (scfconv 8).	Run the **dscf** module.
2.	DFT (try e.g. B3-LYP [func b3-lyp]).	Run the **dscf** module.
3.	RI-DFT (try e.g. PBE [func pbe]).	Run the **ridft** module.
4.	RI-MP2 (with cc2, ricc2, mp2, *, cbas, *, ...)	Run **dscf** and then **ricc2**.

Have a look at your working directory before and after the run. Also inspect the files *control*, *coord*, *basis*, *energy*.

5.14. Manual editing of the control file

Once the *control* file has been set up, many changes to it can also be done using standard UNIX editors (like **vim** or **emacs**).

6. Single Point Calculations

6.1. How to perform a single point calculation

Let us do a Hartree-Fock energy calculation of benzene:

1. Start with a new and/or empty directory.
2. Call **define.**
3. Hit <Enter> when being asked for the name of the input file,
   enter a title for the input file you are going to create, this can be an arbitrary text like:

   ```
   benzene, Hartree-Fock calculation, with symmetry and def-TZVP basis set
   ```

4. Now you will get the first menu of define (here only the first few lines are printed):

   ```
   SPECIFICATION OF MOLECULAR GEOMETRY ( #ATOMS=0     SYMMETRY=c1  )
    YOU MAY USE ONE OF THE FOLLOWING COMMANDS :
    sy <group> <eps> : DEFINE MOLECULAR SYMMETRY (default for eps=3d-1)
    desy <eps>       : DETERMINE MOLECULAR SYMMETRY AND ADJUST
    COORDINATES (default for eps=1d-6)
    susy             : ADJUST COORDINATES FOR SUBGROUPS
    ai               : ADD ATOMIC COORDINATES INTERACTIVELY
    a <file>         : ADD ATOMIC COORDINATES FROM FILE <file>
   ...
   ```

   make sure that your terminal window is large enough to see the first line with the most important data.

5. As you can see in the first line, there are no atoms yet, symmetry is C_1.
6. Load benzene from the structure library (located in $TURBODIR/structures)

   ```
   a !benzene
   ```

   Usually, you would say

   ```
   a coord
   ```

   here at this point, assuming that your coordinates in Turbomole format are in the file *coord* in the local directory. A complete path to a *coord* file can be given here either, like: `a /home/me/infiles/projectA/coord`

7. **define** will scan through the structure library and ask you if you want to take the proposed molecule:

```
  ******     STRUCTURE LIBRARY UTILITY     ******

 ENTRY NO.  1 MATCHING SEARCH STRING benzene :

 formula=c6h6
 file=benzene
 name=benzene
 name=benzol
 # benzene optimized at SCF-level with basis sets of dz quality.
 # max. internal gradient is 2.1d-05 (stre 1 7), total energy is
230.601994.
 # the molecule lies in the xy-plane (for d6h-symmetry)
 # Note: there is one more entry for benzene in this library.

  DO YOU WANT THIS STRUCTURE (DEFAULT=n) ?
```

   just say

   ```
   y
   ```

   here to accept the structure.

8. The main menu is printed again, and if you look at the first line, you will see that 12 atoms have been added:

```
SPECIFICATION OF MOLECULAR GEOMETRY ( #ATOMS=12    SYMMETRY=c1  )
 YOU MAY USE ONE OF THE FOLLOWING COMMANDS :
```

9. Switch on the symmetry detection by entering:

   ```
   desy
   ```

10. Symmetry detection prints out the Schönflies symbol, the generators, and much more. Scroll up your window if you want to see the details.

11. `desy` does its work and the main menu is printed again, now with the right symmetry D_{6h}:

    SPECIFICATION OF MOLECULAR GEOMETRY (#ATOMS=12 SYMMETRY=d6h)
    YOU MAY USE ONE OF THE FOLLOWING COMMANDS :

12. Next typical step is to provide internal coordinates. That is not needed for single point calculations, but since we are already here, and since one never knows if a geometry optimization will follow, and (finally) since you will be asked if you really do not want to use internal coordinates when leaving this menu afterwards, a simple

    ```
    ired
    ```

    is something you should get used to do at this point.

13. Now we are done with this menu and proceed to the next one by entering

    ```
    *
    ```

14. Next menu is the atomic attributes menu, here you usually will just change the basis set. The default basis set,

which is assigned automatically, is def-SV(P) – see chapter 5.7 about basis sets and their quality.

As you can see in the first line of the menu

```
ATOMIC ATTRIBUTE DEFINITION MENU  ( #atoms=12    #bas=12    #ecp=0     )
```

all 12 atoms have a basis set assigned.

We wrote in the title, that we are going to use def-TZVP. To change the basis set accordingly, enter:

```
b all def-TZVP
```

this will assign 'all' elements the basis set 'def-TZVP'.

That is it for this menu, so let us proceed to the next one by entering

```
*
```

15. The third menu where we are in now is the one where **`define`** determines the start molecular orbitals with (e.g.) an extended Hückel guess.

    To do the guess and to choose the correct occupation, just call

    ```
    eht
    ```

    and you will be asked a lot of questions. Just accept all defaults.

    There are TWO important questions during this procedures:

    ```
    ENTER THE MOLECULAR CHARGE  (DEFAULT=0)
    ```

    If you have a charged system, you have to tell define that at this point! For now, we accept the neutral benzene by hitting simply <Enter>.

    and

    ```
    AUTOMATIC OCCUPATION NUMBER ASSIGNMENT ESTABLISHED !
     FOUND CLOSED SHELL SYSTEM !
     HOMO/LUMO-SEPARATION :  0.167139
     ORBITAL  SYMMETRY      ENERGY     DEFAULT
     (SHELL)    TYPE                 OCCUPATION
        11     3e1u        -0.51863      4
        12     1b1u        -0.50821      2
        13     1e1g        -0.48203      4
        14     3e2g        -0.43095      4
        15     1e2u        -0.26381      0
        16     1b1g        -0.08611      0
        17     4e1u         0.39013      0

     DO YOU ACCEPT THIS OCCUPATION ?  DEFAULT=y
    ```

    Here define suggests a certain occupation, which also determines the multiplicity and the choice between restricted and unrestricted calculations. In this case it is a restricted singlet calculation which is just fine.

16. `eht` automatically terminates the occupation menu, so we get directly to the last menu which is called 'general

menu'. Here the settings for the methods (HF, DFT, MP2/CC2, excited states, RI, ...) can be done:

```
GENERAL MENU : SELECT YOUR TOPIC
scf    : SELECT NON-DEFAULT SCF PARAMETER
mp2    : OPTIONS AND DATA GROUPS FOR rimp2 and mpgrad
cc     : OPTIONS AND DATA GROUPS FOR ricc2
pnocc  : OPTIONS AND DATA GROUPS FOR pnoccsd
ex     : EXCITED STATE AND RESPONSE OPTIONS
prop   : SELECT TOOLS FOR SCF-ORBITAL ANALYSIS
drv    : SELECT NON-DEFAULT INPUT PARAMETER FOR EVALUATION
         OF ANALYTICAL ENERGY DERIVATIVES
         (GRADIENTS, FORCE CONSTANTS)
rex    : SELECT OPTIONS FOR GEOMETRY UPDATES USING RELAX
stp    : SELECT NON-DEFAULT STRUCTURE OPTIMIZATION PARAMETER
e      : DEFINE EXTERNAL ELECTROSTATIC FIELD
dft    : DFT Parameters
ri     : RI Parameters
rijk   : RI-JK-HF Parameters
rirpa  : RIRPA Parameters
senex  : seminumeric exchange parameters
hybno  : hybrid Noga/Diag parameters
dsp    : DFT dispersion correction
trunc  : USE TRUNCATED AUXBASIS DURING ITERATIONS
marij  : MULTIPOLE ACCELERATED RI-J
dis    : DISPLAY MOLECULAR GEOMETRY
list   : LIST OF CONTROL FILE
&      : GO BACK TO OCCUPATION/ORBITAL ASSIGNMENT MENU

* or q : END OF DEFINE SESSION
```

The default is a Hartree-Fock calculation, and therefore we do not have to change anything here. Simply enter:

*

17. define ends now and you will find several files in your directory, containing all important settings and data.

18. To run a Hartree-Fock single point energy calculation, call

    ```
    dscf > dscf.out
    ```

19. after the run, look in the output and/or the energy file. The energy file should give:

    ```
    $energy      SCF              SCFKIN            SCFPOT
         1 -230.7733367010      230.6595606167     -461.4328973177
    $end
    ```

    The converged energy is given in the SCF column, kinetic and potential energies are also given.

20. Look in the output *dscf.out* and search for the dipole and quadrupole moment, the number of basis functions, and whatever you like to know.

21. We get more details about the orbitals, the HOMO-LUMO gap and the occupation with the Turbomole own tool

    ```
    eiger
    ```

    Just call it and see what it prints out.

To run a DFT calculation, call **define** again and accept all defaults – it will read in the existing control file, so you will not have to change anything at the first three menus (coordinates, basis sets, molecular orbitals).

In the last menu, enter `dft` and then `on` to switch on this method:

```
STATUS OF DFT_OPTIONS:
 DFT is used
   functional b-p
    gridsize m3

 ENTER DFT-OPTION TO BE MODIFIED

 func    : TO CHANGE TYPE OF FUNCTIONAL
 grid    : TO CHANGE GRIDSIZE
  off:  TO SWITCH OFF DFT
 Just <ENTER>, q or '*' terminate this menu.
```

With <Enter> you will get back to the general menu.

End **define** and run **dscf** again. The energy is now:

```
$energy      SCF              SCFKIN           SCFPOT
    1 -230.7733366808      230.6595789619     -461.4329156428
    2 -232.3278889602      231.4444536342     -463.7723425944
$end
```

Note that energies are always appended to the list of energies!

Next, switch on RI the same way: enter `ri` and then `on`. **define** will automatically assign the auxiliary basis set if you have chosen a basis set where optimized auxiliary basis sets exist (all def- and def2- basis sets).

Run **ridft** and check the energy:

```
$energy      SCF              SCFKIN           SCFPOT
     1 -230.7733366808      230.6595789619     -461.4329156428
     2 -232.3278889602      231.4444536342     -463.7723425944
     3 -232.3286314547      231.4468598575     -463.7754913122
$end
```

Note that the RI and non-RI energies differ: **Never compare absolute energies of RI with non-RI calculations!!**

6.2. Semi-direct runs with **dscf**

Semi-direct runs are optimal for medium-sized SCF calculations (up to 500-1000 basis functions). This example gives you an idea of the computational savings.

1. Get the coordinates of P_{10} from the structure library:

   - call **define** from an empty directory
   - hit two times the <Enter> key
   - say `a !p10` to load the coordinates from the structure library
   - call `desy` and `ired` for symmetry and internal redundant coordinates, resp.
   - and then `*` or `q` to go to the next, the basis set menu
   - enter `b all def-TZVP` here to assign the default triple zeta valence basis set
   - then end this menu with `*` and call `eht` in the next menu. Now hit <Enter> many times to accept all defaults....
   - skip the final menu and end define with `*` or `q`.

   Save the start Mos (`cp mos mos.save`) in an extra file and run **dscf**.

2. After the first run, rename the file *statistics* and copy back the start Mos (`cp mos.save mos`).

   Run **define** to change to a semi-direct run. In order to accomplish that, choose the `scf` option in the last **define** menu and select the `ints` sub option.
   Enter maximum size (e.g. 2000 MB) and name of the two-electron integral file (e.g. *twoint*).
   Please note that entering multiple integral files is neither necessary - since there is no file size limit on all machines TURBOMOLE runs on – nor possible with define.

   Make sure, that the scratch directory is located on a fast (!) access file system. You can give the whole path if the two-electron integral file should be in a directory other than the control file
   (e.g. */scratchdir/myusername/twoint*, provided that the directory */scratchdir/myusername/* exists).

   Start another **dscf** run and compare the timings (have a look at the two *statistics* files, as well).

Notes and hints:

1. The selection of integrals to be stored on disk is controlled by the two keywords `$thime` (threshold for estimated time consumption of integral batch evaluation; idea: we only need to store expensive integrals) and `$thize` (threshold for integral value; idea: only large integrals will contribute in subsequent iterations). The default values of the integrals work well, normally. If the given size of the two-electron integral file becomes too small you get error messages like

   ```
   <put> : FILE SPACE EXHAUSTED --> switching to 'direct' mode !
   written (=   2560) + newbuffer  >  max (=   2560)
   ```

   but the program will continue, nevertheless. Either increase the disk space or use larger values for `$thime` and `$thize`. For highly contracted basis sets like the Dunning series, `$thime 20` will still give some savings for larger cases.

2. Specifying an integral file will also improve the Z-vector equations in **rimp2** and **ricc2** runs.

3. NMR chemical shielding calculations using hybrid functionals like B3-LYP do also require a non-zero value for the integral file size.

It is also possible to let **dscf** calculate the size of the integral file for the given settings of $thime and $thize:

1. Prepare your input as usual with **define**.

2. On the command line, call

```
stati dscf
```

   `stati` is a TURBOMOLE own script that will do nothing but adding the keyword `$statistics <option>` to the control file.

3. Run **dscf** as usual. It will perform a statistics run only and change the value of `size` of the keyword `$scfintunit` accordingly.

4. Now you can edit the *control* file and check if the size is reasonable. If it is too large for your system, change it to an arbitrary value – as mentioned above, **dscf** will fill the file and then continue the calculation in the direct mode.

6.3. Check for convergence

After a single point energy calculation has been done, you can check if and how the energy has converged:

1. The converged energy is written to the file *energy* – however, if there has been such a file before, the energy is appended to the existing ones. You might have to check the date and time of the file to find out if it has been written lately.

2. The header of the MOs (either file *mos* or *alpha* for RHF or UHF cases resp.) gives a hint about the status of the MOs. Do a `head mos` or `head alpha` and check if you can find one of the entries in the first line:

   - `scfconv=6` # converged MOs with energy convergence of 1d-6
   - `expanded` # start MOs from an extended Hückel guess
   - `scfdump=23` # non-converged MOs after 23 SCF iterations

3. If you have **gnuplot** installed, you can call `cgnce <output file>` to get a graphical plot of the convergence behaviour, with the output file of the energy run as option. If no output file is given, `cgnce` takes *job.last*, i.e. the last energy run of a geometry calculation (see chapter 7.1).

6.4. RI-MP2 is much faster than MP2

Create two new subdirectories and copy to each the coordinates of C_8H_8 (from exercise 5.3). If you do not have the coordinates, use the ones from the structure library (see below).

Make sure that the working directory, from which the runs are started later on, is located on a file system with fast access (i.e. directly connected to the node you are running on and shared by as few processors as possible).

- In both directories, run **`define`** and set up the input for (semi-direct, if you like) SCF calculations with a def-SVP (in one directory) and a def-TZVPP basis set (in the other directory), respectively. Don't forget to set the symmetry-information (`desy`)! To add the coordinates, use either `a coord` if you have copied the file, or `a !c8h8` if you do not have the coordinates (any more?).

  Enter `mp2` at the general menu and then assign the auxiliary basis sets with `cbas`. You should get:

  ```
  AUXILIARY BASIS SET DEFINITION MENU
  ( #atoms=16    #cbas=16    )
  ```

  And then `other` and `emp2` to do a MP2 energy only calculation.

- accept all defaults but change the core memory to some larger value (should be 70 to 80 % of the RAM per CPU, some memory should be left to the system for I/O cache). **`define`** will automatically set the keyword
  ```
  $denconv .1d-6
  ```
  thus requesting a tighter convergence threshold. All post-SCF programs will terminate, if this keyword is not found in *`control`*

- Run **`dscf`**.

- Prepare also the conventional MP2 run. Issue the command

  ```
  mp2prep -e -p 2000 -m 700
  ```

  to prepare a run with a maximum disk usage of 2000 MB and a maximum memory usage of 700 MB (adapt these values as necessary). The script **`mp2prep`** works only properly, if the file *`mos`* has been created by a converged SCF run (this is the reason, why we ran **`dscf`** before).

- Now, we can start the MP2 calculations (**`rimp2/ricc2`** and **`mpgrad`**)
  Note that those two programs do not disturb each other. There is no need to run them in different directories. However, save the output from both runs, because the energy is not appended to the energy file (no geometry optimization, so both energies are written to the 1st cycle and hence, the second program overwrites the energies of the first one)

Calculate the RI-error in the correlation energy in percent:

$$\Delta RI = (E_{corr}^{RIMP2} - E_{corr}^{MP2}) / E_{corr}^{MP2} * 100$$

and compare to the change in the correlation energy between def-SVP and def-TZVPP.

6.5. RI-SCF works best for large basis sets

Make three subdirectories. Call **define** in each directory, and load the coordinates of BF from the structure library (`a !bf`). Prepare semi-direct (!) **dscf** runs, employing the cc-pVTZ, cc-pVQZ and cc-pV5Z basis sets respectively (see chapter 6.2 to see how semi-direct runs are prepared) . Keep copies of the initial *mos* files and run the calculations. Then, prepare the RI-SCF calculations. To do that, just run through **define** until you reach the last menu. Enter the `rijk` sub menu and switch on this option. This will automatically load the appropriate auxiliary basis sets. Choose a sufficiently large value for the core memory. Compare the computation times.

6.6. Don't get confused by auxiliary basis sets

To speed up calculations using the Resolution-of-the-identity (RI) approximation, TURBOMOLE currently provides three kinds of auxiliary basis sets, which you should not mix up.

1. Coulomb-Fitting (`$jbas`). Basis sets declared in this data-group are needed for RI-DFT calculations, where only the Coulomb part of the Fock operator is approximated. They can be kept rather small.
2. Coulomb- and Exchange-Fitting (`$jkbas`). If additionally the exchange part of the Fock operator needs to be approximated (RI-SCF calculations), a larger fitting basis is needed. It also can be used for RI-DFT runs (if the accuracy of the `$jbas` auxiliary basis is not satisfactory for you).
3. Correlation-Fitting (`$cbas`). These basis sets are needed for RI-MP2 and RI-CC2 calculations. They contain basis functions with high angular momentum.

Have a look at the three different kinds of fitting basis sets by doing the following:

- Run **define** and input just one atom (e.g. carbon).
- Choose the def-TZVPP basis set and provide dummy start orbitals.
- In the last menu, enter the `ri` section and switch RI `on`. This will automatically make the program write the `$jbas` data group. Then enter the `rijk` menu and leave it again. This causes the `$jkbas` data group to be written.
- In the last menu, go to the `mp2` menu to get the `$cbas` data group by entering `cbas`.
- Have a look at the *auxbasis* file. It contains the three data groups `$jbas`, `$jkbas` and `$cbas`. Try to find out the differences.

6.7. DFT versus RI-DFT versus MARI-DFT

For large RI-DFT calculations (more than 2000 basis functions), the multipole-accelerated RI-J approximation (MARI-J) leads to a considerable speed-up. The loss in numerical precision is much lower than that due to the RI-approximation. The multipole acceleration produces almost no overhead, so it can be used for any large DFT calculation.

In `$TURBODIR/structures/c216h36/` you can find the coordinates of a graphite sheet ($C_{216}H_{32}$). For a system of this size, the multipole approximation is faster by nearly a factor of 2.

Setting up the input is rather easy. Prepare a RI-DFT run as usual. In the final menu of **define** - additionally to the sub menus `ri` and `dft` - you also have to visit `marij`. Just switch on this option and accept the defaults. Only if your calculation contains basis functions with higher angular momenta (f-functions), you will have to use higher maximum multipole moments.

For comparison, you may start a conventional DFT run. It will take ages!

6.8. Finding occupation numbers

Dealing with transition metal clusters one is often faced with the problem that the extended Hückel (EHT) guess does not provide the correct occupation. Playing around with occupation numbers often makes the user feel like Sisyphus. Fractional occupation numbers (FON) which are automatically determined in a Fermi-distribution motivated manner (actually an error function is used) can improve convergence in these cases (see manual, keyword `$fermi`).

- Create a new directory and run **`define`**, read in the coordinates of Pd_4 from the structure library (`a !pd4`) and detect the symmetry (should be D_{2d}). As basis set accept the default def-SV(P) basis.

- In the "OCCUPATION NUMBER & MOLECULAR ORBITAL DEFINITION MENU" choose
  `eht`
  and answer `no` to the next question. This chooses EHT parameters for d^9s^1 occupation instead of d^{10}.

  Answer with `no` the question: 'DO YOU ACCEPT THIS OCCUPATION ?'.
  This provides you a menu for assignment of occupation numbers. Choose the quintet state by entering
  `u 4`
  (alternatively you may choose a singlet by entering `s` or `u 0`).

- In the final **`define`** menu choose a RI-DFT calculation using the BP86 functional.

- Enter the menu of "NON-DEFAULT SCF PARAMETER" and select
  `fermi`
  Set the final temperature to 25 K, the annealing factor to 0.90, and the HOMO-LUMO gap criterion to 1.0.

- In the same "NON-DEFAULT SCF PARAMETER" menu select
  `damp`
  Set the start value for damping to 3.0 and the final value to 1.0.

- Finally, open *`control`* file and increase value of `$scfiterlimit` to 50.

- Run the **`ridft`** calculation.

Save the output and the energy file, then try to do a few calculations with different occupations and start orbitals, but without using `$fermi`. Check how close you can get in the total energy compared to the previous run.

Warning: Fermi pseudo smearing is a very nice feature to get an occupation you would not have expected to be the lowest one... But if you leave `$fermi` in the control file and then start a geometry optimization, **`ridft`** or **`dscf`** will redo the occupation in each geometry cycle! That might be what you want, but the run has the possibility to switch from one hyper surface to another, leading to a completely different minimum structure with a completely different occupation than you might have expected. This minimum does not have to be lower in energy than the one with the fixed occupation numbers...

6.9. SCF convergence

If your energy did not converge after a first run, you can change all the keywords that are responsible for the convergence behavior in the control file:

1. `$scfiterlimit` # sets the maximum number of SCF iterations
2. `$scfdamp` # sets the damping of the DIIS, damping `starts` with the start value
   # and will either be decreased or increased by `step` and decreased
   # only to the limit of `min`.
3. `$fermi` # use it to let **`dscf`** or **`ridft`** determine the occupation automatically.
   # Check at the end of the run if your occupation numbers are all close
   # to an integer value. Values between 0.2 and 0.8 indicate either a not
   # fully converged SCF step or possibly a multi reference case (which can
   # not be treated in TURBOMOLE)
4. `$scforbitalshift` # helps to get a better convergence by shifting the virtual and/or occupied
   # orbitals up or down, resp. - please refer to the manual (keywords section)

6.10. Counterpoise calculations

The basis set superposition error (BSSE) can be corrected for using the counterpoise method. The following example explains how to perform a counterpoise calculation with TURBOMOLE.

> Note that this example is a simple BSSE case for a dimer. More complicated BSSE calculations for up to three fragments and counterpoise corrected geometry optimizations can be done with the module **`jobbsse`**. Please read chapter 3.6 of the manual if you want to use that tool.

The coordinates of the dimer of H_2O in C_s symmetry are available from the structure library. The structure has already been optimized.

First of all the energy of the dimer has to be evaluated at the MP2-level:

1. Crate a new working directory and run **`define`** as for a SCF run. Load the coordinates of the water dimer from the structure library (`a !h2o_2`). Choose the basis sets def-SV(P) and create start-MOs for **`dscf`** from Hückel guess.
2. Go to the `mp2` section in the last menu
   1. Let **`define`** choose an appropriate auxiliary basis set by entering `cbas`.
   2. Enter
      ```
      freeze
      ```
      to get in the frozen core assignment menu and choose
      ```
      core 0
      ```
      in order not to have frozen electrons.
3. Run a **`dscf`**-run first and a **`rimp2`**-run afterwards to get the energy of the dimer at the MP2 level (E_{DIM}).

Next, you can evaluate the energy of each water molecule, in the presence of the basis set of the second one:

1. Create a new directory and copy in it the *control* and the *coord* files.

2. In the control file change the symmetry group of the system to C_1 and the $atoms keyword, so that one of the two water molecules displays charge zero:

```
$atoms
 o 1  \
 cbas=o def-SV(P)          \
 basis =o def-SV(P)
 h 2-3                     \
 cbas=h def-SV(P)          \
 basis =h def-SV(P)
 o 4  \
 charge =0.000000          \
 cbas=o def-SV(P)          \
 basis =o def-SV(P)
 h 5-6                     \
 charge =0.000000          \
 cbas=h def-SV(P)          \
 basis =h def-SV(P)
```

3. Alternatively (and that is the easier way to do it), start **define** with the input and change the charge of the elements in question to zero by entering in the Atomic Attributes Menu:

```
c 4-6 0.000
```

4. Run **define** to get start-MOs and to switch off the frozen core approximation as shown above.

5. Run a **dscf**-run first and a **rimp2**-run afterwards, to get the energy of the monomer, already corrected for the BSSE (E_{1B}).

6. Repeat the same procedure for the second molecule of water, to get the energy of the monomer E_{2B}.

You can evaluate the dissociation energy, corrected for the BSSE, in the following way:

$$\Delta E_B = E_{DIM} - E_{1B} - E_{2B}$$

Furthermore, you can calculate the energies of the molecules of water in the geometry of the complex:

1. Create a new directory and copy in it the *coord* file.

2. Delete in the *coord* file the coordinates of one of the two water molecules.

3. Run **define** as usual.

4. Run a **dscf** and a **rimp2** step, to get the energy of the monomer E_1.

5. Repeat the same procedure for the other water molecule, to get the energy of the monomer E_2.

Now you can estimate the dissociation energy (without structure relaxation effect) in the following way:

$$\Delta E = E_{DIM} - E_1 - E_2$$

The BSSE is usually defined as the difference between the two dissociation energies:

$$BSSE = \Delta E - \Delta E_B$$

6.11. Parallel runs

The procedure to set up and run parallel jobs depends much on the computer architecture and the queuing-system.

There are four different parallelizations implemented in the current version of Turbomole, but for their usage no additional keywords or settings are needed. Also the user does just have to tell Turbomole whether a parallel job will be running on a single system with many CPUs or cores (SMP) or on a cluster using several (different) nodes conntected by a network interface:

1. MPI – parallel versions of dscf, grad, ridft, rdgrad, mpgrad, ricc2
   for ridft and rdgrad hybrid functionals and RI-K are not supported in the MPI version

2. SMP – parallel versions for shared memory systems of dscf, ricc2, aoforce, escf, and egrad are available, and a 'GlobalArray' version of ridft and rdgrad using MPI and shared memory on one node or distributed memory on several nodes is available. This is an independent parallelization and includes support for hybrid functionals and RI-K calculations. Technically more demanding than the MPI version, but usually more efficient.

In general, the parallel version can be used the same way the serial one is used. All you have to do is to set the environment variable PARA_ARCH to MPI for the MPI parallelization or to SMP for the multi-core version, and, if you do not want to do the calculations on the default 2 nodes, set PARNODES to the number of CPUs you would like to use.
For the SMP version, which is the recommended way to run post SCF calculations on a multi-core system, the binaries will be called by wrapper scripts named like the serial binaries. Hence, calling this version does not differ from calling the serial binaries.

MPI Version for parallel jobs over a network using different nodes:

- set `PARA_ARCH=MPI`
  the sysname script will append a _mpi to its output:

  ```
  export PARA_ARCH=MPI
  ```

- set `PARNODES` to the number of CPUs you like to use:

  ```
  export PARNODES=4
  ```

- to run parallel single-point jobs, it is recommended to reset the PATH:

  ```
  export PATH=$TURBODIR/bin/`sysname`/:$PATH
  ```

  since now sysname prints out a different name.

- The name of the nodes where the parallel job will run is also needed. If you use a supported queuing system (PBS, LSF, SGE), Turbomole will automatically find the right nodes from the provided settings.

  Otherwise write the names of the machines in a file (each line just one name, so for using several CPUs or cores on one node, add additional lines with the same name) and set the environment variable `HOSTS_FILE` to this file. Turbomole will process this list and use either just as many nodes as provided or just the number which has been set with `PARNODES` – depending on which number is larger.

SMP versions of dscf, ricc2, aoforce, escf, egrad, ridft, rdgrad, grad:

- set `PARA_ARCH=SMP`
  the sysname script will append a _smp to its output.
- set number of threads with `PARNODES` and the `PATH` just in the same way as for the MPI version.

- most SMP binaries utilize shared memory on the system, so memory limits (for shared memory but not just for them) are very important to be large enough. In addition to the usual stack size limit problem, make sure that your maximum shared memory you are allowed to use on your system is large enough (a few GB or 70-90 percent of the total memory should be sufficient):
  - `cat /proc/sys/kernel/shmmax` shows the amount of allowed shared memory
  - use `sysctl` to set new values, but you have to be root to do that!
- for ridft and rdgrad: The default shared memory that is used is per process for the matrices is 300 MB for heap and 10 MB for stack. For large cases this can be too small and an error will be given in the output. In order to increase the default values, just set:
  - `$paroptions`
    `ga_memperproc` *`stacksize heapsize`*
    heapsize and stacksize have to be given in double words (which is the same as the amount in Bytes divided by 8).

The most common way to run jobs is to have a queuing system like PBS, LSF or SGE. For those queuing systems, TURBOMOLE will use the information about the available nodes directly, so it will automatically start the jobs where they are supposed to run. For other queuing systems: please ask your system administrator.

In general, if no PBS/LSF/SGE is used, TURBOMOLE will check if it finds an environment variable HOSTS_FILE containing the name of a file with a list of available machine names. Otherwise the node where the job is started will be used.

The input has to be in a directory that can be accessed on each node, having the same path name. Usually one starts parallel calculations from an NFS disk.

Note: avoid the usage of the shared NFS disk as much as possible !! Otherwise this might dramatically slow down the calculation. See the hints below and carefully read Hints in the Turbomole User Forum.

Some hints:

1. On clusters of SMP nodes, the parallel programs will run more efficiently if the jobs are distributed over the nodes, rather than running on the same node. Communication is usually not so much an issue, but reading/writing to the memory and disk is faster for clients that do not have to share the bandwidth with other clients.

2. When using **`ridft`** for larger inputs, do not forget to switch on `$marij`. Multipole accelerated RI-J can be more than a factor of 6 faster than usual RI-J calculations for big systems. You get that for free and without additional errors, **so it is always safe to turn it on**.

3. **NumForce** can also be run in parallel, but here it is much more efficient to run the serial binaries at a time instead of the parallel version – all single-point jobs can be run independently.
   To run it in parallel, add the -mfile option with the name of a file which contains a list of nodes (just like $HOSTS_FILE):

   ```
   NumForce -mfile $HOSTS_FILE [...]
   ```

6.12. CCSD(T) with OpenMP

Let us now try a single point energy calculation of Water on CCSD(T)/cc-pVTZ level.

1. start in an empty directory

2. assuming TURBODIR is already set, now we set the path to the SMP version.
```
export PARA_ARCH=SMP
export PATH=$TURBODIR/bin/`sysname`:$PATH
```

3. call **define** and type in the following:
```
<Enter>
This is an example of a CCSD(T) calculation
a ! h2o
y
desy
ired
*
bb all cc-pVTZ
*
eht
<Enter>
<Enter>
<Enter>
cc
freeze
*
cbas
*
ricc2
ccsd(t)
*
*
*
```

4. run a Hartree-Fock energy calculation:
```
dscf > dscf.out
```

5. run the Coupled-Cluster energy calculation:
```
ccsdf12 > ccsdt.out
```

6. look at the output.
At the beginning of dscf.out and ccsdf12 you will the line: 'OpenMP run-time library returned nthreads = 2'.
At the end of ccsdt.out the final energies of MP2, SCS-MP2, SOS-MP2, CCSD, and CCSD(T) are summarized.

7. Structure Optimizations

7.1. Getting in contact with jobex

jobex is a shell script that drives the geometry optimization. Here, we will have a look at a HF/SV(P) geometry optimization of Cp^-.

- Create a new input for this molecule, just the way it has been done in chapter 4 or 6.1. Note that the input itself does not contain the information about the kind of job you want to run: Single point or structure optimization.
- In order to see how the optimization is working, we first proceed step by step. First, we carry out the energy calculation (run **dscf**). This will create a file called *energy* (and converged MOs in *mos*, of course).

  After that, we shall run the gradient program. Run the module **grad**. It will produce the file *gradient*.

  Finally, the module **statpt** will estimate new coordinates. You can monitor the current structure with the help of the command `dist`. Try this command before and after the **statpt** run.
- **jobex** will carry out these three steps automatically. On the command line, type

  ```
  jobex -h
  ```

  to get some further information. Then just run **jobex** without any further command line arguments. After completion, inspect the *energy* and *gradient* files. A useful command for the latter is

  ```
  grep cyc gradient
  ```

  Note, that there is one more energy in the *energy* file. The reason is that for a SCF (and DFT) optimization, **jobex** tests the convergence after each energy calculation.
- A RI-DFT optimization would need:

  ```
  jobex -ri
  ```
- In another directory, set up a RI-MP2 calculation and start the optimization by typing

  ```
  jobex -ri -level cc2
  ```

7.2. Did it converge?

Jobex finishes after the calculation did converge, but it also stops if the maximum number of cycles (default is 20) has been reached. Now, how do we get more informations about the job that has just finished?

1. Make two new directories.
2. Call **define** there and load acrolein from the structure library:

- hit <Enter> and then give a title
- enter `a !acrolein`
- then `desy` and `ired`, as usual

3. Prepare a RI-DFT calculation, accepting all defaults (basis set, functional, occupation, etc.). Hence, you can just say <Enter> and * all the time – just remember set `dft` to `on` as well as setting `ri` to `on` in the last, the general menu.

4. Copy the input to the other directory.

5. In one directory call

```
jobex -ri -c 5
```

6. And in the other directory

```
jobex -ri -c 100
```

7. Compare the two directories.

8. To check what jobex was about to do, look at the file *job.start*. There you will find the information about the options that have been given, the method that has been chosen and the convergence criteria that have been applied.

9. You will find a file called *converged* in the second directory and a file *not.converged* in the first directory.

10. Open *not.converged* in an editor. You can see the criterias given from jobex in `$convcrit` and the current criteria in `$convinfo`.

11. Do a `grep cyc gradient` in both directories.

12. Look at the end of the files *job.last*

7.3. Preoptimization

1. P_7H_3 is a rather nice cage molecule (its point group we naively assume to be C_{3v}, in exercise 8 we will examine this), set up the input for a RIDFT structure optimization (def-SV(P) basis and B-P functional).
Create a new working directory, call **`define`** and load P_7H_3 from the structure library:

- hit <Enter> and then give a title
- enter
  `a !p7h3`
- then `desy` and `ired`, as usual.
- enter two * until you reach the Molecular Orbital menu and generate start MOs with
  `eht`
  accept all defaults by hitting <Enter> many times...
- now you should be in the General Menu
- switch `on dft` and `ri`
- end **`define`**

Start `jobex -ri`
Create another subdirectory and start again by reading in the coordinates of P_7H_3. After symmetrization, try the

```
ff
```

command to pre-optimize the structure using the force field program (intuitive usage). Run a RIDFT geometry optimization starting from this pre-optimized structure. Document the number of optimization cycles with and without pre-optimization.

2. Starting from the RIDFT/SV(P) structure, optimize the geometry at the RIDFT/TZVP level. Create two subdirectories.
    1. Copy the *coord* file into the first directory, set up the input and run **jobex**.
    2. For the second run which you prepare in the other directory, use as well the *coord* file as well as the *forceapprox* file from the SV(P) calculation (make sure, that a line saying `$forceinit off` is contained in the control file before starting **jobex**).

Compare the convergence of the two structure optimizations.

7.4. Constrained optimization

In the structure library, you will find the converged ground state structure of H_3CBr (RI-MP2/TZVPP). Assume, we are interested in the energy path upon approach of a chlorine anion.

- Load the coordinates into **define** (`a !h3cbr`), get the symmetry (via good old `desy`) and add a chlorine atom at an appropriate position (somewhere on the negative z-axis).
  Define a bond between chlorine and carbon by entering the command sequence

  ```
  idef
  f stre 1 6      # defines a [f]ixed bond between atom 1 c and 6 cl
  ```

  After several blank lines you are back in the geometry menu. Test the quality of the B-matrix with `imet`. Find out the number of that newly defined internal coordinate (enter `dis`). Then use the command

  ```
  imanat <# of int coord>
  ```

  to change this distance to 250 pm (actually, you have to enter `2.5 A` since only Ås are supported). Now we are ready to leave the geometry menu.

- Assign def-TZVPP basis sets to the atoms.
  In the molecular orbital menu, remember to set the total charge to -1.

- The input preparation is finished by running through the `mp2` options in the last **define** menu.
  We ignore the warning concerning the too small gap between frozen and active orbitals (it is due to the rather high-lying d-orbitals of bromine).

- Now you can start the RIMP2 optimization by

  ```
  jobex -ri -level cc2
  ```

- Inspect the results. If you liked it, you can decrease the carbon-chlorine distance (use `man` in **define**) and look for an estimate of the transition structure.

Before sending the results to JACS, please consider two things:

a) Freezing the d-orbitals of 3rd row elements may be problematic, larger basis set like def-QZVPP are adequate basis sets for including them.
b) We should have tested the effect of diffuse functions on chlorine and bromine (anions!).

For more complex structures, the automatic generation of internal coordinates might fail. In that case, just proceed as written above and define the fixed coordinates by using the command

```
idef
```

After that, go back to the geometry menu of **define** and do

```
ired
```

The automatic definition of redundant internal coordinates ignores the non-redundant coordinate definitions.
But: fixed internal coordinates will be taken into account, building redundant internal coordinates around the given fixed ones.

The more fixed internal coordinates you have, the more difficult it is to find the remaining set of redundant internal coordinates – and the higher the possibility that the set of redundant internal coordinates will get linear dependent during the optimization!

7.5. Molecular dynamics - simulated annealing

Simulated annealing serves for finding new structures on complex hyper surfaces, e.g. for metal clusters. Here, we chose a more obvious example: the $PtCl_4^{+2}$ complex.
The coordinates of that molecule, but with a distorted tetrahedral structure are available in the structure library.

1. Run **define** as for a RI-DFT run, but the group symmetry of the complex should remain C_1.
   Load $PtCl_4$ by entering `a !ptcl4`. Choose basis sets (def-SVP) and create start MOs (do not forget that the charge is +2).
   In the 'GENERAL MENU', choose `dft` (default functional and grid size are appropriate) and `ri` options.
   Adjust the option `ricore` according to the available RAM per processor.
   Set the maximum number of SCF iterations to 100 (`$scfiterlimit`) and the daming to a start value of 3 (`$scfdamp start=3.0 step=0.050 min=0.050`).

2. Run **mdprep**. Most default parameters are appropriate, change only the following ones:
   1. request 150 MD steps in the section 'NUMBER OF MD STEPS';
   2. in the section 'USER-DEFINED ACTIONS' choose a simulated annealing calculation;
   3. in the sub-menu 'SIMULATED ANNEALING / QUENCHING' choose the anneal option;
   4. enter an annealing rate of 0.985 and a time of 1000, from which the anneal will start to occur.

3. The program **mdprep** creates two files: *mdmaster* and *mdlog*. The first is a command file, which contains the keywords for the module **frog**, while the latter is an output file. Be curious and inspect these files.

In order to run the structure optimization, you have to use the command:

```
jobex -ri -md
```

The procedure consists of a **ridft** step, followed by a gradient and by a molecular dynamics step. In the file

gradient the total energy, the coordinates and the gradient for each interaction are collected. In the file *mdlog* information on the time and the displacements are dumped. The file *job.last* contains the output of the gradient, of the **ridft** and of the MD steps of the last complete iteration. There should be a script `log2egy` which accomplishes that.

In order to understand the output, you should `grep` from the *gradient* file the numbers of the cycles versus the total energy and perhaps plot a graphic. Then you can choose the coordinates of the geometry with the minimal energy and run a conventional optimization to refine the structure.

7.6. Transition states

Searching transition state is usually a highly non-trivial task, simply because there are many of them while you are searching for the one with the lowest barrier. Usually, you have already the reactant and product structures when looking for the transition state.

There are two different steps needed to get a transition state:

1. you first have to guess an approximate transition state somewhere between the reactant and product structures,
2. if you have a guess structure for the transition state (like the one from 1.), and if that one is within the quadratic region around the real state, you can apply a procedures like quasi–Newton–Josephson methods. They are based on the restricted second-order method, which employs Hessian shift parameters. The program **statpt** is doing that for you.

While step 2 is more or less straight forward and can be done in a similar way a geometry optimization (search for minima) is performed, step 1 requires a lot of work, chemical intuition, experience, and time – at least if you want to find *the* transition state rather than *a* transition state. Some remarks and details can be found in the TURBOMOLE documentation, chapter 3.2 and 3.2.4.

Let's start with step 2:

If you like to see how step 2 works (i.e. if you have a guess for a transition state structure), get the coordinates of acrolein from structure library, and execute the following steps:

- Run **define**, load the coordinates (`a !acrolein`) and leave the geometry menu without specifying internal coordinates (answer `no` after leaving the menu).
  As basis set specify `3-21g hondo`.
  Provide Hückel MOs and continue until the last menu where you enter the

  ```
  stp
  ```

  section and set the index of the transition vector (`itvc`) to `1`. Please note that in **statpt**, the six zero eigenvalues of rotation and translation are not between the negative and positive values as usual, but shifted away. So `itvec 4` is the 4th eigenvalue, counted from minus infinity upwards, leaving out the zeros.

- Run the **dscf**, **grad** and **aoforce** modules in order to provide the start information for the transition state search. Please see chapter 8 for a detailed description of the module **aoforce** and how to perform force constant calculations.

- Run the command

  ```
  jobex -trans
  ```

to start the transition state search.

- Run **aoforce** again to check, whether a transition state has been reached.

Here you did a full force constant calculation (see chapter 8) with the input you are using for finding the transition state. For real life applications, the input for this is usually a bit different:

1. The system in question is much bigger,
2. The basis set is a bigger one, usually at least def-SVP for (RI-)DFT and def-TZVP for MP2/CC2 calculations, to get reasonable results.
3. The symmetry is set to C_1 only, otherwise you are doing a constrained search.

For such realistic inputs, getting the full Hessian can be the by far most time consuming step of the calculation. Generally, the quality of the eigenvector for a transition state search does not have to be too high, using a lower and cheaper method might just take a few geometry steps more, but saving much time at the initial Hessian calculation.

Doing the force constant calculation at RI-DFT/B-P86/def-SV(P) level is usually good enough. In addition, if the system is big, a LES search can be done at this level (see chapter 8.2 about how to perform such a calculation).

Now to step 1:

There is no black box method in TURBOMOLE that finds a good guess structure for the transition state. However, some of the features can be of great help:

a) Constraint minimization by freezing internal, internal redundant, or Cartesian coordinates.

   Fixing internal coordinates is described in chapter 7.4. Freeze the internal coordinates which you think are involved in the transition and run a usual geometry optimization.

   Change the value of the internal coordinates with **define** by using `iman` in the internal coordinate menu, and redo this steps for a few structures.

   Use the structure with the highest total energy to get as close to the transition state as possible.

   Note about fixing Cartesian coordinates:

   > To fix a Cartesian coordinate, open the *coord* file and append an `f` to each line of the atoms that should be fixed. Please make sure that there is a blank between the element name and the `f`!

b) Scanning the potential energy surface (PES) along an internal (z-matrix) coordinate using the **tmole** script.

   This step is very similar to a), but here you do not have to distort anything yourself. **tmole** is able to read in z-matrix coordinates with parameters. The additional `%scan` option in the **tmole** input file determines along which coordinate is scanned. An example input can be found in the Manual of **tmole**.

   To run the calculation, simply call

   ```
   tmole tmole.in
   ```

   at the command line. **tmole** prints the results in a file called *potentialcurve.molden* which can be opened and viewed in **molden** or directly in an editor.

c) Distortion of the structure along an imaginary mode using the script **screwer**.

If your first crude guess gives more than one imaginary frequency, you can distort the structure along one of the (negative) modes. To do that, make sure that you have converged MOs from an energy run, the gradients from a gradient run and the Hessian and the vibrational modes from an **aoforce** run. Then, just call

```
screwer
```

and follow the instructions. You will be asked for the number of the mode along which the coordinates will be changed. Please look at the output (scroll up the window if necessary) and choose one of the imaginary frequencies. Next you will be asked for the step length, given as a temperature in Kelvin. If the value is too low, **statpt** might optimize the structure back to the initial guess – and if it is too high, you might get to the other side of the hill on the PES...

d) Most convenient is the usage of TmoleX for scan jobs. The builder within TmoleX can define bond lengths, angles and torsions as fixed, and this can be used to scan along one or along several of the internal coordinates. See the TmoleX manual for details.

A jobex of your own?

jobex, like many other TURBOMOLE tools, is a shell script. If you know something about shell programming, why not have a look at it? Just get a local copy and customize it.

8. Vibrational normal modes

8.1. aoforce: Analytical force constant calculations for HF and DFT (ground states)

The calculation of HF and DFT force constants is quite easy: After having performed a single point energy calculation or a geometry optimization, just start the program **aoforce**.

- Retrieve the geometry-optimized BP86/SV(P) calculation of P_7H_3 from exercise 7.3, or redo the input generation and the **jobex** calculation. Perform a force constant calculation with the program **aoforce**:

  ```
  aoforce > force.out
  ```

  Check the file *force.out* and the *control* file.

  While *force.out* contains the detailed data about the vibrational spectrum, the zero point energy, etc., the keyword `$vibrational spectrum` in the *control* file contains a list of the modes:

  ```
  $vibrational spectrum
  #  mode     symmetry     wave number   IR intensity    selection rules
  #                         cm**(-1)        km/mol          IR      RAMAN
       1        a2           -768.31         0.00000        NO       NO
       2        e            -667.21         0.00000        YES      YES
       3        e            -667.21         0.00000        YES      YES
       4                        0.00         0.00000         -        -
       5                        0.00         0.00000         -        -
       6                        0.00         0.00000         -        -
       7                        0.00         0.01127         -        -
       8                        0.00         0.00379         -        -
       9                        0.00         0.01826         -        -
      10        a2           208.36         0.00000        NO       NO
      11        e            247.39         0.52958        YES      YES
      12        e            247.39         0.52958        YES      YES
      13        e            268.18         0.01352        YES      YES
  ...
  ```

  There are three negative eigenvalues: The C_{3v}-structure is found to be a transition state!

- Run the module
  `vibration`
  to lower the symmetry of the molecule along the "most imaginary" normal mode with the number `1`. The default value for the displacement (i.e. "room temperature") should be fine.

  After **vibration** has finished, copy the coordinates from the *control* file, found in the keyword `$newcoord` to the coordinate file *coord* – replace the coordinates and rerun

  ```
  define
  ```

  Now, **define** will find that the coordinates do not fit to the former symmetry, that is still given in the *control* file:

COORDINATES REFERENCED BY CONTROL INPUT FILE control DO NOT COMPLY WITH SCHOENFLIES SYMBOL c3v .
THESE COORDINATES AND SCHOENFLIES SYMBOL ARE IGNORED !
HIT >return< TO CONTINUE WITH STANDARD GEOMETRY MENU

So you have to reload the coordinates simply by saying:

```
a coord
```

To check for the new symmetry with

```
desy
```

which will find C_3 symmetry.

```
ired
```

will redo the internal coordinates according to the new structure (if you forget that here, **jobex** will fail to run properly later on. If that happens, just recall **define**, say `y` when being asked for changes of the structure and call `ired` now – accept the rest of the input until you reach the end of **define**).

And, you will also have to redo the start mos with `eht`.

- Before we continue, save the energy file:
  ```
  cp energy energy.c3v
  ```

- Perform a geometry-optimization of the molecule in the new symmetry (C_3) also on the BP86/SV(P)-level.

  The energy lowering by going from C_{3v}- to C_3-symmetry should be about 341.8 kJ/mol (or 0.13018 Hartree):

```
tail  -n 2 energy.c3v energy

==> energy.c3v <==
     7 -2390.781033598      2383.424468007     -4774.205501605
$end

==> energy <==
    25 -2390.911215563      2383.657356328     -4774.568571890
$end
```

Please note:

1. Small force constant calculations, like this example, work well with the default core memory. In case of larger molecules, however, huge matrices have to be stored, and it is advantageous to process several of them at the same time. For this purpose, add

   ```
   $maxcor <N>
   ```

   to the *control* file, where `N` specifies the core memory given in MB - but remember that some additional quantities, e.g. the core memory for RI-matrices, cause further memory requirements. Depending on the total size of the available memory, approximately 50% of it might be a good choice.

2. **aoforce** will always compute the Hessian, and from it the vibrational modes, even if your gradients are not

zero! In such cases, you are not in a region where a quadratic description of the potential energy surface is sufficient. Furthermore, non-zero gradients contribute to a high extend to the Hessian. Be aware that the numbers you get from such a calculation do not have to be meaningful.

3. TmoleX can be used to visualize the vibrational modes, and also to distort a structure along a mode.

8.2. Lowest Eigenvalue Search

One of the most common application of the calculation of vibrational modes is a check for the minimum or transition state of an optimized structure. Hence, it is not necessary to compute the complete Hessian and vibrational spectrum, since already the lowest eigenvalue determines the kind of a stationary point. A negative eigenvalue, i.e. an imaginary frequency, indicates a transition state of (at least) first order, whereas a positive lowest eigenvalue is a prove for a (local) minimum structure.

For larger systems, a Hessian calculation can be more expensive than the complete geometry optimization. **aoforce** scales with N^3, where N is the number of basis functions.

The Lowest Eigenvalue Search (LES) approach calculates only the lowest N eigenvalues and eigenvectors numerically (Davidson iteration scheme). This reduces the scaling behavior to N^2 and gets thus more efficient compared to the full run, the bigger your input is.

The setup of an LES calculation is the same as for the full one, the keyword `$les` switches on the LES mode of **aoforce**.

1. Load the coordinates of the P_7H_3 exercise from chapter 7.3, do an SCF energy calculation and then add

   ```
   $les all 2
   ```

   to the control file.

2. Run **aoforce** as before and compare the frequencies with the full run.

Note:

- LES is not faster than the calculation of the full spectrum if:

  1. The system is small to medium sized (as in the example here),
  2. The number of eigenvalues that shall be computed is too large,
  3. The order of the symmetry group is very high (reducing the 'largeness' of your system, see 1.).

- Since LES will calculate the eigenvector, it can also be used to generate an input Hessian matrix for a transition state search (see chapter 7.6) using **statpt**:

  1. You have to add (by hand) two more keywords to the control file:

     ```
     $h0hessian
     $nomw
     ```

  2. **statpt** will use the full Hessian instead of the one from the LES search if both are found.

8.3. NumForce: RI-MP2 force constant calculation on methane

NumForce enables numerical force constant calculations for all levels of theory with a gradient implemented.

As an example, we will perform a numerical RI-MP2/TZVP force constant calculation for methane with the **NumForce** script.

- Prepare an RI-MP2 calculation (energy and gradient calculation of MP2).
- Run **dscf** and **ricc2**.
- To run **NumForce** just issue the command

  ```
  NumForce -level cc2 -central
  ```

  This chooses RI-MP2 (options `-ri` and `-level mp2`) and additionally the use of central differences (better numerical stability) is ensured by the option `-central`.

Please note:

The default increment for numerical differentiation is 0.02 au. To get more accurate results, you may have to lower this value, but only, if you increase the quality of the wave function employing stricter convergence criteria (i.e. modify `$scfconv` and `$denconv` in *control*).

NumForce actually is a shell script. If you know some shell script programming, you can have a look at it and maybe improve it for your own purposes (of course you should copy it first).

If you have forgotten the options of **NumForce**, just call `NumForce -help` to get a short overview.

8.4. Analysis of Normal Modes in Terms of Internal Coordinates

A frequently asked question is which internal coordinate (bond, angle, ...) is involved in which mode of the vibrational spectrum. This can be answered by an analysis of the normal modes in terms of internal (non-redundant !) coordinates.

Chapter 8.1 of the TURBOMOLE documentation gives a nice description about how to perform such a calculation.

8.5. Thermodynamic data from vibrational frequencies

From a full vibrational spectrum (not from a lowest eigenvalue calculation), it is possible to get various thermodynamic functions at various temperatures and pressures.

After an **aoforce** or **NumForce** run, just call the 'free enthalpy' program:

```
freeh
```

in the directory of your input. **freeh** is an interactive program:

1. After calling it, `freeh` will first print out the normal modes from the control file. From the coordinates and the point group it finds out the symmetry number σ needed in the quasi-classical rotational partition sum. Just

accept the proposed σ and go ahead.

2. Next, you will be asked for a scaling factor for the frequencies:

```
wave numbers calculated by scf theory tend to be
1-10% too large and may be corrected by scaling
-----------------------------------------------
default value of scaling factor (BP/SVP):   0.9914
enter new value for corr, if you want to change
-----------------------------------------------
```

This scaling factor should be chosen depending on the applied method. The proposed scaling factor is the one that is recommended for BP/SVP calculations.
Other usual scaling factors[1] are, for a small SVP or 6-31G(d) like basis set:

Method	Scaling Factor
HF	0.8953
DFT: B-P86	0.9914
DFT: B-LYP	0.9945
DFT: B3-LYP	0.9614
MP2	0.9434

3. Next, you will be asked for a range of temperatures and pressures:

```
enter the range of temperatures (K) and pressures (MPa)
 at which you want to calculate partition sums and free enthalpies :

 tstart=<real> tend=<real> numt=<integer> pstart=<real> pend=<real>
nump=<integer>

 default values are  :
 tstart=298.15 tend=298.15 numt=1 pstart=0.1 pend=0.1 nump=1

 or enter  q  or  *  to quit
```

You do not have to write the complete line every time. If you accept the default temperature and pressure (298.15 K and 0.1 MPa), just hit <Enter>. Otherwise, just change the values of the temperatures or pressures:

```
tend=350 numt=2 pend=0.5 nump=4
```

The input above will let the start temperature and the start pressure at the default values while changing the end temperature to 350K and the end pressure to 0.5 MPa. `numt` and `nump` are the number of temperatures and pressures between the start and end values that will be evaluated.

1 Taken from Scott, A. P., Radom, L., *Harmonic Vibrational Frequencies: An Evaluation of Hartree-Fock, Møller-Plesset, Quadratic Configuration Interaction, Density Functional Theory, and Semiempirical Scale Factors*, J. Phys. Chem. A, 1996, **100**, 16502
and
Wolfram Koch, Max. C. Holthausen, *A Chemists Guide to Density Functional Theory*, Wiley-VCH, 2001

4. **freeh** will print out a list of several properties for the given range of temperature and pressure. Here an example for benzene, RI-DFT calculation using def-SV(P) basis set and C1 symmetry for the input given above (`tstart` is default, `tend` is 350, number of temperatures `numt` is 2, pressure is from default 0.1 MPa to `pend` of 0.5 MPa, number of pressures `numt` is 4):

```
  zero point vibrational energy
  -----------------------------
  zpe=   260.9     kJ/mol

T         p           ln(qtrans)  ln(qrot) ln(qvib) chem.pot. Energy   entropy
(K)      (MPa)                                      (kJ/mol)  (kJ/mol) (kJ/mol/K)
298.15   0.1000000      17.14     11.41     0.83    188.07    273.65   0.29534
298.15   0.2333333      16.29     11.41     0.83    190.17    273.65   0.28830
298.15   0.3666667      15.84     11.41     0.83    191.29    273.65   0.28454
298.15   0.5000000      15.53     11.41     0.83    192.06    273.65   0.28196
350.00   0.1000000      17.54     11.65     1.23    172.36    278.22   0.31077
350.00   0.2333333      16.69     11.65     1.23    174.82    278.22   0.30373
350.00   0.3666667      16.24     11.65     1.23    176.14    278.22   0.29997
350.00   0.5000000      15.93     11.65     1.23    177.04    278.22   0.29739

 T         P               Cv                 Cp
 (K)      (MPa)         (kJ/mol-K)     (kJ/mol-K)
298.15   0.1000000      0.0801568      0.0884711
298.15   0.2333333      0.0801568      0.0884711
298.15   0.3666667      0.0801568      0.0884711
298.15   0.5000000      0.0801568      0.0884711
350.00   0.1000000      0.0959260      0.1042403
350.00   0.2333333      0.0959260      0.1042403
350.00   0.3666667      0.0959260      0.1042403
350.00   0.5000000      0.0959260      0.1042403
```

You might have to scroll up the windows to see the output since `freeh` prints out a detailed explanation of the used formulas and some hints.

9. Excited states

9.1. **escf**: UV/Vis and CD Spectrum of Pentahelicene

- Starting from the coordinates in the structure library (`a !pentahelicene`), set up the input for an RI-DFT/SV(P) run (and please switch on symmetry by calling `desy`).

- Before leaving **`define`**, enter the

  ```
  ex
  ```

  submenu of the "GENERAL MENU" . Here you can see the properties and methods **`escf`** and **`egrad`** are capable of. If you have switched `dft on`, then e.g. `rpas` will do a singlet excitation at TDDFT level, otherwise it would be a TDHF (RPA) calculation.

  Specify a TDDFT calculation of singlet excited states by typing

  ```
  rpas
  ```

  Go to the next question by entering

  ```
  *
  ```

  and enter

  ```
  a 3
  ```
  and
  ```
  b 3
  ```

  to calculate 3 states of each A and B symmetry (if there is only symmetry A, you have probably forgotten poor `desy`). You can leave the other options unchanged. For larger calculations you may consider to increase `rpacor` which determines the number of vectors to be treated simultaneously (saves much time!).

- Run **`ridft`** and – subsequently – **`escf`**. Have a look at the results.

Note:

1. The irrep given in `$soes` is the one for the excited state vector, i.e. the direct product of the irreps of the ground state and the excited state.
2. Add `$spectrum <unit>` and/or `$cdspectrum <unit>` to the *`control`* file, and you will get the files *`spectrum`* and *`cdspectrum`* with a table containing the excitation energies (in the given unit) and the oscillator and/or rotatory strengths. Possible units are `eV`, `nm`, `au`, or `1/cm`.
3. Do not forget that you have to apply the selection rules yourself. The calculated oscillator strengths for the triplet excitations from a singlet ground state are of equal size than the ones for the singlet excitations. Keep in mind which excitations are suppressed because of the selection rules (spin).

After a calculation, the converged excitation vectors are saved in files (*`sing_*, trip_*`*). A following run with a larger number of excitations will read in and use the already converged eigenvectors.

9.2. *egrad: Excited state structure of CH_2O*

To optimize the structure of an excited state with TDDFT, one has to specify which excited state should be followed. Load the coordinates of CH2O from the structure library.

- Let the symmetry at C_1, since we do not know right now how the excited state will look like,
- Start from a new or empty directory and call **define**

  - hit <Enter> and then give a title
  - enter
    a !ch2o
  - and accept the found structure with y – then do ired
  - enter two * until you reach the Molecular Orbital menu and generate start MOs with
    eht
    accept all defaults by hitting <Enter> many times...
  - now you should be in the General Menu
  - switch on dft and ri
- Go to the ex menu and choose rpas for singlet.
- For gradient calculations, only one irrep is allowed – here it is much easier to edit the *control* file directly, rather than using **define**. Look at $soes if only one irrep is given as option.
- If no additional option is given, the highest number of the entry in $soes will be taken to compute the excited state gradient and structure.
  Sometimes, when two excitations are lying close in energy, the algorithm is more stable if you include some higher excitation vectors in addition. In that case, if the highest number in $soes is not the one you want to use, add the keyword
  $exopt <N>
  with <N> as the number of the excited state you have in mind.
- The module **egrad** does the energy and the gradient calculation. For single point runs, just start **egrad** instead of **escf** and look at the output.
- To optimize the excited state structure, run

```
jobex -ri -ex       # and other options like -c 100 if needed
```

- Look at the excited state structure. Call

```
t2x > exstate.xyz
```

  and load the file *exstate.xyz* in a viewer that is able to plot multiple xyz structures (**molden**, **jmol**, ...).

9.3. TDDFT excited state vibrational spectrum

Since the module **egrad** of TURBOMOLE calculates the gradients of the excited states analytically, second derivatives can be done numerically by using the script **NumForce** (see chapter 8.3).

We will now calculate the vibrational spectrum of the first excited (singlet) state of CH_2O at TDDFT level.

1. Use the directory of the last chapter with the converged excited state structure of CH_2O.

2. Since we have already specified which excited state we want to use for the gradient/structure calculation, we do not have to change the input file.

3. Just start **NumForce** with the option `-ex <number of state>` in addition to the usual options:

   ```
   NumForce -ex 1 -ri
   ```

   Note that you have to tell **NumForce** here which excited state in C_1 you want to follow. **NumForce** has to distort the geometry to do the numerical second derivatives for each of its steps. Hence, the symmetry for the calculation will automatically be lowered to C_1. The `$soes` keyword contains an `irrep` different than `a`, so we have to choose the right excitation in C_1 ourselves (by looking at the output of **jobex**, counting all excitations).

Got a transition state? Then check why:

- DFT grid? Set gridsize to 4 and redo the calculation from scratch (!).
- Basis set too small? Redo the calculation with def-TZVP.
- Convergence criteria? Call jobex with -energy 6 -gcart 4 in addition and start again from scratch (default is -energy 6 -gcart 3).

*9.4. **ricc2**: Excited states of HCP*

1. Start to make a usual input with **define** using the coordinates of HCP from the structure library (`a !hcp`). Choose cc2 in the General Menu and there:

   ```
   ricc2
   cc2
   *

   cbas
   *

   exci
   list irrep
   ```

   `list irrep` gives you the syntax of the `irrep` command in this submenu (unlike most other submenus, you have to specifiy a line just the way you would add the keyword to the control file).
   It also prints out the irreps of the chosen point group and possible multiplicities.

   ```
   irrep=a1  nexc=1
   irrep=a2  nexc=3
   *
   *
   *
   ```

2. After that, you can run the **ricc2** module. Note that this module currently knows Abelian point groups only if used for excited state calculations, therefore the symmetry group given in the *control* file is C_{2v}. Look

carefully at the output (D1-diagnostic, $\%T_1$ of the excitations).
Note that CC2 excitation energies are optimized in three steps.

3. You can also specify more than one method. Try:

```
$ricc2
  ccs
  cis(d)
  cc2
$excitations
  irrep=a2  nexc=1
```

Important General Note about restarts of ricc2:

Remove the *restart.cc*, *alles.cc*, *syminfo* files and all CC* files before rerunning ricc2 if you have changed the input - the program would try a CC2 restart and thus skip methods, fails to converge, stops with error messages or produces nonsense results!

4. You certainly are also interested in properties of the excited states. Try e.g.

```
$ricc2
 cc2
$excitations
 irrep=a2  nexc=1
 expval operators=diplen,qudlen
$response
 expval operators=diplen,qudlen
```

The $response keyword switches on the calculation of ground state properties. Inspect the output file carefully.

9.5. Circular dichroism and UV/Vis spectra calculations at CC2 level

If you want to calculate optical spectra, the oscillator and rotator strength have to be requested. An input looks like this:

```
$ricc2
  cc2
$excitations
 irrep=a nexc=2
 spectrum operators=diplen,dipvel,angmom
$spectrum ev
$cdspectrum ev
```

When given the $spectrum and the $cdspectrum flags the output is parsed in individual files, which can directly be loaded in plot programs, like e.g. gnuplot.

9.6. Excited state frequency analysis at CC2 level

NumForce (see chapter 8.3) can also be used to calculate the vibrational frequencies of excited states. We will try that for Ammonia and RI-CC2:

1. Prepare the input and load the coordinates from the structure library (`a !ammonia`).

2. Do not apply symmetry, so this time try to avoid entering `desy` and keep C_1 symmetry.

3. Accept the default def-SV(P) basis (yes, it is much too small, so please do not publish the results).

4. Go to `cc2` in the last menu and do:
   1. `cbas` # assign auxiliary basis sets, accept the ones **`define`** chooses for you
   2. `ricc2` and then `cc2` # switch on CC2 as method
   3. `geoopt cc2 (a 1)` # to get the vibrational spectrum of the first excited state, we need gradients
   4. `*`
   5. `exci` # go the the excitation menu
   6. `irrep=a nexc=1` # switch on the excited state calculation, otherwise the `(a 1)` # option of `geoopt` will not work!
   7. `*` and `*` and `*`

5. End define and run a geometry optimization at CC2 level:
   ```
   jobex -level cc2
   ```

6. Since we have chosen the gradients and the energy of the excited states, **`jobex`** will do a geometry optimization of the first excited state. If you delete the keyword `$excitations`, you will get the ground state.

7. Run
   ```
   NumForce -level cc2 > force.out
   ```

8. Take a look at *`force.out`* and *`numforce/aoforce.out`*.

Note: do not call `NumForce` with the `-ex` option! This does only work for TDDFT excited state calculations, whereas the CC2 part always is completely controlled by the keywords `$ricc2` and `$excitations`.

10. COSMO: dealing with solvation effects

Solvation effects can be treated by the COSMO (COnductorlike Screening MOdel). If you want to know more about COSMO, please read chapter 11 of the TURBOMOLE documentation or visit www.cosmologic.de

Briefly: COSMO is a continuum solvation model, almost identical to the CPCM called method in many other quantum chemistry programs. COSMO builds a metal cavity around the molecule, corresponding to an electrostatically ideal solvent of $\varepsilon=\infty$.

The cavity construction starts with a union of spheres of radii R_i + RSOLV for all atoms i. So the most important input parameter for COSMO is the radius for each element that should be taken for the cavity construction.

This is done interactively by the script

```
cosmoprep
```

that can be called in a directory containing a usual gas phase input.

Here we just do a geometry optimization with DFT (Hartree-Fock can be used the same way). To learn how to do vibrational frequency calculations or how to use COSMO with MP2, please see chapter 12.2.6 of the TURBOMOLE documentation.

Now, let us calculate the structure of benzene in a solute:

1. Start with define, load benzene from the structure library with

   ```
   a !benzene
   ```

2. Let the symmetry at C_1 and define internal redundant coordinates (`ired` and `*`).

3. Choose def-TZVP basis set (`b all def-TZVP` and `*`)

4. Do a `eht` and accept the occupation for charge=0 (`eht` and many times <Enter>).

5. Switch on `dft` an `ri`

6. Get **`define`** to an end with `q` or `*` and start

   ```
   cosmoprep
   ```

7. Accept all defaults until you get to the

   ```
   ******************************************************
   ***            radius definition menu              ***
   ******************************************************

   ( #atoms=   12 --- #radius=    0 )
   ```

8. Here you have to tell cosmoprep which radius for each element you want to use. Unless you have your own radii, there are two possibilities:

   1. optimized radius – for a limited set of elements, optimized radii have been determined by fitting the

results of many calculations to experiments,

2. bondii radius – unoptimized radii for all elements

A good approach is to use optimized radii for all elements available, and the bondii for the rest.
Enter

```
r all o
```

to assign optimized radii to all atoms.
In case of benzene, all atoms have got a radius now:

```
( #atoms=   12 --- #radius=   12 )
```

9. For elements where optimized radii are missing, e.g. Si in $SiMe_3$, you would get something like:

```
( #atoms=   13 --- #radius=   12 )
```

so for one of your 25 atoms the optimized radius is missing and you have to assign bondii radii.
Scroll up your windows and search for one or more lines like:

```
no optimized radius available for atom     1 (si)
```

and then tell cosmoprep to use bondii for Si:

```
r "si" b
```

and check if the number of atoms is now equal to the number of radii.

10. Exit the radius assignment with * and enter a name of the cosmo file in the next section – or use the default by hitting <Enter>.

11. Now check the control file and search for cosmo. With those keywords in the control file, energy and gradient calculations at Hartree-Fock and DFT level will include solvation effects – you do not have to specify any additional option with jobex:

```
jobex -ri > jobex.out
```

12. Check the job.last file and look at the energy output there.

11. Calculation of NMR chemical shifts

11.1. At the HF or DFT level

Please refer to chapter 9 and section 12.2.18 (Keywords for Module **mpshift**) of the documentation for an overview of the capabilities and the usability of **mpshift**.

Prepare an input for a calculation of the shieldings of tetramethylsilane, a typical reference molecule for ^{13}C shifts:

1. DFT, non-hybrid functional, without RI

    1. Call **define** and load the minimum structure of TMS from the structure library by entering

        ```
        a !sime4
        ```

        at the geometry menu. Let `desy` determine the symmetry of the coordinates (T_d).
    2. Accept the default basis set def-SV(P). Usually a basis set of SVP quality is sufficient for DFT calculations when non-hybrid functionals are used, as long as no transition state elements with non-d^0 or d^{10} occupation are involved.
    3. Do an extended Hückel guess with `eht`, switch on `dft` in the last menu and accept the defaults (B-P functional with m3 grid).
    4. Call **dscf** to get converged Mos.
    5. Call **mpshift** or the script **chemshift**.
    6. The total, the isotropic, and the anisotropic shieldings of all nuclei can be found at the keyword
       `$nmr dft shielding constants`
       in the *control* file if you have called **mpshift**, or in the file *shieldings* if you have called **chemshift**.

2. DFT, non-hybrid functional, with RI

    1. Just repeat the last calculation, but do an **ridft** input and start it before running **mpshift**.
    2. **mpshift** does not use the RI-J approximation itself, but it can do NMR shielding calculations starting either from a conventional DFT run or from an RI-DFT run.

3. HF or hybrid functionals (B3-LYP, PBE0, ...)

    1. Prepare the input as in the DFT case, but choose the def-TZVP basis set this time and select `b3-lyp` as functional (or switch off DFT for a Hartree-Fock calculation). For HF and DFT with hybrid functionals, a TZVP basis set is recommended.

    2. A fully direct calculation is not possible, so you can either set the maximum file size of the *twoint* file to an arbitrary number (in MB, as usual), or let a statistics run determine the maximum file size needed for this input: Call
       ```
       stati dscf
       ```
       to add the `$statistics` keyword to the *control* file and call **dscf** to start the statistics calculation.

    3. Edit the control file and add a path to a local directory at the keyword
       `$scfintunit` at `file=twoint`
       If the *twoint* file is too big for your system, set it to a smaller number. **mpshift** (just like **dscf**) will use only as much space as provided by the keyword and computes the rest of the integrals directly.

4. Call **dscf** and then **chemshift**

5. Note that HF results are written to the keyword
   `$nmr rhf shielding constants`
   while DFT ones to
   `$nmr dft shielding constants`

11.2. At MP2 level

1. Create a usual **dscf** input like the one in the section before.

2. MP2 chemical shieldings need a very well converged density, so you have to do a single-point **dscf** calculation with at least `$denconv .1d-6`. This keyword is added by the script **mp2prep** when preparing usual MP2 energy or gradient calculations, but not when preparing MP2 chemical shifts. It can not be set within define, so you have to add it 'by hand' if you do just a **mpshift** calculation after a single-point HF run.

3. Do a statistics run for **dscf**, add `$denconv .1d-6` to the control file and run **dscf**.

4. `mp2prep -c` prepares the input for the NMR shieldings calculation at MP2 level. On many systems, `mp2prep` has got problems determining the free disk space, so it is highly recommended to use the `-p <MB>` option of `mp2prep`

   ```
   mp2prep -c -p 400
   ```

5. Start **mpshift**. It will do the SCF and the MP2 calculation of the nuclear magnetic shieldings.

11.3. Hints

- Only RHF calculations can be done.

- Do not use ECPs. The electrons in the core potential will not be taken into account during the calculation, so the results will just be nonsense.

- The integral routines of **mpshift** are some kind of 'old fashioned' and have a limitation that does not occur in other programs: The maximum contraction number of primitives in a basis is limited to 10, so big basis sets like cc-pVDZ, cc-pVTZ, cc-pVQZ and cc-pV5Z for elements beyond Ne can not be used at all.

12. Visualization of orbitals and densities

12.1. TmoleX

The easiest way to visualize surfaces of orbitals, densities and much more is to use TmoleX. Choose the property which you like to see in the Result section of TmoleX (e.g. by reading in the control file of a completed calculation) by clicking on the button called '3D Surfaces'.

12.2. Using molden or molekel

Just call the TURBOMOLE program **tm2molden** after a calculation. You will be asked about the data that should be included in the **molden** input file. If the basis set and the molecular orbitals are written to the input file, **molden** (or **molekel**) will be able to plot the orbitals and the density by calculating those properties itself.

12.3. Using gOpenMol

The TURBOMOLE modules **dscf**, **ridft**, **ricc2**, and **egrad** are evaluating the new keywords for property and visualization purposes (see chapter 10 of the documentation).

If you have an old calculation, you can add one of those keywords and call

```
dscf -proper
```

or any other of the programs mentioned above. Only the property/visualization step will be done in such a case.

To plot, say, the total density of water, just do a usual input and add

```
$pointval dens
```

to the *control* file and run the calculation.

After the run, start **gOpenMol**:

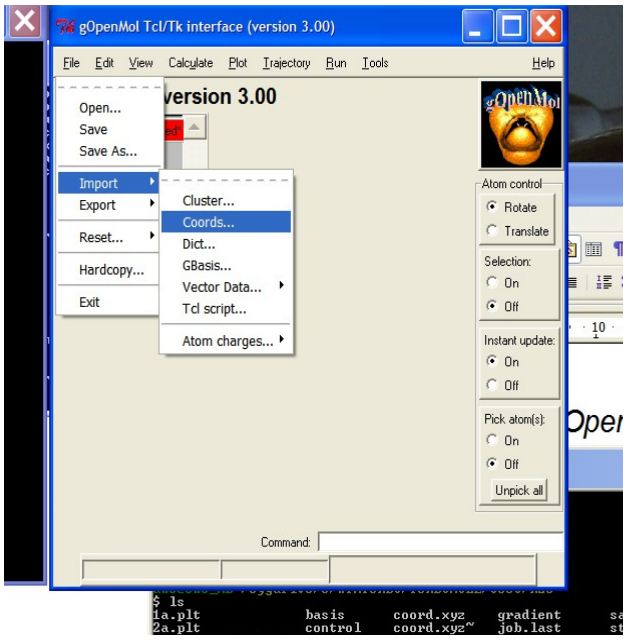
1. Load in the coordinates by using the Menus of gOpenMol:

   ```
   File -> Import -> Coords...
   ```

   You will get a menu where you can select the coordinates from. The TURBOMOLE programs will automatically write out an xyz file, so usually you can just load in the file *coord.xyz* and click on the <Apply> button.

   Note:

   gOpenMol can also read in directly the Turbomole coordinate file *coord*, but in the current version it expects it with *.txt*



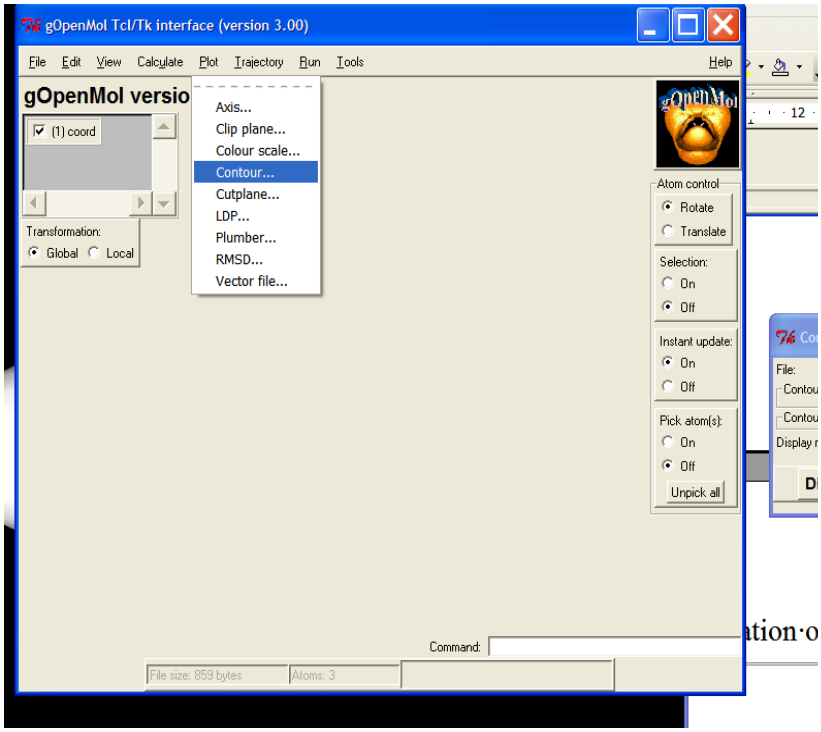
ending. You have to tell **gOpenMol** that the file *coord* is in TURBOMOLE format explicitly (switch off the field 'Select file format by extension' and choose TURBOMOLE).

2. Now go to the menu:

   `Plot -> Contour...`

   and you will get a new window where you can load in the plotted (total) density, called *td.plt*

   First click on `browse`, load *td.plt* and then on `import`.



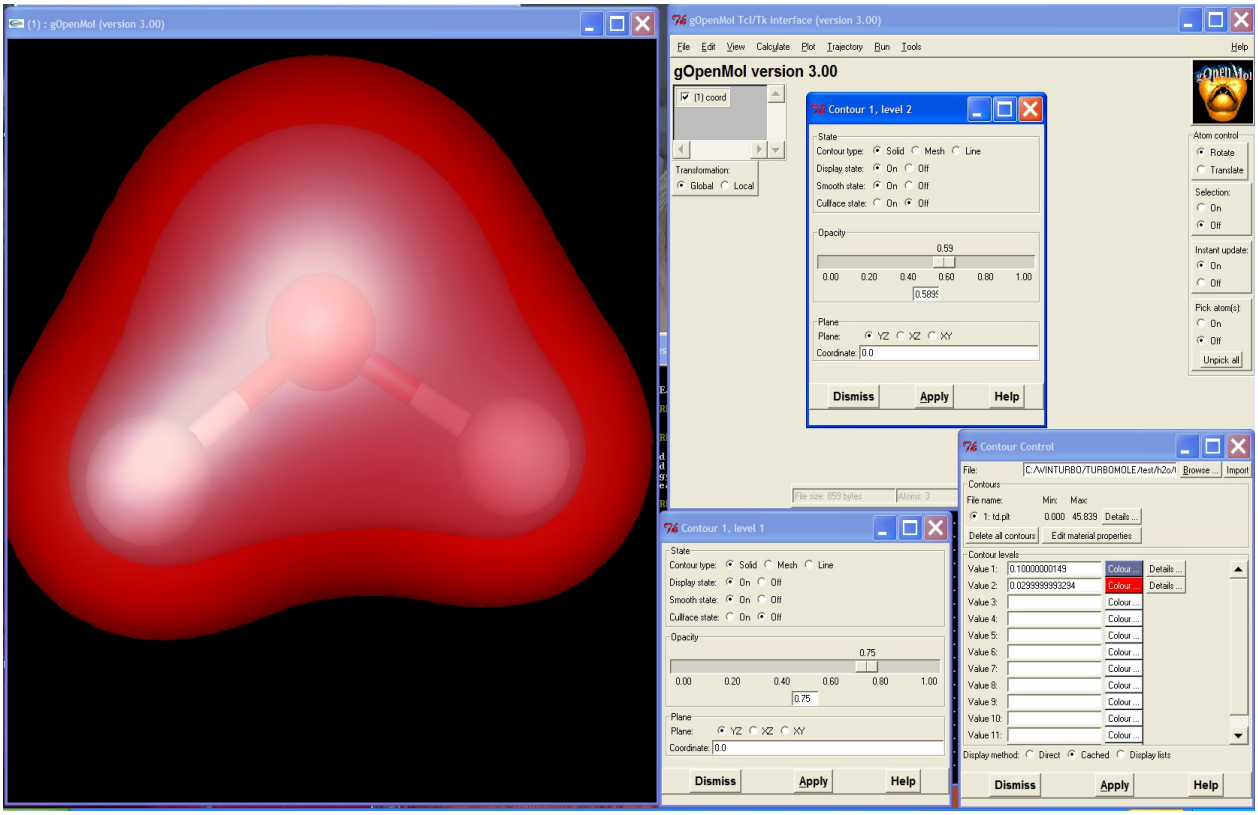
3. The Contour Control Window will expand, writing out the minimum and maximum value of the density that has been read in.

   For each of the 10 fields, called Value 1 to Value 10, you can now enter a value of the density that shall define the iso-density surface of the contour that will be plotted.

   With `Colour...` and `Details..` you can change the colour and the opacity of each contour.

   Do not forget to click on `Apply` each time you change something in the menu.

Here is an example of two iso-density surfaces of Water, one at a value of 0.1 a.u. in blue, and one at a value of 0.03 in red.



To plot MOs, it is recommended to do a single point calculation first and then look at the orbitals and the occupation by calling the script

```
eiger
```

You will get a list of the orbitals together with the occupation and the energy. This gives a hint about the MOs which might be interesting to have a look at.

Let us make an input for benzene (`a !benzene` from the structure library), call `desy` for the symmetry, choose def-SVP and RI/DFT-BP86. Run **`ridft`** and call **`eiger`**:

```
Total energy =   -232.0860361135 H =     -6315.3859631 eV

 HOMO-LUMO Separation
   HOMO:    14.     1 e1g   -0.23205215 H =     -6.31446 eV
   LUMO:    15.     1 e2u   -0.03843656 H =     -1.04591 eV
   Gap :                    +0.19361560 H =     +5.26855 eV

Number of MOs=     76, Electrons=     42.00, Symmetry: d6h

   Nr.   Orbital    Occupation       Energy
   [...]
   18.     1 b1g                  +0.107418 H =        +2.923 eV
   17.     4 e1u                  +0.085010 H =        +2.313 eV
   16.     4 a1g                  +0.042434 H =        +1.155 eV
   15.     1 e2u                  -0.038437 H =        -1.046 eV
   14.     1 e1g      4.000       -0.232052 H =        -6.314 eV
   13.     3 e2g      4.000       -0.303729 H =        -8.265 eV
   12.     1 a2u      2.000       -0.333745 H =        -9.082 eV
   11.     3 e1u      4.000       -0.377379 H =       -10.269 eV
   10.     1 b1u      2.000       -0.402110 H =       -10.942 eV
   [...]
```

As you can see, the HOMO is MO number 14 and the LUMO is MO number 15. So to plot the HOMO and the LUMO, add

```
$pointval mo 14-15
```

to the *`control`* file and call

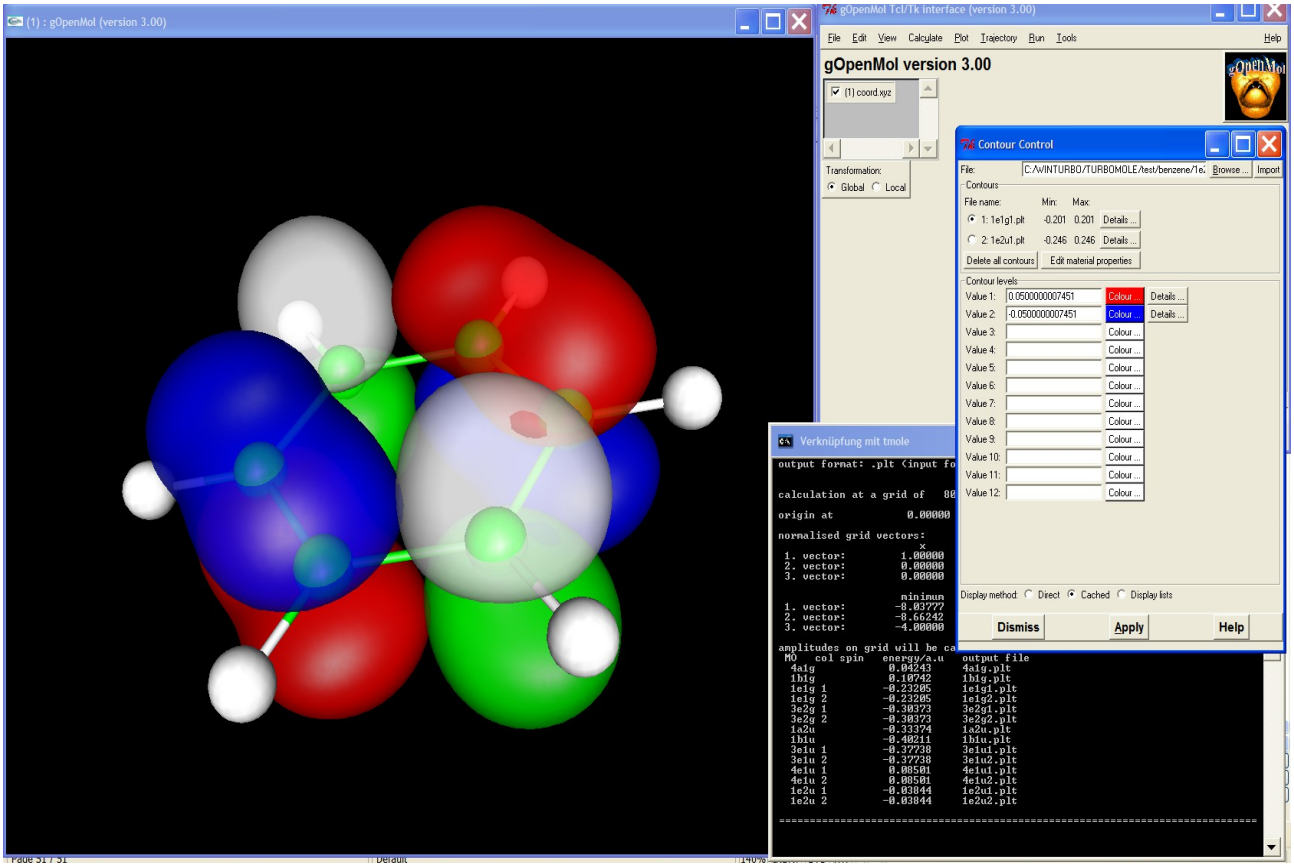
```
ridft -proper
```

If you look at the output, you will see which file name is associated to which orbital – usually that is quite clear from the file name anyway.

Start **`gOpenMol`**, load in the coordinates as shown above, go to the Plot/Contour menu and load the HOMO file. Import it and specify two different contours – one with a positive value and one with a negative. Change the color of one of them and let them plot by clicking on `Apply`.

Now you can load the LUMO in addition, import it and plot again two contours with different colors.

Play a bit with the values and colours. Below is a screen shot of such a plot. Red and blue is the HOMO and green and grey is the LUMO.



Look at the Turbomole documentation about how to plot differential densities of excited states, MP2 densities, spin densities, ...

12.4. Using an arbitrary program to plot 3D grids

To choose a simple xyz format instead of the one for **gOpenMol**, add

```
$pointval mo 13-14 fmt=xyz
```

to the *control* file and run an energy calculation.

This will write out the grid and the value of the MO in a file using the format:

```
x  y  z  f(x,y,z)
```

and, in addition, some lines at the beginning that contain the most interesting data of this plot:

```
#origin           0.000000      0.000000      0.000000
#vector1          1.000000      0.000000      0.000000
#vector2          0.000000      1.000000      0.000000
#vector3          0.000000      0.000000      1.000000
#grid1  start   -8.037772  delta    0.203488  points     80
#grid2  start   -8.662417  delta    0.201452  points     87
#grid3  start   -4.000000  delta    0.205128  points     40
#title for this grid
#amplitude of MO    1e1g 1         -0.23205a.u.
#plotdata
# cartesian coordinates x,y,z and f(x,y,z)
       -8.03777189         -8.66241737         -4.00000000          0.00000002
       -7.83428399         -8.66241737         -4.00000000          0.00000002
       -7.63079610         -8.66241737         -4.00000000          0.00000003
       -7.42730820         -8.66241737         -4.00000000          0.00000004
       -7.22382031         -8.66241737         -4.00000000          0.00000006
       -7.02033241         -8.66241737         -4.00000000          0.00000008
       -6.81684451         -8.66241737         -4.00000000          0.00000011
```

Load this in your favorite spread sheet or statistics program.

Since Turbomole V6.0 it is also possible to write out the 3D grid information in the commonly used cube format, just set fmt=cub.