**Ordnerstruktur**

* Im Ordner **Bilder\_Arbeit** sind alle Bilder aus der Doktorarbeit zusammengefasst. Die Bilder sind in Unterordner der entsprechenden Kapitel einsortiert.
* Im Ordner **Data** sind alle aktuellen Simulationsdaten der Pythonsimulationen gespeichert. Im Unterordner **LevelPop** sind die Berechnungen der einzelnen Levels bzw. unterschiedlichen Ionen gespeichert. Im Ornder LevelPop spaltet sich das Ganze nochmals in die Ordner **HCO** und **CD** auf. Neben **LevelPop** gibt es noch einen Unterordner **LIICG**. In diesem Ordner befinden sich die Dateien mit den Ergebnissen der jeweils berechneten LIICG Signale.
* Im Ordner **Originfiles** sind die Originfiles zusammengestellt. Dieser Ordner ist in Unterordner aufgeteilt für **CD+, C3H+, HCO+** und **Felion\_Tests**.
* Im Ordner **PythonSimulation** sind die zwei aktuellen Versionen der Pythonsimulation (einmal für HCO+ und einmal für CD+) abgespeichert. Für genauere Beschreibungen der Simulationen siehe unten.

**Ordner Data**

Die Ergebnisse der Simulationen werden immer in .txt Dateien abgespreichert. Zu einem Durchlauf der Simulation gehören immer zwei txt-Files. Einmal für die Simulation mit Licht und einmal für die Simulation ohne Licht. In den Dateinamen ist immer die angenommene THz-Leistung vermerkt, der verwendete Faktor a (also k3\_1 = a\*k3\_0), die verwendete Heliumdichte, die simulierte Speicherzeit und die angenommene Temperatur.

Beispiel:

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

Das erste File beinhaltet die simulierten Daten ohne eingestrahltes Licht und das zweite File die Daten mit Licht. Da beide Datein zu einem Run gehören sind die Einstellungen identisch gewesen. T

THz Power = 2e-5 W / a = 0.5 / n\_He = 1e16 / TrapTime = 600ms / Temperatur = 5.7K

Da die Files recht groß sind sollte man sie am besten nicht mit dem Standardeditor öffnen, sondern besser mit NotePad++ z.B.. Die Files sind immer identisch aufgebaut. Jede Zeile entspricht einem Zeitschritt in der Simulation. In der ersten Spalte stehen die CD+ Ionen im Grundzustand (CD0), dann die im ersten angeregten Rotationslevel (CD1), dann die im zweiten (CD2), dann kommen die HeCD+ Cluster, dann die He2CD+ und in der letzten Spalte die Gesamtzahl als Kontrolle, dass nichts verloren geht. In der Pythonsimulation gibt es ein Skript mit dem man diese Files plotten kann.

Im Unterordner **LIICG** befinden sich auch ein entsprechendes File, dass zu den beiden oberen Files gehört. Der Dateiname lautet

LIICG\_THzPower=2e-05\_a=0.5\_n\_He=1e+16\_TrapTime=600\_Temp=5.7b

In diesem File stehen nur zwei Zahlen. In der ersten Spalte die THz Power in der Simulation und in der zweiten Spalte das berechnete Signal nach 600ms Speicherzeit unter den angegebenen Bedingungen. Man kann sich jetzt natürlich vortrefflich darüber streiten, ob man dieses File wirklich braucht...

**PythonSimulationen**

Es gibt zwei Simulationsprogramme. Eines für CD+ und eines für HCO+. Beide Programme arbeiten im Prinzip identisch, verwenden nur andere Konstnaten passend für das entsprechende Ion.

Allgemein sind die Programme immer unterteilt in 6 Python Files.

* Class\_Definitions.py
* Main.py
* Kinetic\_Simulation.py
* SemiAnalytic.py
* SemiAnlalytic2.py
* Plot.py

Im folgenden soll eine kurze Übersicht über die Funktionsweise der einzelnen Files gegeben warden. Wichtig ist bei allen Files, dass alle Pfade statisch sind! Sprich, wenn man die Simulation auf einem anderen Rechner laufen lässt, muss man die Pfade an den entsprechen Rechner anpassen!

**Class\_Definitions.py**

Hier werden die unterschiedlichen Konstanten definiert, die in verschiedenen Programmteilen verwendet werden. Wichtig ist hier, dass in der class **Ratecoefficients** die richtigen Werte eingetragen sind (für die Ratenkoeffizienten zur Bildung und Zerstörung des der Clusterionen bei einer bestimmten Temperatur). Alle Anderen Werte dienen hier nur der Initialisierung und können somit ignoriert werden bei Benutzung der Simulation. Das ist sicherlich nicht geschickt gemacht, dass man in einer Klasse die richtigen Werte setzen muss und in den anderen nicht. Dies wäre sicherlich ein Punkt der verbessert werden könnte.

**Main.py**

Main.py dient der Steuerung der numerischen Simulation. Man hat hier mehrere Optionen. Diese unterscheiden sich dadurch wieviele verschiedene Parameter variert werden sollen. Dementsprechend viele ineinandergreifende FOR Schleifen existieren dann um alle Werte mit allen anderen zu kombinieren. In der Variante **simulation\_NoVariation** wird nur eine Einstellung simuliert. Außerdem gibt es noch die Möglichkeit Messdaten mit in die Simulation zu plotten (bei **simulation\_single\_data**). Für jede Art der Simulation muss man die THz Power, die TrapTIme, die Temperatur und eventuell weitere Werte (oder deren Bereich) definieren. Dies geschieht immer vor den FOR Schleifen. Die Laufzeit der Simulation kann, insbesondere wenn mehrere Parameter in großen Bereichen variiert werden, durchaus eine Stunden in Anspruch nehmen. Jede der Routinen rufe dann die Datei **Kinetic\_Simulation.py** auf. In dieser wird mit den übermittelten Werten die eigentliche Simulation angewickelt.

**Kinetic\_Simulation.py**

Unter **calculate\_time\_conditions** kann man die Timestepweite der Simulation in ms einstellen (Standardwert 5e-3). Unter **set\_ratecoefficients** muss man die Ratenkoeffizienten zur Bildung und Zerstörung der Clusterionen eintragen (bei der entsprechenden Temperatur). In **calculate\_einstein\_coefficients** werden die Einsteinkoeffizienten gesetzt bzw. berechnet (eventuell muss hier die Übergangsfrequenz und der Einstein A Koeffizient eingegeben werden). **calculate\_occupancy** berechnet die Boltzmannbesetzung für die gegebenen Levels bei der gegebenen Temperatur aus. **calculate\_rates** berechnet die Raten für die Simulation. **kinetic\_light\_on** und **kinetic\_light\_off** führt die numerischen Berechnungen für die Systeme mit und ohne Licht durch. **plot\_figure** wird im Moment nicht mehr verwendet. **simulation** liest alle Daten ein, initialiesiert die Arrays für die Datenspeicherung und ruft die oben erläuterten Subfunktionen in der entsprechenden Reihenfolge und entsprechend häufig auf um die eingestellte Speicherzeit zu simulieren. In diesere Routine werden auch die oben beschriebenen Files mit den entsprechenden Besetzungen der einzelnen Ionen erzeugt.

**SemAnalytic.py**

Hier kann man man für eine bestimmte Temperatur und Übergangsfrequenz ein maximales gemessenes Clusterdepletion Signal eingeben (LIICG\_Meas heißt die Variable im Code dafür) und das Programm berechnet ein entsprechendes a. Dieser Teil basiert auf der Idee, die am Ende des Unterkapitels 5.3.1. in der Arbeit beschrieben ist.

**SemAnalytic2.py**

Dieser Teil des Programmes basiert auf der in der Arbeit theoretisch hergeleiteten Formel für die Signalstärke (ebenfalls Kapitel 5.3.1.). Auch hier hat man die Möglichkeit verschiedene Parameter zu variieren (Helium Dichte, THz Power (einmal mit Daten plotten einmal ohne)) oder mit einem festen Satz die Signalstärke auszurechnen. Entsprechend der gewünschten Bedingungen muss man unter define constants bzw. in den verschiedenen Unterfunktionen die Werte (oder die Wertebereiche) entsprechend eintragen. Die Ergebnisse der Simulationen werden jeweils nur als Bild gespreichert.

**Plot.py**

Hier finden sich verschiedene (teilweise überholte Routinenen) mit denen man die Ergebnisse der numerischen Simulation in Python plotten kann. Ich habe am Ende nur noch die Routine **plot\_LevelPop** verwendet um die Bilder für die Arbeit zu erstellen.