Contents

Code Documentation **1**

Library File Structure and Breakdown **1**

1 "Models" files description **2**
1.1 Supporting functions for constructing the hamiltonian: "support_ham.py (dense)" 2
1.1.1 File and Function structure 3
1.1.2 Testing procedures 3
1.2 Density matrix calculation: "density_matrix.py" (dense) 4
1.2.1 File and Function structure 4
1.2.2 Testing procedures 4
1.3 Defining dense operator for Coupled Cluster: "t_amplitudes_sub_class.py" & "t_amplitudes_sub_class_fast.py" (dense and sparse) 5
1.3.1 File and Function structure 5
1.3.2 Testing procedures 6

2 "Core" files description **6**
2.1 Classic approach of constructing a Hamiltonian: "hamiltonian" (dense), & "hamiltonian" (sparse) 6
2.1.1 File and Function structure 6
2.1.2 Testing procedures 7
2.2 Natural Orbitals Approximation: "hamiltonian_big.py" (dense) & "hamiltonian_big.py" (sparse) 8
2.2.1 File and Function structure 8
2.2.2 Testing procedures 8
2.3 Iterative procedure to solve for residuals: "t_amplitudes_periodic_fast" (dense and sparse) & "t_amplitudes_periodic" & "t_amplitudes_guess" 9
2.3.1 File and Function structure 9
2.3.2 Testing procedures 10
2.4 Time-dependent CCC approach: "de_solve_one_thermal_dense" & "de_solve_one_thermal" & "de_solve" & de_solve_one_thermal_sparse 11
2.4.1 File and Function structure 11
2.4.2 Testing procedures 11

Code Documentation

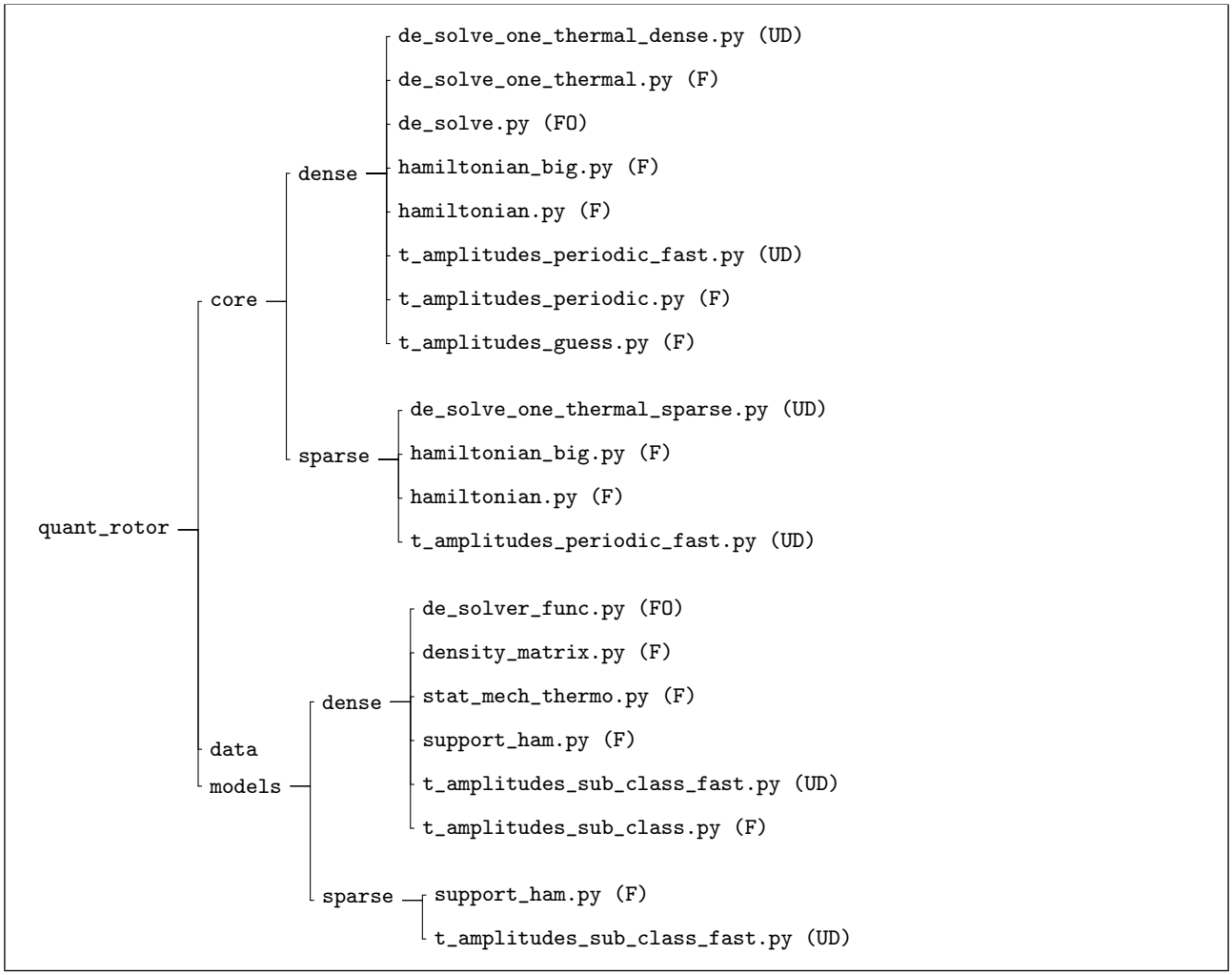
This document is created to give a general overview of the code for the means of reviewing and tracking progress as well as for training purposes.
Important! Read the introductions to every section very carefully because the trees represent different things in different sections.
The GitHub repository 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.

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".



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)

File descriptions

This section provides a short description of relevant files and their interactions.
Additionally, this section will present the description of the functions and function structure as well as notes on the ways to test them.
Take note of "..." they are usually hyperlinks to the omitted parts.

1 "Models" files description

This section describes files in "models".

1.1 Supporting functions for constructing the hamiltonian: "support_ham.py (dense)"

Approach to constructing one site and two sites operators and potential and kinetic hamiltonian operators. (consult Configurational Coupled Cluster document Sections 1 & 2)

1.1.1 File and Function structure

The first tree shows the file structure and right after the associated function structure. The files which are on the same level with functions mean that you can find those function in the associated files.

File structure of dense support_ham:

Doesn't use any imported files.

Function structure of dense support_ham:

```
write_matrix_elements ─┬ free_one_body
                       └ interaction_two_body_coplanar

basis_m_to_p_matrix_conversion ─┐ create_inverse_index_map ─┐ m_to_p

H_kinetic

H_potential
```

File structure of sparse support_ham:

Doesn't use any imported files.

Function structure of sparse support_ham:

```
build_V_prime_in_p ─┐ vector_in_p ─┐ m_to_p

build_V_in_p ─┐ vector_in_p ─┐ m_to_p

H_kinetic_sparse

H_potential_sparse
```

1.1.2 Testing procedures

This section will talk about how to test each of the functions. Ideally for the simplicity and speed one would test the functions with the previous iteration of the function but in absence of that option there are tests that can be done to ensure the correctness.

Dense support_ham:

---

write_matrix_elements: —fill—
—fill—

---

basis_m_to_p_matrix_conversion: —fill—
—fill—

H_kinetic: —fill—
—fill—

H_potential: —fill—
—fill—

Sparse support_ham:

---

build_V_prime_in_p: —fill—
—fill—

build_V_in_p: —fill—
—fill—

H_kinetic_sparse: —fill—
—fill—

H_potential_sparse: —fill—
—fill—

1.2 Density matrix calculation: "density_matrix.py" (dense)

Approach to constructing one site and two sites reduced density matrices (RDM). (consult Configurational Coupled Cluster document Section 9)

1.2.1 File and Function structure

The first tree shows the file structure and right after the associated function structure. The files which are on the same level with functions mean that you can find those function in the associated files.

File structure of dense density_matrix:

```
...density_matrix ─┐ support_ham
```

Function tructure of dense density_matrix:

```
density_matrix_1
density_matrix_2
                  ┌ write_matrix_elements ─┐ ...
dencity_energy ───┼ basis_m_to_p_matrix_conversion ─┐ ...
                  ├ density_matrix_1
                  └ density_matrix_1
```

1.2.2 Testing procedures

This section will talk about how to test each of the functions. Ideally for the simplicity and speed one would test the functions with the previous iteration of the function but in absence of that option there are tests that can be done to ensure the correctness.

Dense density_matrix:

---

write_matrix_elements: ...

---

basis_m_to_p_matrix_conversion: ...

---

density_matrix_1: —fill—
—fill—

---

density_matrix_2: —fill—
—fill—

---

dencity_energy: —fill—
—fill—

1.3 Defining dense operator for Coupled Cluster: "t_amplitudes_sub_class.py" & "t_amplitudes_sub_class_fast.py" (dense and sparse)

These files construct CCC operators with are used in the iterative scheme.(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_sub_class_fast" (sparse) -> "t_amplitudes_periodic_fast" (dense) -> "t_amplitudes_sub_class".
The file and function structure as well as the ways to test them are universal between files t_amplitudes_sub_class.py & t_amplitudes_sub_class_fast.py (dense and sparse).

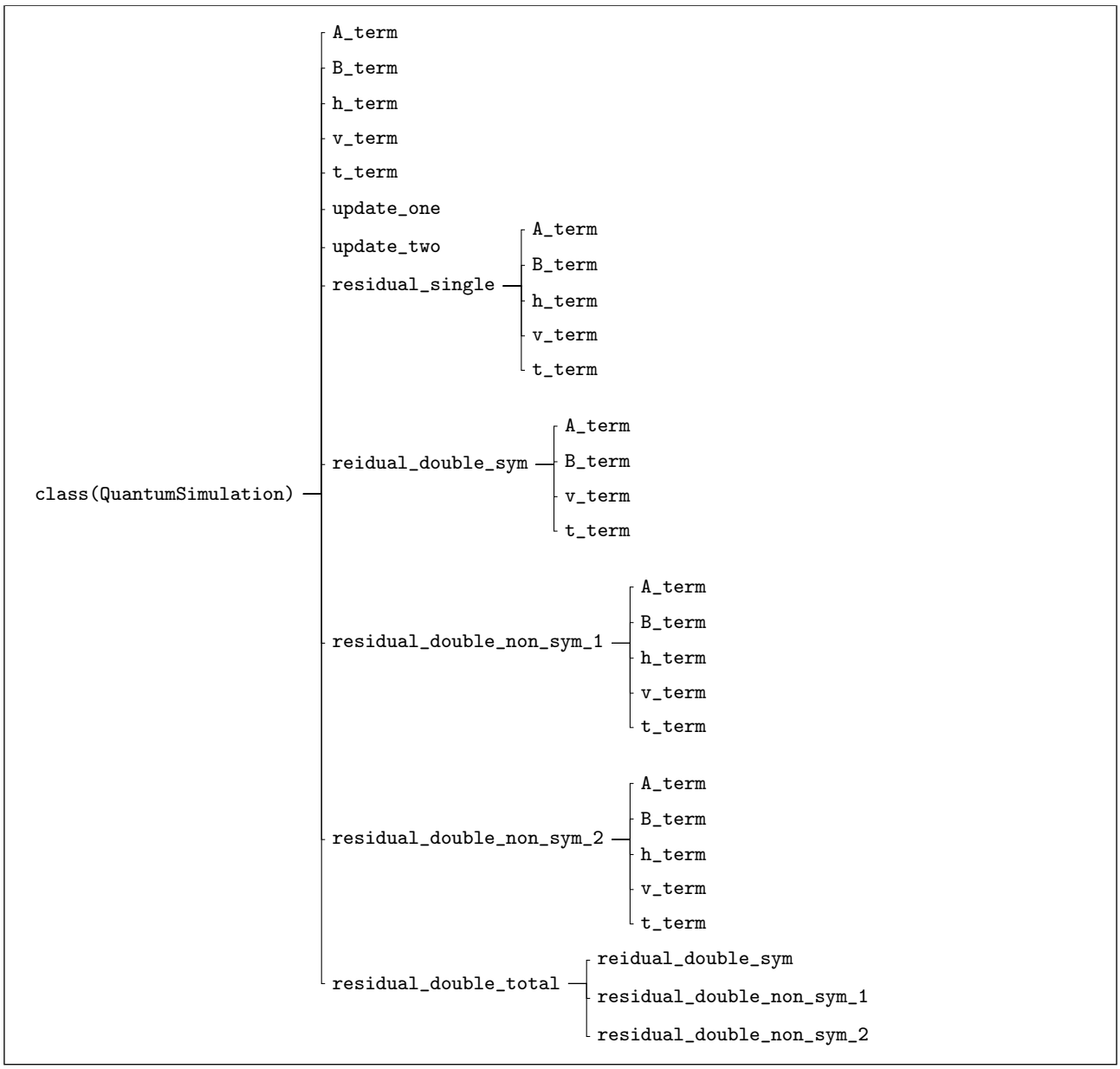
1.3.1 File and Function structure

The first tree shows the file structure and right after the associated function structure. The files which are on the same level with functions mean that you can find those function in the associated files.

File structure of dense t_amplitudes_sub_class.py & t_amplitudes_sub_class_fast.py (dense and sparse):

Doesn't use any imported files.

Function tructure of dense t_amplitudes_sub_class.py & t_amplitudes_sub_class_fast.py (dense and sparse):



1.3.2 Testing procedures

This section will talk about how to test each of the functions. Ideally for the simplicity and speed one would test the functions with the previous iteration of the function but in absence of that option there are tests that can be done to ensure the correctness.

Dense t_amplitudes_sub_class.py & t_amplitudes_sub_class_fast.py (dense and sparse):

A_term: —fill—
—fill—

B_term: —fill—
—fill—

---

h_term: —fill—
—fill—

---

v_term: —fill—
—fill—

---

t_term: —fill—
—fill—

---

update_one: —fill—
—fill—

---

update_two: —fill—
—fill—

---

residual_single: —fill—
—fill—

---

reidual_double_sym: —fill—
—fill—

---

residual_double_non_sym_1: —fill—
—fill—

---

residual_double_non_sym_2: —fill—
—fill—

---

2 "Core" files description

This section describes files in "core". Take note of "..." they are usually hyperlinks to the omitted parts.

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).

2.1.1 File and Function structure

The for the files the first tree shows the file structure and right after the associated function structure.
The files which are on the same level with functions mean that you can find those function in the associated files.

File tructure of dense hamiltonian:

```
...hamiltonian ─┐ quant_rotor.models.dense.support_ham
```

Function tructure of dense hamiltonian:

```
                    ┌ write_matrix_elements
                    ├ basis_m_to_p_matrix_conversion
hamiltonian_dense ──┤
                    ├ H_kinetic
                    └ H_potential
```

File structure of sparse hamiltonian:

```
...hamiltonian ─┐ quant_rotor.models.sparse.support_ham
```

Function structure of sparse hamiltonian:

```
                     ┌ build_V_in_p
hamiltonian_sparse ──┤
                     ├ H_kinetic_sparse
                     └ H_potential_sparse
```

2.1.2 Testing procedures

This section will talk about how to test each of the functions. Ideally for the simplicity and speed one would test the functions with the previous iteration of the function but in absence of that option there are tests that can be done to ensure the correctness.

Dense hamiltonian:

---

write_matrix_elements: ...

---

basis_m_to_p_matrix_conversion: ...

H_kinetic: ...

H_potential: ...

---

hamiltonian_dense: —fill—
—fill—

Sparse hamiltonian:

---

build_V_in_p: ...

H_kinetic_sparse: ...

H_potential_sparse: ...

---

hamiltonian_sparse: —fill—
—fill—

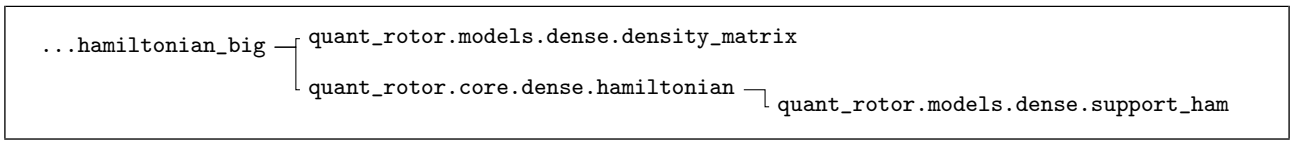
2.2 Natural Orbitals Approximation: "hamiltonian_big.py" (dense) & "hamiltonian_big.py" (sparse)

Approach to constructing a hamiltonian by approximation through reduced density matrices. (consult Configurational Coupled Cluster document Section 9)
The files are updated and optimized versions of each other in the order of newer to older: "hamiltonian_big" (sparse) -> "hamiltonian_big" (dense).

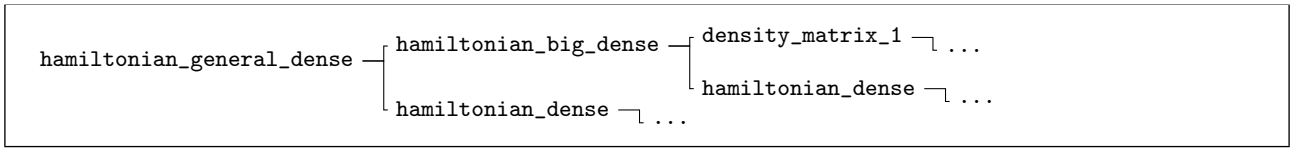
2.2.1 File and Function structure

The first tree shows the file structure and right after the associated function structure. The files which are on the same level with functions mean that you can find those function in the associated files.

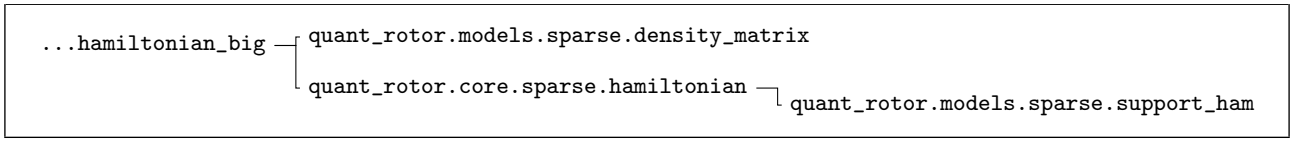
File structure of dense hamiltonian_big:



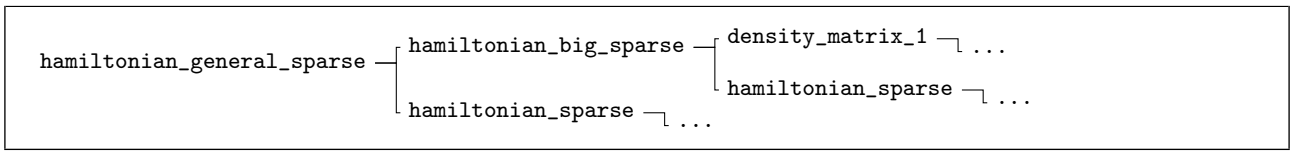
Function structure of dense hamiltonian_big:



File structure of sparse hamiltonian_big:



Function structure of sparse hamiltonian_big:



2.2.2 Testing procedures

This section will talk about how to test each of the functions. Ideally for the simplicity and speed one would test the functions with the previous iteration of the function but in absence of that option there are tests that can be done to ensure the correctness.

Dense hamiltonian_big:

---

hamiltonian_general_dense & hamiltonian_big_dense: —fill—
—fill—

---

hamiltonian_dense: ...

---

density_matrix_1: ...

Sparse hamiltonian_big:

---

hamiltonian_general_sparse & hamiltonian_big_sparse: —fill—

—fill—

---

hamiltonian_sparse: ...

---

density_matrix_1: ...

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

These files provide 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" (sparse) -> "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)

2.3.1 File and Function structure

The first tree shows the file structure and right after the associated function structure. The files which are on the same level with functions mean that you can find those function in the associated files.

File structure of t_amplitudes_guess:

```
...t_amplitudes_guess ─┐ quant_rotor.models.dense.support_ham
```

Function structure of t_amplitudes_guess:

```
intermediate_normalisation
t_1_amplitutde
t_2_amplitutde
                    ┌ write_matrix_elements
amplitute_energy ──┤ basis_m_to_p_matrix_conversion
                    ├ t_1_amplitutde
                    └ t_2_amplitutde
t_1_amplitude_guess_ground_state ─┐ intermediate_normalisation
                                  └ t_1_amplitutde
t_2_amplitude_guess_ground_state ─┐ intermediate_normalisation
                                  └ t_2_amplitutde
```

File structure of "t_amplitudes_periodic:

```
...t_amplitudes_periodic ─┐ quant_rotor.models.dense.t_amplitudes_sub_class
                          └ quant_rotor.models.dense.support_ham
```

Function structure of "t_amplitudes_periodic:

```
HF_test
```

```
t_periodic ─┬ write_matrix_elements
            ├ basis_m_to_p_matrix_conversion
            └ class(QuantumSimulation) ─┬ residual_single
                                        ├ residual_double_total
                                        ├ update_one
                                        ├ update_two
                                        ├ B_term
                                        ├ h_term
                                        ├ v_term
                                        └ t_term
```

File structure of dense & sparse "t_amplitudes_periodic_fast:

```
...t_amplitudes_periodic ─┬ quant_rotor.models.dense.t_amplitudes_sub_class_fast
                          └ quant_rotor.models.sprase.support_ham
```

Function structure of dense "t_amplitudes_periodic_fast:

```
t_periodic ─┬ build_V_in_p
            └ class(QuantumSimulation) ─┬ residual_single
                                        ├ residual_double_total
                                        ├ update_one
                                        ├ update_two
                                        ├ B_term
                                        ├ h_term
                                        ├ v_term
                                        └ t_term
```

2.3.2 Testing procedures

This section will talk about how to test each of the functions. Ideally for the simplicity and speed one would test the functions with the previous iteration of the function but in absence of that option there are tests that can be done to ensure the correctness.

Dense t_amplitudes_guess:

---

intermediate_normalisation: —fill—
—fill—

---

t_1_amplitutde: —fill—
—fill—

---

t_2_amplitutde: —fill—
—fill—

---

amplitute_energy: —fill—
—fill—

---

t_1_amplitude_guess_ground_state: —fill—
—fill—

---

t_2_amplitude_guess_ground_state: —fill—
—fill—

Dense t_amplitudes_periodic & Dense/Sparse t_amplitudes_periodic_fast:

---

write_matrix_elements: ...

---

basis_m_to_p_matrix_conversion: ...

---

build_V_in_p: ...

---

class(QuantumSimulation): ...

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".

2.4.1 File and Function structure

The first tree shows the file structure and right after the associated function structure. The files which are on the same level with functions mean that you can find those function in the associated files.

File tructure 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
```

File 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
```

File 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
```

2.4.2 Testing procedures

This section will talk about how to test each of the functions. Ideally for the simplicity and speed one would test the functions with the previous iteration of the function but in absence of that option there are tests that can be done to ensure the correctness.