Computing electronic excitation spectra with DFT and CC2

14 Jan 2010

Mikael Johansson with the help of Michael Patzschke

mikael.johansson@iki.fi

http://www.iki.fi/~mpjohans

TDDFT excitation energies

- The computation of electronic excitation spectra (UV/VIS) has now become possible for large molecules, *via* efficient implementations of time-dependent density functional theory (TDDFT). Despite its name, the most common usage of TDDFT is to compute excited state properties within the linear response framework. In this case of excitation energies, no time dependence in modelled.
- It can be noted that if you can get excitation energies, you can also get excited state *potential energy surfaces*.

- With modern software packages, TDDFT calculations can routinely be performed for molecules with hundreds of atoms. With special software and a lot of hardware, even larger systems can be studied.
- Thus, it is possible to study for example biological systems (at least parts thereof) and large molecular assemblies

- The quality of excitation energies that you can get from TDDFT depends somewhat on the exchange-correlation functional used
    - Relative excitation energies are reproduced better than absolute values
    - Hybrid functionals usually perform better compared to normal GGA functionals. The part of exact exchange that they include help to correct for the wrong asymptotic behaviour of most common functional
- All-in-all, however, the results you get from TDDFT have been found to be quite satisfactory, especially considering how cheap they are to compute.
- But there are some special types of excitations that the approximate density functionals of today describe *quite poorly*
    - Conical intersections at the excited PES
    - Excitations that have double excitation character
    - Rydberg excited states
    - **Charge transfer excitations**
- In charge transfer excitations, the incoming photon excites an electron which then moves over a large distance in the molecule; *TDDFT severely underestimates these excitation energies!*

CC2 excitation energies

- The singles and approximate doubles coupled cluster model CC2 is an alternative to TDDFT for computing excitation energies on large systems

- It is not as efficient as TDDFT (yet, at least), but quite large systems can already be treated. Of the wave function based methods of computing excitation energies it has, arguably, the best price/performance ratio

    - It *can* describe charge transfer excitations
    - Still problems with double excitations; but you will get a diagnostic that tells you if this is a problem

First, we need to learn the basics of how to use TURBOMOLE

The program system TURBOMOLE – An introduction

TURBOMOLE can compute a lot of stuff, mainly at **HF, DFT, and CC2 level**. You can get excitation energies, vibrational spectra, circular dichroism spectra, excited state geometries, *etc*. It is quite fast, and has the nice ability to use very high symmetry groups, **not only Abelian** (non-degenerate) point groups.

- **Home pages:**
  - http://www.turbomole.com
  - http://www.cosmologic.de (More up-to-date, at least for now)
    - Manuals
    - Tutorial
  - http://www.turbo-forum.com/

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 a limited number of methods and/or properties
    - Scripts are used to combine the functionalities of the programs
    - Input can be changed by text editors
    - Output processed by standard UNIX tools (editors, grep, awk, ...).

Setting up the TURBOMOLE environment on Murska

- Today, we will be running most calculations interactively
- Log into Murska, and request an interactive session
    - `bsub -n 4 -W 4:00 -q interactive -Ip xterm`

- To get access to the interactive programs, give the command:
    - `module load turbomole/6.0` (6.1 is available, but broken...)

Defining your calculations

- **First**, you will need a file which contains the coordinates of your molecule
- In Turbomole, this file is named `coord`, and has the following basic structure:

```
$coord
    0.00000000000000      0.00000000000000     -0.12953952211169      o
   -1.43361127788040      0.00000000000000      1.02808348962460      h
    1.43361127788040      0.00000000000000      1.02808348962460      h
$end
```

- Note that the Cartesian coordinates are given in **atomic units, not Ångströms**!
  - The tools `x2t` and `t2x` for transforming between TM format and standard XYZ format are handy
  - OpenBabel can handle many more formats
    - `module load openbabel`
    - `babel –ixyz molecule.xyz –otmol coord`
    - `babel –ipdb enzyme.pdb –otmol coord`

- **Second**, you will use a program called `define` to specify what kind of calculation you want to perform. Start it in your work directory with the command `define`

- The `define` program will interactively ask you a lot of stuff
  - What are your coordinates?
  - What is the symmetry of your molecule?
  - What basis set do you want to use? **NOTE: Use the new def2 basis sets instead of the old def!**
  - The charge of your molecule?
  - What level of calculation do you want to use?
  - *etc.*

We will go through these interactively during the session, but some points to consider:

- At least the **m4** grid is recommended
- The following XC functionals can be used:
  - LDA (s-vwn, pwlda)
  - BP86 (b-p)
  - PBE(0) (pbe, pbe0)
  - B3LYP (b3-lyp)
  - BHLYP (bh-lyp)
  - BLYP (b-lyp)

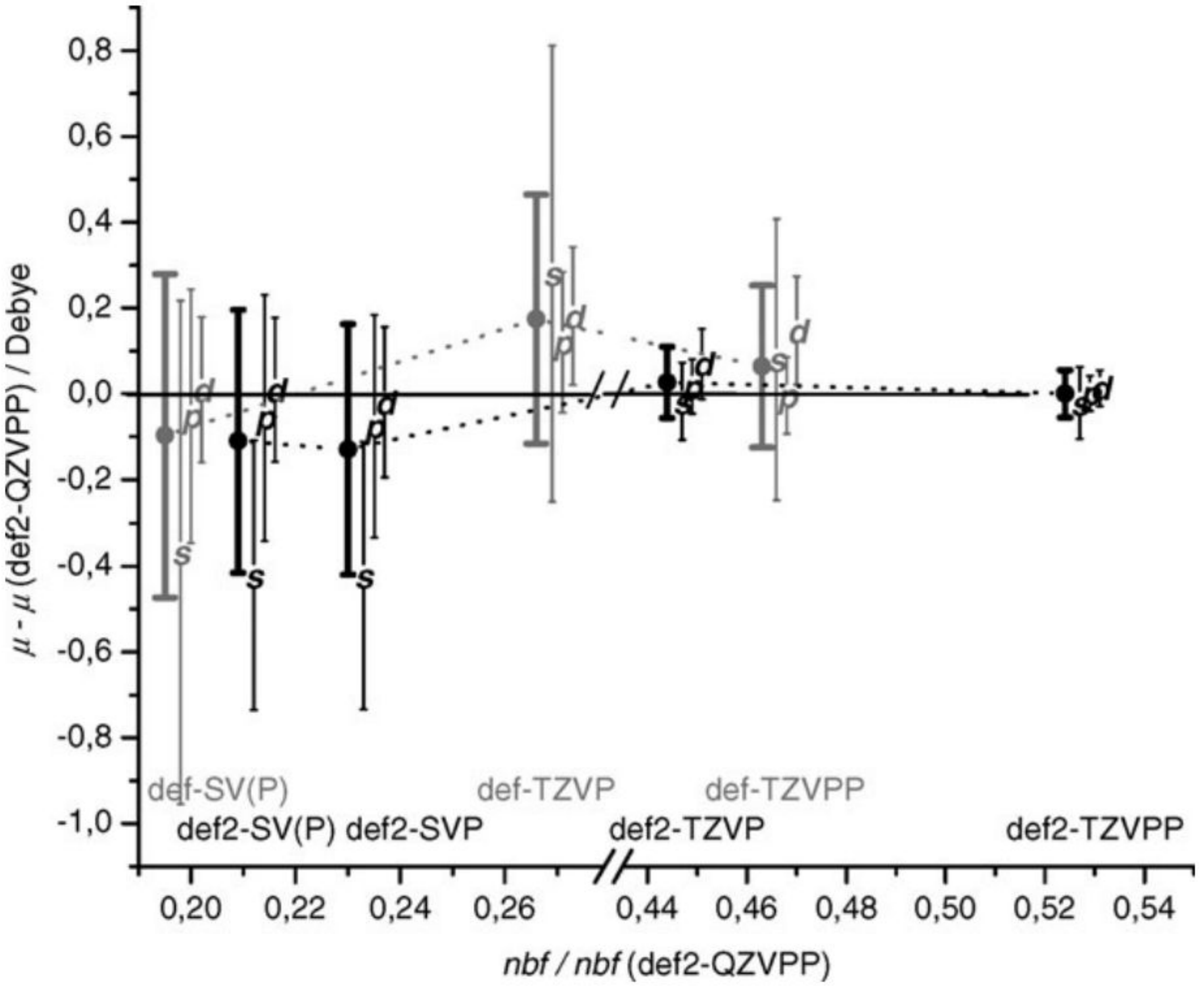


Fig. 4 Errors in dipole moments, μ, in D, at the DFT(BP86) level for

- After running `define`, you will have a file named **`control`** in your directory. This contains the specifications of your calculation.

- Most of the parameters that the `define` program has put in here are OK, but a few have to be changed:
  - Remove the line that says `$scfdump`. This **keyword** tells Turbomole that it should save the orbitals after every SCF cycle to disk. This will slow down your calculations, as the disk system on Murska is horribly slow.
  - To further prevent disk usage, edit the line starting with `$thize`, increasing its parameter to:
    - `$thize      0.10000000E+99` **(only relevant for non-RI calcs!)**
  - For the response calculations, a well converged reference state is needed, so tighten the energy and density convergence criteria, by editing the following keywords:
    - `$scfonv 7` (a higher value usually doesn't hurt, but could be beneficial)
    - `$denconv 1.d-7`

Running your calculations – HF and DFT single points

- SCF single points can be computed with two programs
    - `dscf` non-RI for Hartree—Fock and hybrids
    - `ridft` RI-DFT for pure functionals (for GS also for HF exchange)

- This must be run before the calculation of the excitation spectra, to get the ground state configuration

- When the job has finished, it is good to check that the calculation has converged. A quick check is to see if the file named `energy` in your directory contains a line with the final energy. It is also a good idea to have a look at the `dscf.out / ridft.out` file for extra information about the computation.

- Also, always use `eiger` for a quick check that you have not ended up in an excited state!
    - For the excitation energy calcs, the HOMO—LUMO gap also needs to be positive; at DFT level, this is not always possible...

DFT geometry optimisations

- It might be a good idea to optimise the geometry of your molecule
- At DFT level, this is simple, just run the `jobex` script:
  - `jobex -dscf`   non-RI DFT
  - `jobex -dscf -ri`   RI-DFT

- After optimisation, the directory should contain a file named `GEO_OPT_CONVERGED`
  - If it doesn’t something didn’t go smoothly...

- For some speedup, the SCF convergence thresholds don’t need to be as tight as for the response calcs!

DFT singlet excitation energies

- To get Turbomole to compute excitation energies after a successful ground state DFT calculation *via* `dscf`, it needs some extra information on what to calculate. This can be done *via* `define` or by editing the `control` file by hand. A few lines have to be inserted (before the `$end` keyword!):
  - `$scfinstab rpas`
  - `$soes`
  - ` a1            3`
  - ` b1            3`
  - `$rpacor    500`

- `$scfinstab rpas` tells Turbomole to perform a TD-DFT calculation for **singlet** excitations
- The `$soes` keyword defines how many excitations in each irrep are to be computed (the existing irreps depend on the symmetry of the molecule)
  - **NOTE!** After a (successful) run, you can increase the number of excitations in an irrep and start the calculation again; TM will then reuse the already computed excitations, which saves time!
- `$rpacor` tells the program to use about 500 MB of memory for the calculation. The more, the faster, usually.

- You can also let TM create a file for you with all the excitation energies in brief form, by adding:
  - `$spectrum eV  /  $spectrum nm  / $spectrum 1/cm  /  $spectrum a.u.`
- Unfortunately, only irreps with allowed transitions are included

- The name of the program that computes excitation energies (at HF and DFT levels) is called `escf`
  - `escf > escf.out`
- Check the `escf.out` file for the information you need

DFT triplet excitation energies

- Triplet excitation energies are obtained simply by changing the `$scfinstab` keyword:
  - `$scfinstab rpat`

DFT excitation energies from open-shell ground states

- Electronic excitations for open-shell species can also be obtained easily
- After computing the GS, excitation energies are obtained in the same way as for closed-shell species, with the following change to the `$scfinstab` keyword:
  - `$scfinstab urpa`

DFT circular dichroism (CD) spectra

- Easy, just add the following to `control` and run `escf`:
  - `$cdspectrum eV  /  $cdspectrum nm  / $cdspectrum 1/cm  /  $cdspectrum a.u.`

Visualizing how the electron moves when excited with DFT

- It is possible to compute the electron density difference between the ground and excited state
- This will show how the electron moves within the molecule when excited
- For this, it is necessary to calculate excited state gradients. Even if we will not optimize the geometry, the gradient calculation provides the information for the density difference
- You need to again edit your control file a bit. First, insert the following line to compute the difference densities:
    - `$pointval`
- Next, you need to specify which excitation you want to study closer (**only one at a time!**). So you need to edit the `$soes` keyword to only contain one irrep. In this context, the highest excitation specified will be computed. For example:
    - `$soes`
    - `  b1          2`
- This will compute the gradient, and at the same time the density difference for *only* the second state in the irrep b1 of your molecule
- To compute others, you need to change the `$soes` keyword and redo the calculation

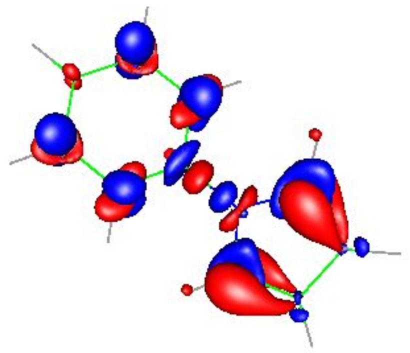
- The name of the program that computes excited state gradients (at HF and DFT levels) is called `egrad`:
    - `egrad > egrad.out`

- After finishing the job, you should have a file called `ed.plt` which contains a 3D grid of the density difference.
- In an open-shell calculation, you will also have a file called `esd.plt`, with the differential spin density

- **Note!** If you change the `control` file to compute the density differences for other irreps, **remember to save** your `ed.plt` file with another name; it will be overwritten!

- To have a look at the electron density difference, we will use the gOpenMol program installed on Murska. We will go through its usage interactively, but to start up the program, use the following commands. The first initializes the environment, the second starts the actual program
  - `module load gopenmol`
  - `rungOpenMol`

Example: The second lowest excitation in phenylpyrrole, computed at B3LYP level:



CC2 excitation energies

- To get CC2 excitation energies after a successful ground state Hartree—Fock calculation *via* `dscf`, you again need to edit the `control` a bit. Insert the following lines:
    - `$ricc2`
    - `  cc2`

- The `$ricc2` keyword defines the model to use. The ricc2 program can actually do computations using several different wave function models; today we want CC2
    - See the manual for other options:
        - CCS/CIS, CIS(D), ADC(2), ...

- To get **only (singlet) excitation energies**, insert the following:
    - `$excitations`
    - `  irrep=a1   multiplicity=1 nexc=3`
    - `  irrep=b1   multiplicity=1 nexc=3`

- The `$excitations` keyword is the CC2 equivalent of the `$scfinstab` and `$soes` keywords used for DFT, above. Again, make a line each for all the irreps
  - `multiplicity=1` says that we want singlet excitations, `nexc=3` defines how many excitations in each irrep should be computed.
  - For triplet excitation energies, just change the multiplicity to 3

- To get also **oscillator strengths**, add this to the `$excitations` group:
  - `  spectrum  states=all`
- If you are computing triplet excitations, this line **must be removed**, otherwise TM crashes

- To get the total **density of the ground state**, add:
  - `$response`
  - `  static relaxed`
- This should give you a file named something like `cc2-gsdn-1a1-000-total.cao`

- To get **densities of excited states**, gradients have to be computed, just as in DFT; add this to `$excitations`:
  - `  xgrad states=(b1 2)`
- Again, only one state at a time can be computed
- You will get a file named something like `cc2-xsdn-1b1-002-total.cao`

- **For parallel runs, `$SHAREDTMPDIR`** tells Turbomole that temporary files will be written in the same directory, and has to be inserted manually into the `control` file!
  - `$SHAREDTMPDIR`

- As for DFT, you can use the `$spectrum` and `$cdspectrum` keywords; at CC2 level they even seem to work properly!

- The name of the program that computes excitation energies at CC2 level is called `ricc2`
- In contrast to DFT, the same program computes "everything", that is, depending on the contents of the `control` file, either only GS properties, or excitation energies, or gradients, etc.
  - `ricc2 > ricc2.out`

- Check the `ricc2.out` file for the information you need

- **Note:** Presently, only non-degenerate point groups can be used for CC2 excitation energies
  - **Practical consequence:** If the symmetry is higher, you need to include all excitations that would have the same energy (in different subsymmetries, even C1) in your calculation, otherwise convergence problems!

Visualizing how the electron moves when excited with CC2

- To get the **density difference** between the GS and the XS, `ricc2` has to be run in analysis mode after inserting the following keywords into `control`:

```
o $anadens
o  calc ED-1b1-2 from                     <- You can naturally change the file name
o    1d0 cc2-xsdn-1b1-002-total.cao <- Check that you have these files in your
o   -1d0 cc2-gsdn-1a1-000-total.cao    working directory!
o $pointval
```

- You will get a file called `ED-1b1-2.plt` by running:

```
o ricc2 –fanal > ricc2-fanal.out
```

Optimising excited state geometries at DFT level

- TM has analytical gradients for excited state optimisations
- Running the optimisation is (in principle) just as easy as for ground states
- After **selecting which state to optimise**, specified as above, just run `jobex` with an extra parameter `-ex`:
  - `jobex -ri -ex > jobex.out`

- **NOTE:** Excited state optimisations can be **much more problematic** than for ground states
  - When the geometry changes, **the order of the excited states** might change too!
  - As the excited state to be computed is specified **and fixed**, the optimisation might suddenly start to optimise an unwanted state!
  - Sudden jumps in gradients are a sign of trouble, and require manual inspection!
    - `grep dE gradient`
  - Further problems with excited states becoming near-degenerate; not fun
  - Also, with a large enough distortion of the geometry during optimisation, the (aufbau) orbital occupations of different symmetries might change!

Optimising ground/excited state geometries at CC2 level

- To optimise **GS geometries** at CC2 level, the `$ricc2` field in `control` needs to be modified to:
  - `$ricc2`
  - `  cc2`
  - `  geoopt model=cc2`
- Check that you do not compute any unnecessary things! (`$excitations`, ...)
- After this, jobex should be run with the following parameters:
  - `jobex -level cc2 > jobex.out`

- **Excited state geometries** need slightly more input in `control`:
  - `$ricc2`
  - `  geoopt model=cc2 state=(b1 2)`
  - `$excitations`
  - `  irrep=b1 nexc=2`
  - `  exprop states=all operators=diplen,qudlen`

- After this, jobex should be run as for the GS (**no `-ex` option!**):
  - `jobex -level cc2 > jobex.out`

Computing excited state vibrational frequencies at DFT level

- Presently, analytical second derivatives are not available, so **frequencies have to be computed numerically**
- This is done with the `NumForce` script, which automatically perturbs the optimised geometry as needed
    - Therefore, the **symmetry will be lowered to C1 for the single points**
    - Thus, the excitation "number" in C1 needs to be known!
- We will interactively go through how to transform a symmetric molecule to C1 with `define`, and how to figure out which excitation is the relevant one

- **For NumForce, a very tight SCF convergence is crucial!** Minimum:
    - `$scfconv 8`
- The effect of the DFT grid is also notable, even m4 might be too small for converged results!

- After this, `NumForce` is started with the following parameters:
    - `NumForce -ri -ex 2 > numforce.out`
- `-ex 2` tells `NumForce` that it is the second excited state (in C1 symmetry) that we want

- If the geometry distortions lead to a flip in the order of the excited states, grim times are ahead
    - One can try to use a **smaller step size** for the displacements and hope for the best!

Computing excited state vibrational frequencies at CC2 level

- Just as at DFT level, `NumForce` must be used, and thus C1 symmetry!
- The control file should thus contain something like:
  - `$ricc2`
  - `  geoopt model=cc2 state=(a 2)`
  - `$excitations`
  - `  irrep=a nexc=2`
  - `  exprop states=all operators=diplen,qudlen`
- Again, make sure that the excitation number (`nexc`) corresponds to the state you want!
- Then, run `NumForce` in the same way as you would for a GS, **again, with CC2, no `-ex` parameter**:
  - `NumForce -level cc2 > numforce.out`