Code Documentation

This document is created to give a general overview of the code for the means of reviewing and tracking progress as well as for training purposes.

The GitHub repository contains more files that could be useful, but this document would only talk about the "quant_rotor" library. The files in a discarded folder are either completed projects or discarded ideas; files that are in the main fouler are usually files with the work in progress.

1 Library File Structure and Breakdown

This section will provide a general overview for the library structure as well as a small note on every file and a reference to more material about those files.

The library is broken down into three main folders:

- "core" contains the main files that are to be run or imported. They have the most high-level functions.
- "data" was designed to hold any input of output of the code it is not in use at the moment.
- "models" holds all supporting files and code which are used by "core".

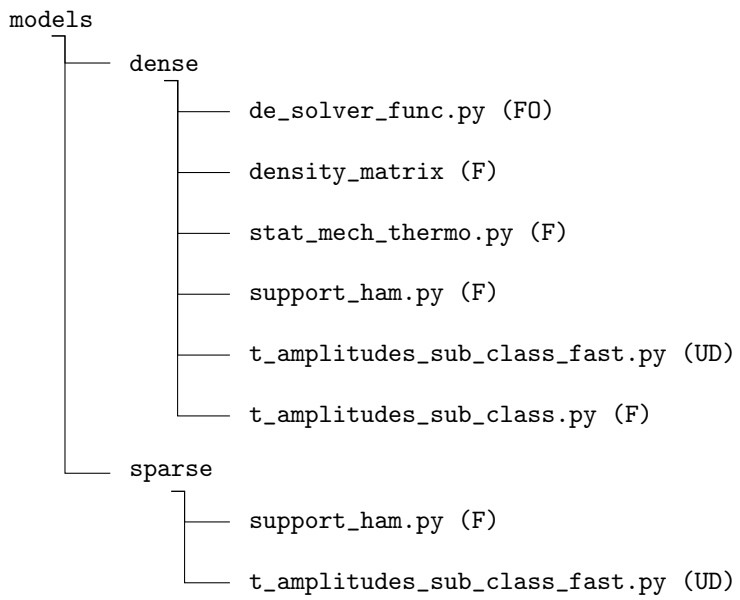
```
quant_rotor
    ├─ core
    ├─ data
    └─ models
```

Looking more specifically at the files They are marked with letters in parentheses as such:

UD - for under development
F - finished file
FO - finished files gotten from external sources

Additionally "core" and "models" are separated on two sub-folders which generally contain the same files but optimized for sparse or dense matrix handling.(consult Wikipedia for sparse matrices and sparse in python Scipy)

```
core
├── dense
│   ├── de_solve_one_thermal_dense.py (UD)
│   ├── de_solve_one_thermal.py (F)
│   ├── de_solve.py (FO)
│   ├── hamiltonian_big.py (F)
│   ├── hamiltonian.py (F)
│   ├── t_amplitudes_periodic_fast.py (UD)
│   ├── t_amplitudes_periodic.py (F)
│   └── t_amplitudes_guess.py (F)
└── sparse
    ├── de_solve_one_thermal_sparse.py (UD)
    ├── hamiltonian_big.py (F)
    ├── hamiltonian.py (F)
    └── t_amplitudes_periodic_fast.py (UD)
```



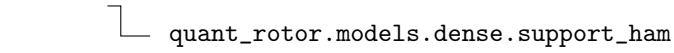
2 File descriptions

This section provides a short description of relevant files and their function structure from "core" with referencing to all files used from "models"; reference to any useful documentation on the topic.

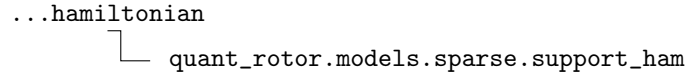
2.1 Classic approach of constructing a Hamiltonian: "hamiltonian" (dense), & "hamiltonian" (sparse)

Classical approach to constructing a Hamiltonian. (consult Configurational Coupled Cluster document Section 1 and 2)
The files are updated and optimized versions of each other in the order of newer to older: "hamiltonian" (sparse) -> "hamiltonian" (dense).

Structure of dense hamiltonian:



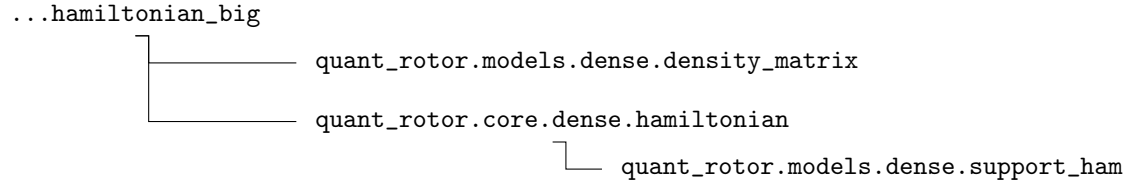
Structure of sparse hamiltonian:



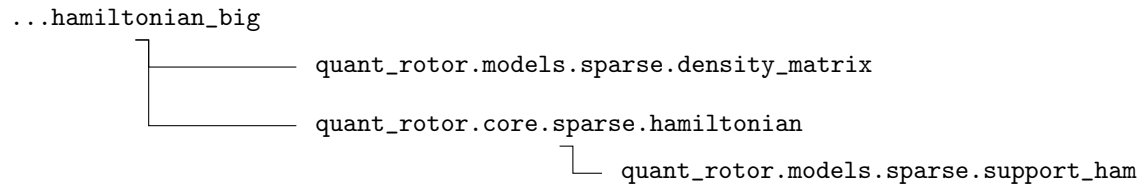
2.2 Reduced density matrix from ED approach: "hamiltonian_big.py" (dense) & "hamiltonian_big.py" (sparse)

Approach to constructing a hamiltonian by approximation through reduced density matrices. (consult Configurational Coupled Cluster document Section 1 and 2)
The files are updated and optimized versions of each other in the order of newer to older: "hamiltonian_big" (sparse) -> "hamiltonian_big" (dense).

Structure of dense hamiltonian_big:



Structure of sparse hamiltonian_big:

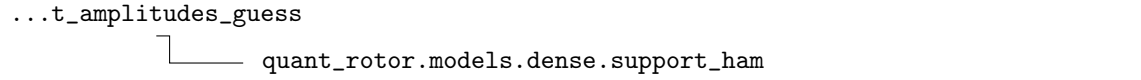


2.3 Iterative procedure to solve for residuals: "t_amplitudes_periodic_fast" (dense and sparse) & "t_amplitudes_periodic" & "t_amplitudes_guess"

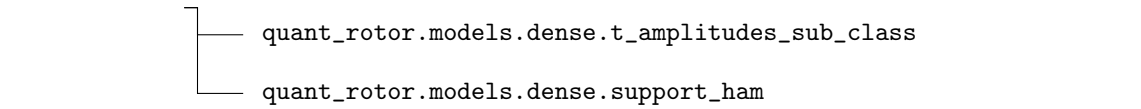
This files provides iterative approach to CCC.(consult Configurational Coupled Cluster document Sections 3, 5, 6 and 7) The files are the updated and optimized versions of each other in the order of newer to older: "t_amplitudes_periodic_fast" (dense) -> "t_amplitudes_periodic_fast" (dense) -> "t_amplitudes_periodic".

The "t_amplitudes_guess" file is used to make a prediction which than later be given to the iterative solver. Was introduce in attempt to medigate the problem of bit g. (consult Configurational Coupled Cluster document Section 8)

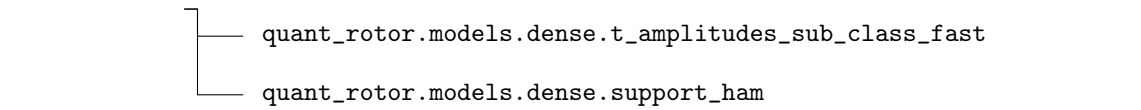
Structure of t_amplitudes_guess:



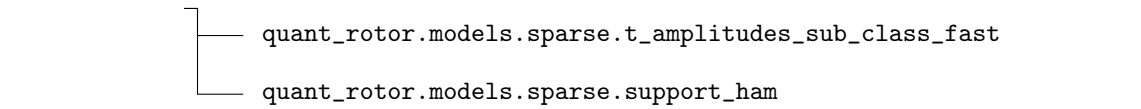
Structure of "t_amplitudes_periodic:



Structure of dense "t_amplitudes_periodic_fast:



Structure of sparse "t_amplitudes_periodic_fast:



2.4 Time-dependent CCC approach: "de_solve_one_thermal_dense" & "de_solve_one_thermal" & "de_solve" & de_solve_one_thermal_sparse

This files provides a Runge Kutta method for solving a differential equation to find the first and second residuals by time probagation.(consult Configurational Coubled Cluster document Section 10) The files are updated and optimised versions of each other in the order of newer to older: "de_solve_one_thermal_sparse" -> "de_solve_one_thermal_dense" -> "de_solve_one_thermal" -> "de_solve".

Structure of de_solve_one_thermal & de_solve:

```
...de_solve_one_thermal_dense & de_solve
                        ├─ quant_rotor.models.dense.de_solver_func
                        ├─ quant_rotor.models.dense.t_amplitudes_sub_class
                        └─ quant_rotor.models.dense.support_ham
```

Structure of de_solve_one_thermal_dense:

```
...de_solve_one_thermal_dense
                  ├─ quant_rotor.models.dense.de_solver_func
                  ├─ quant_rotor.models.dense.t_amplitudes_sub_class_fast
                  └─ quant_rotor.models.sparse.support_ham
```

Structure of de_solve_one_thermal_sparse:

```
...de_solve_one_thermal_sparse
                  ├─ quant_rotor.models.dense.de_solver_func
                  ├─ quant_rotor.models.sparse.t_amplitudes_sub_class_fast
                  └─ quant_rotor.models.sparse.support_ham
```

3 Script examples

This section will provide examples of calling functions, description of the output, simple testing procedures and a high level descriptions. Any files referenced in this section will be located in "script_examples". (if you use jupiter there will be a jupiter compiled file with all of the scripts combined and ready to run)

Listing 1: Example script

```
# Python script block
print("Hello, world!")
for i in range(5):
    print(i)
```