Folder structure

- All images from the doctoral thesis are summarized in the folder **Images_Work** . The pictures are sorted into subfolders of the corresponding chapters.
- All current simulation **data for** the python simulations are stored in the **Data** folder . The calculations of the individual levels or different ions are stored in the subfolder **LevelPop** . In the LevelPop folder, the whole thing splits up again into the **HCO** and **CD** folders . In addition to **LevelPop,** there is also a subfolder **LIICG** . The files with the results of the calculated LIICG signals are located in this folder.
- The **origin files** are compiled in the folder **Originfiles** . This folder is divided into subfolders for **CD +, C3H +, HCO +** and **Felion_Tests** .
- Folder **Python simulation** the two current versions of Python simulation (once for HCO + and one for C D +) down isol eichert. For more detailed descriptions of the simulations see below.

Data folder

The results of the simulations are always saved in .txt files. A run of the simulation always includes two txt files. Once for the simulation with light and once for the simulation without light. The assumed THz power is always noted in the file names, the factor a used (i.e. k3_1 = a * k3_0), the helium density used, the simulated storage time and the assumed temperature.

For example :

LevelPopulation_off_THzPower = 2e-05_a = 0.5_n_He = 1e + 16_TrapTime = 600_Temp = 5.7

LevelPopulation_on _THzPower = 2e-05_a = 0.5_n_He = 1e + 16_TrapTime = 600_Temp = 5.7

The first file contains the simulated data without light and the second file contains the data with light. Since both files belong to a run, the settings were identical. T

THz Power = 2e-5 W / a = 0.5 / n_He = 1e16 / TrapTime = 600ms / temperature = 5.7K

Since the files are quite large, it is best not to open them with the standard editor, but rather with NotePad ++, for example. The files are always structured identically. Each line corresponds to a time step in the simulation. The first column shows the CD + ions in the ground state (CD0), then those in the first excited rotation level (CD1), then those in the second (CD2), then the HeCD + clusters, then the He2CD + and in the last column the total number as a control that nothing is lost. In the python simulation there is a script with which you can plot these files.

In the subfolder **LIICG** there is also a corresponding file that belongs to the two files **above** . The file name is

LIICG_THzPower = 2e-05_a = 0.5_n_He = 1e + 16_TrapTime = 600_Temp = 5.7b

There are only two numbers in this file. In the first column the THz power in the simulation and in the second column the calculated signal after 600ms storage time under the specified conditions. You can of course argue excellently whether you really need this file ...

Python simulations

There are two simulation programs. One for CD + and one for HCO +. In principle, both programs work identically, only use other constants suitable for the corresponding ion.

In general, the programs are always divided into 6 Python files.

- Class_Definitions.py
- Main.py
- Kinetic_Simulation.py
- SemiAnalytic.py
- SemiAnlalytic2.py
- Plot.py

The following is a brief overview of how the individual files work. It is important for all files that all paths are static! In other words, if you run the simulation on another computer, you have to adapt the paths to the corresponding computer!

Class_Definitions.py

The different constants that are used in different program parts are defined here. It is important here in the class that **Ratecoefficients** the correct values are entered (for the rate coefficients for the formation and destruction of the Clusteri on s at a certain temperature) . All other values are only used for initialization and can therefore be ignored when using the simulation. It is certainly not cleverly done that you have to set the right values in one class and not in the other. This would surely be a point of improvement .

Main.py

Main.py is used to control the numerical simulation. You have several options here. These differ in how many different parameters are to be varied. Accordingly, there are many interlocking FOR loops to combine all values with all others. In the **simulation_NoVariation** variant , only one setting is simulated. There is also the option of plotting measurement data in the simulation (at **simulation_single_data**). For each type of simulation you have to define the THz Power, the TrapTIme, the temperature and possibly other values (or their range). This always happens before the FOR loops. The runtime of the simulation, especially if several parameters are varied in large areas, can take up to one hour. Each of the routines then calls the **Kinetic_Simulation.py** file . The actual simulation is processed in this with the transmitted values.

Kinetic_Simulation.py

Under **calculate_time_conditions** you can set the time step width of the simulation in ms (default value 5e-3). Under **set_ratecoefficients** you have to enter the rate coefficients for the formation and destruction of the cluster ions (at the corresponding temperature). In **calculate_einstein_coefficients** the Einstein coefficients are set or calculated (may need to here the crossover frequency and the Einstein A coefficient to be entered). **calculate_occupancy** calculates the Boltzmann occupation for the given levels at the given temperature. **calculate_rates** calculates the rates for the simulation. **kinetic_light_on** and **kinetic_light_off** perform the numerical calculations for the systems with and without light. **plot _figure** is no longer used at the moment . **simulation** reads in all data, initializes the arrays for data storage and calls the sub-functions explained above in the appropriate order and accordingly frequently to simulate the set storage time. In this routine, the files described above are generated with the corresponding populations of the individual ions.

SemAnalytic.py

Here you can enter a maximum measured cluster depletion signal for a certain temperature and transition frequency (LIICG_Meas is the variable in the code for it) and the program calculates a corresponding a. This part is based on the idea presented at the end of subchapter 5.3.1. is described in the work.

SemAnalytic 2 .py

This part of the program is based on the theoretically derived formula for the signal strength (also chapter 5.3.1.). Here, too, you have the option of varying various parameters (helium density, THz power (once with data, once without)) or using a fixed set to calculate the signal strength. According to the desired conditions, the values (or the value ranges) must be entered accordingly under define constants or in the various sub-functions. The results of the simulations are only saved as an image.

Plot.py

There are various (sometimes outdated routines) with which you can plot the results of the numerical simulation in Python. In the end I only used the routine **plot_LevelPop** to create the images for the work.