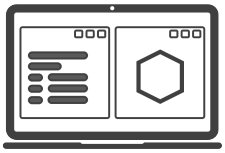


Introduction to Computational Chemistry
Exercises Part 2

Eno Paenurk & Patrick Finkelstein & Felix Pultar

May 3, 2023

Goals:

- Use the `conda` package manager to install `autodE`, `icctools`, and `xtb`
- Use `icctools` to calculate simple reaction profiles
- Understand the basic principles of locating transition state
- Recognize and solve challenges during more complicated calculations

1 Introduction

In this exercise, we will use `autodE` powered by `RDKit`, `ORCA`, and `xtb` to calculate reaction pathways. `RDKit` provides cheminformatic tools such as `SMILES` parsing and conformer generation. `ORCA` is a powerful software package that provides efficient methods for electronic structure calculations. `xtb` is a new and extremely fast computational method and software package that we will use to quickly generate good first guesses for further calculations. We will discuss these software packages in depth during the next weeks. For now, we will just use them in the background to do the heavy lifting for us. `autodE` automates the computational pipeline of modelling chemical reactions but its effective usage requires some skill with the command line. To install `autodE`, `RDKit`, `xtb`, along with their respective dependencies, we will use the `conda` package manager. We will also use the package `setuptools` to install `icctools` from the `git` repository.

2 Updating the `git` repository

Make sure you have the latest version of the `git` repository we provided last week.

Visit `https://github.com/pultar/icc-2023` and donwload the material as you learned last week. Transfer the content of the downloaded `ZIP` file to the folder with the material you created last week on your cluster.

Change directory to the "materials" folder you created last week and enter:

```
git pull
```

You should see two new folders named "session-2" and "icctools".

3 Using conda

3.1 Introduction

Programs that are not installed through the central module system of your cluster or on your computer can often be installed and managed using `conda`. Most programs will depend on other packages to do their job. For example, `autodE` requires an installation of `Python`, `RDKit`, and many more modules. `conda` knows about these dependencies and will install all required packages in the newest possible version. Furthermore, once installed, `conda` can isolate these packages in so-called environments leading to a simpler setup and management of the necessary software. It is possible to install `conda` with many packages pre-installed (`Anaconda`). However, to reduce clutter, we have decided to go with a minimal installation using `Miniconda`. We have already installed `Miniconda` centrally (on Euler @ ETHZ) and created an environment named `icc-2023`. This environment will contain all packages not installed on Euler but necessary for the course. Grace users will already find a module `Miniconda` preinstalled.

> If you are an advanced user and have installed your own version of `conda` prior to this course, jump to the section "Installation on Local Machine" to find out which packages we will use.

3.2 Installation on Euler

We have installed the software for you already in a shared folder on Euler:

```
/cluster/project/workshops/icc/icc-2023
```

You only have to tell your Bash environment where to find `Miniconda`.

```
/cluster/project/workshops/icc/icc-2023/software/miniconda3/bin/conda init
```

This command will modify your `.bashrc` file. Feel free to check out what changed. As you know from the first week, modification of `.bashrc` requires either logging out and logging in again or:

```
source ~/.bashrc
```

Now you should see a `(base)` showing up next to your terminal prompt suggesting the `base` environment is active. Here, many common programs are installed, for example `conda` itself. To verify your installation, type:

```
which conda
```

The expected outcome is: `/cluster/project/workshops/icc/icc-2023/software/miniconda3/bin/conda`
The final step is to load the `icc-2023` environment. The required command is:

```
conda activate icc-2023
```

The following command deactivates the new environment and switches to the base environment:

```
conda deactivate
```

Repeated deactivation deactivates `conda` for the current session. Log out and in again to re-activate `conda` base environment. If you don't want `conda` be activate every time you login, type:

```
conda config --set auto_activate_base false
```

You will have to manually activate `conda` later using the full path.

3.3 Installation on Grace

`Miniconda` is already installed on Grace and accessible by loading the respective module:

```
module load miniconda
```

Once `conda` is loaded, create an environment for the course:

```
conda create -n icc-2023
```

and activate the new environment:

```
conda activate icc-2023
```

The software required for the course can be installed like so:

```
conda config --add channels conda-forge;
conda config --set channel_priority strict;
conda install autode xtb setuptools;
```

This list of commands will let `conda` know where to get packages from (`conda-forge`), and install `autodE` and `xtb` with all their dependencies.

3.4 Installation on Local Machine

Go to `https://conda.io/en/latest/miniconda.html` and download the correct version for your operating system (Mac OS X, Linux or Windows). For Linux and Mac OS X, you generally want the `bash` version. If you have a new Mac with the M-Series chip, make sure to hit the correct link. For Windows, 64-bit architecture is probably what you are looking for.

The installation process will differ depending on your operating system. Windows presents you with a graphical interface that you can follow. Mac OS X and Linux require use of the command line:

1. Navigate to your `Downloads` folder
2. Run `sh Miniconda3-latest-MacOSX-arm64.sh`
3. Follow the instructions of the installer, the default are typically fine
4. Have the installer run `conda init` when prompted

> If you want to install `Miniconda` in a location different than your `home` folder (the folder with your user name), preceed the command with `sudo` and type your password once requested:
>
> ```
> sudo sh Miniconda3-latest-MacOSX-arm64.sh
> ```

At this point you want to restart your terminal in order to have `.bashrc` reloaded. After the restart, you should see `base` next to your terminal prompt indicating installation of `conda` has been successful. The following command should give you the location of `conda` based under the installation prefix you provided:

```
which conda
```

> Note: on a Mac the output might resemble a shell script and not a file location.

Once `conda` is installed and activated, create an environment for the course:

```
conda create -n icc-2023
```

and activate the new environment:

```
conda activate icc-2023
```

The software required for the course can be installed like so:

```
conda config --add channels conda-forge;
conda config --set channel_priority strict;
conda install autode xtb setuptools;
```

This list of commands will let `conda` know where to get packages from (`conda-forge`), and install `autodE` and `xtb` with all their dependencies. This step might take some time depending on your internet connection and the computer you use.

3.5 Installation of `icctools` (Grace and Local Machine)

Make sure you have the `conda` environment activated (name of the environment shown next to your current terminal prompt) and change directory into “icctools”. From there, run:

```
python -m pip install -e .
```

3.6 Verification of the Installation

After the installation is complete, you can verify your installation with:

```
python -c "import autode as ade; print(ade.__version__)";
python -c "import rdkit; print(rdkit.__version__)";
python -c "import icctools; print(icctools.__version__)";
xtb --version
```

> Be warned that copy & paste from a PDF can introduce additional whitespace, which may lead to an error in these commands. If the output is different than expected, just type in the commands one by one to exclude this possibility.

These first two commands will execute very short Python snippets that try to load `autodE`, `RDKit`, and `icctools`, respectively, and query their versions. The last command will print the version of `xtb`. If finished successfully, the output should read `1.3.5` (`autodE`), `2022.09.5` (`RDKit`), 0.0.1 (`icctools`), and `6.6.0` (`xtb`). You might get a newer version if you go through this installation process in the future.

4 A Primer on Python

`Python` is a general purpose programming language. It is not necessary for this course to learn how to program as we will provide code templates that you can adjust for your needs. Some minimal guidelines:

- To execute `Python` code, put it in a `.py` file and run `python file.py` from the same folder
- Code is executed sequentially from top to bottom
- Lines that start with '`#`' are comments and will be ignored
- Numerical and string values such as `SMILES` strings, temperatures, number of processors can be assigned to placeholders called variables
- Some variables have associated functionality that can be called using the dot (.) operator with parameters given in parentheses (see table below)

Command	Description
`from icctools import librxn`	Use code provided by the `librxn` module of `icctools`
`rxn_smiles = "CCl.[F-]>>CF.[Cl-]"`	Assigns the variable `rxn_smiles` a string value
`rxn_temperature = 298.15`	Assigns the variable `rxn_temperature` a numerical value
`if condition:`	Runs following indented lines if condition(s) are met
`print(values, separated, by, comma)`	Prints the value(s) in the parentheses
`rxn.calculate_reaction_profile(...)`	Calculates reaction profile of variable `rxn` (parameters omitted here for brevity).
`librxn.print_results(rxn)`	Prints the results of the calculation performed

5 Running calculations with `icctools` and `autodE`

5.1 Getting the Templates

Copy `autode-sample.py` from "session-2/code" of the materials folder into a new folder in your workspace, e.g., "part2". We also recommend to create new folder within "part2" for every calculation you want to run. For example, if you want to run a calculation named "isoxazoline", do:

```
mkdir isoxazoline
```

5.2 Setting up the Submission Script

We provide you with a script that makes submitting `autodE` calculations to Slurm easier. Copy either `subade_euler.sh` or `subade_grace.sh` (depending on your cluster) from "session-2/code" to the `bin` directory in your home directory, rename it to `subade.sh`, and give it executing permissions (see Week 1). Now you can call `subade.sh` in any directory. Test this out by typing:

```
subade.sh
```

in any directory. This should print the usage description of the script. You will learn to use it below.

If you are using MobaXterm or FileZilla, use the graphical interface to rename using the options shown after a right-click on the respective file.

Use the command `mv` (move) to rename the files

```
mv subade-euler.sh subade.sh
```

5.3 Setting up `xtb` with `ORCA`

In this exercise, we will use `xtb` to calculate reaction pathways. Keep in mind that `xtb` was not parametrized to give good total energies (it was parametrized to give good geometries, frequencies, and non-covalent interactions – for that reason, the `xtb` methods are prexifed by that's what the acronym GFN in the `xtb` methods stands for). However, it will allow us to run calculations quickly and get used to the tools.

Unfortunately, you have to do a little hacking to enable the interface to `xtb`. Fortunately for you, we have figured it out and made it easy. Copy the "setup_orcaxtb.sh" script from "session-2/code" to the `bin` directory in your home directory. This script will set up a folder, which links the global installation of `ORCA` on the cluster with your local installation of `xtb`. Make sure that the `ORCA` module is loaded and the "icc-2023" `conda` environment is active. Enter the `bin` directory, make the script executable, and type:

```
./setup_orcaxtb.sh
```

You can now use `xtb` as the high-level method in `autodE`, and you can remove the setup script.

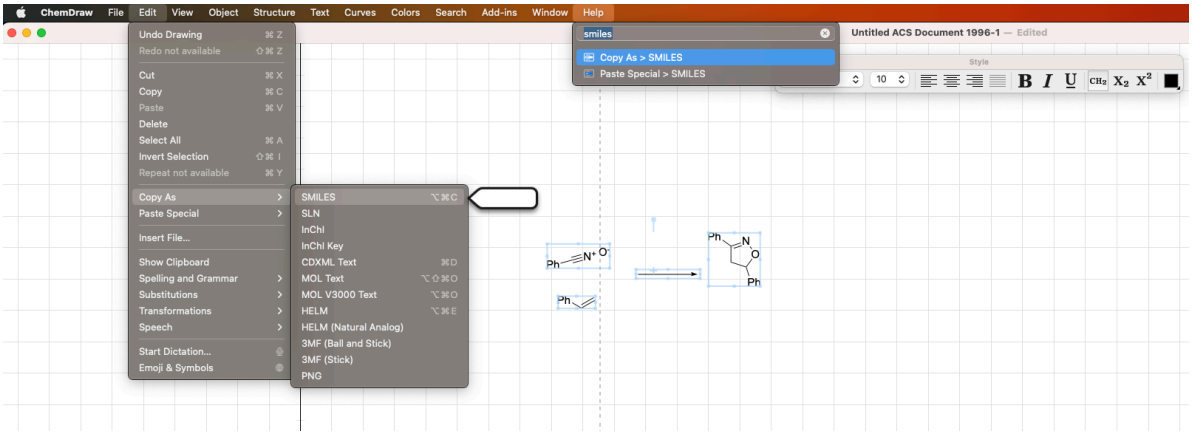
What is `bin`?
The `bin` directory is a standard directory in Unix-like systems. Any executable in that directory can be called from anywhere without specifying the path to it. More generally, executables can be globally called from any directory that is contained in the PATH variable. After having set up `xtb`, your PATH variable should also contain a few xtb-specific directories. You can check your PATH variable by typing:

```
echo $PATH
```

5.4 Final Steps

5.4.1 Python Script

The `Python` script defines how the actual calculation is performed. For defining the reaction, you only have to change the variables `rxn_name`, `rxn_smiles`, `rxn_solvent`, and `rxn_temperature` (in Kelvin) accordingly. The `rxn_name` variable defines the naming for the output files and folders, and you can choose it freely. The reaction `SMILES` for a reaction "a + b → c + d" is defined as: `a.b>>c.d` (molecules separated by a dot and starting materials separated from products by >>). The easiest option to obtain the reaction `SMILES` is via ChemDraw. Draw the reaction as shown below, select all molecules including the reaction arrow, and find the option "Copy as > `SMILES`" from the menu bar (shortcut Opt + Cmd + C on Mac, Alt + Ctrl + C on Windows).



Make sure ChemDraw recognized your reaction as such. As a rule of thumb: if the option "Clean up Reaction" (Shortcut Shift + Cmd + X on Mac, Shift + Ctrl + X on Windows) does the right thing, `SMILES` extraction will also work. If only parts of the reaction are recognized, it is sometimes easier to select each molecule individually and create the reaction `SMILES` yourself using the dot and the >> operators.

> Note: `SMILES` might not work for some compounds, e.g. spirocyclic compounds, certain organometallics, and molecules with many adjacent stereocenters. We will show you ways how to also incorporate those challenging molecules in your calculations during the next session.

For solvent name definitions, refer to the manuals of `xtb` (`https://xtb-docs.readthedocs.io/en/latest/gbsa.html`) and `ORCA` (`https://www.orcasoftware.de/tutorials_orca/prop/CPCM.html`). Note that some trial & error may be required as different software packages call the same solvent by a different name (e.g. dichloromethane vs CH_2Cl_2).

You additionally have to specify the resources by setting the variables `n_cores` (number of cores for parallelization), `memory_per_core` (memory in MB per core), and the calculation method. We list several options in the script, but we recommend starting with `xtb` (`librxn.Method.XTB`) for the sake of speed. The following computational methods are provided for your convenience:[1]

Python Code	Description
`librxn.Method.XTB`	semiempirical (very fast)
`librxn.Method.BP86`	cheap DFT (reasonably fast)
`librxn.Method.B3LYP`	DFT of choice for organic chemists (slow)
`librxn.Method.PBE0`	DFT of choice for inorganic chemists (slow)

5.4.2 Running the Calculations

If you haven't done so already, load `ORCA` and its dependencies via the module system.

```
env2lmod
module load gcc/9.3.0 orca/5.0.3 openmpi/4.1.4
```

Make sure the correct `conda` environment is activated (see above). Note that these steps may need to be repeated every time you log in unless you have modified your `bash` configuration as demonstrated last week. Make sure you are in the directory with the Python script and submit the calculation by typing:

```
subade.sh isoxazoline.py
```

Depending on your system and the computational resources provided, you will have to wait for some time. `autodE` produces many files and folders during and after the calculation. The generated file that ends with "`reaction_profile.pdf`" contains the generated plot shown in the presentation provided the calculation was successful. Here is an overview of the files assuming `rxn_name` was set to `isoxazoline`:

[1] Don't worry about the acronyms now, we will have many more sessions to cover what these things mean.

File	Description
`autode.log`	A log file documenting the status and potential issues of the calculation
`isoxazoline.out`	Key electronic energies, enthalpies, and free energies of the reaction. Number and value of imaginary frequencies.
`isoxazoline.err`	Errors related to your calculation (should be empty)
`isoxazoline_reaction_profile.pdf`	Plot of the reaction profile. May contain warnings if more than one imaginary frequency was found or no transition state could be located.
`isoxazoline/conformers`	Structure files of conformers found
`isoxazoline/output`	Final output of the calculation (energies, structural files, visualization of imaginary mode, …)
`isoxazoline/reactants_and_products`	Geometry optimizations of reactants and products
`isoxazoline/single_points`	High-accuracy computations on reactants, products, and transition states
`isoxazoline/thermal`	Hessian computations on reactants and products to estimate Gibbs free energies
`isoxazoline/transition_states`	Structure files and Hessians of transition states found

`icctools` also saves transition state templates in a folder in your home directory, e.g.:

```
/cluster/home/fpultar/.icc-2023/autode-lib
```

Future calculations of the same or similar systems will be much faster with these saved templates. Make sure the pdf with the reaction profile was generated, the `.out` file containts the calculated energy values, and the `.err` file is empty.

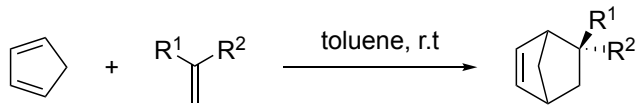
6 Mechanism calculations

We will study different Diels-Alder reactions and try to computationally reproduce experimentally observed trends. One of these reactions was used by Dauben to provide an efficient total synthesis of the natural product cantharidin, a poison isolated from the Spanish fly.

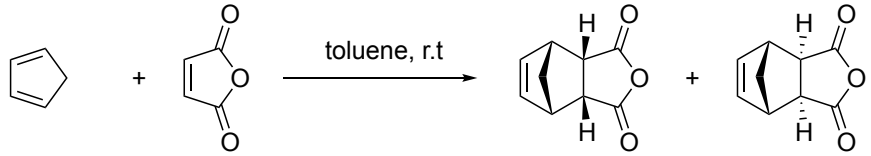
Disclaimer: Studying reaction mechanisms is highly complex and we are cutting many corners in order to make these computations accessible to a wider audience. The overall goal of the course is not to transform you into a computational expert aware of all the intricacies but to give you the tools necessary to study problems you are interested in on a deeper level than Chem3D, Dreiding models, etc. allow. Remember that all models are wrong but some models are useful.

6.1 Diels-Alder Stereoselectivity

Consider the following Diels-Alder reactions and experimentally observed *endo* / *exo* ratios:



1 : R_1 = H, R_2 = H, *no endo / exo selectivity*
2 : R_1 = CHO, R_2 = H, *endo: 80, exo: 20*
3 : R_1 = CN, R_2 = Me, *endo: 12, exo: 88*



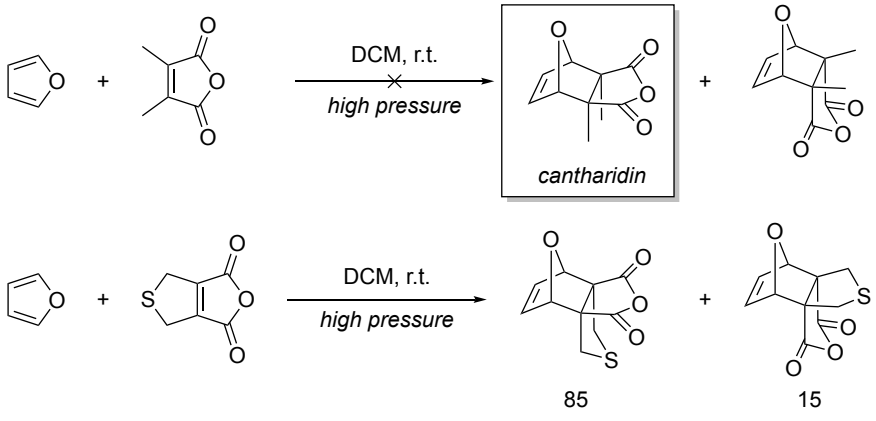
4 cyclopentadiene + maleic anhydride, *endo: 80, exo: 20*

1. Using `icctools`, model the simplest base case **1**. Do you observe the expected reaction profile?
2. Model the remaining three reactions for both *endo* and *exo* products (submit calculations in parallel on Euler and Grace to save time). Is the qualitative influence of the substituents on ΔG and $\Delta G^{\ddagger}$ as you were expecting from your undergraduate education?
3. How does the relative reaction barrier differ? Can you explain the experimentally observed product distribution? Assume a strictly kinetically controlled reaction and use the equation $\ln(\frac{k_A}{k_B}) = -\frac{\Delta\Delta G^{\ddagger}}{RT}$.
4. If you have time, try different methods (e.g., `librxn.Method.XTB` and `librxn.Method.BP86`) on the same reaction. How does the calculated reaction profile change?

Hint: Use a different working folder for every calculation.

6.2 Diels-Alder in Total Synthesis

Dauben tried to access the natural product cantharidin from furan and 3,4-dimethylfuran-2,5-dione. Unfortunately, even high pressures and temperatures did not lead to the desired product. However, furan reacted with 4,6-dihydro-1H,3H-thieno[3,4-c]furan-1,3-dione under high pressure at room temperature to the desired product, which could be converted to cantharidin in only one additional step.



access cantharidin in 1 step

1. Use `icctools` to model both reactions leading to *endo* and *exo* products, respectively.

2. Given the results of your calculations, do you hypothesize Dauben's problem was related to kinetics (high activation barrier) or thermodynamics (unfavorable ΔG)? Do you have a hypothesis why the introduction of an additional sulfur atom was helpful?

Reference: J. Am. Chem. Soc. **1980**, *102*, 6893-–6894