NExt ApplicationS of Quantum Computing

D4.6: Automatic Subspace Generation (ASG) v1.0

Document Properties

Contract Number	951821
Contractual Deadline	M40 (31/12/2023)
Dissemination Level	Public
Nature	Other
Editors	Venkatesh Kannan, ICHEC
Authors	Bruno Chagas, ICHEC Goar Sánchez-Sanz, ICHEC Pablo Lauret Martínez de Rituerto, ICHEC Venkatesh Kannan, ICHEC
Reviewers	Jan Reiner, HQS Alexis-Julien Bouquet-Gais, Atos (Eviden)
Date	07/12/2023
Keywords	quantum chemistry, molecular fragmentation
Status	Draft
Release	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 951821

History of Changes

Release	Date	Author, Organisation	Description of Changes
0.1	20/11/2023	Venkatesh Kannan	Template initiated
0.2	20/11/2023	Pablo Lauret Martínez de Rituerto, Venkatesh Kannan	Submitted for First Review
0.3	01/12/2023	Pablo Lauret Martínez de Rituerto, Venkatesh Kannan	Submitted for Second Review
1.0	07/12/2022	Pablo Lauret Martínez de Rituerto, Venkatesh Kannan	Submitted to PMT for EC portal

Table of Contents

1. Executive Summary

The NEASQC project aims at demonstrating and advancing the capabilities of NISQ-era devices through the development of practically-relevant use-cases.

In WP4 to develop chemistry use-cases, Task 4.1 is to develop a tool to fragment the structure of a given molecular system into subsystems that retain the chemical meaning. To address this, the Automatic Subspace Generation (ASG) is a new tool designed to fragment the structure of the system into smaller subsystems. The motivation is that the smaller subsystems could fit into a quantum computer compared to the larger molecular system to perform energy calculations, and can be explored by the users of the AGS tool.

This deliverable D4.6 titled “Automatic Subspace Generation (ASG) v1.0” is a short report accompanying the ASG open source software called `KraChem` that has been produced and is available in the NEASQC GitHub repository under the project `qfrag` (Chagas et al., 2023). The deliverable outlines the key functions of `KraChem` along with pointers to the repository of its implementation and a quick start user guide.

2. Overview of KraChem

`KraChem` (Chagas et al., 2023) is a Python library designed for working with chemical molecules, fragmentation, based on chemical files input and mathematical tools from graph theory.

The `KraChem` package has two main functionalities:

- Intermolecular fragmentation: Separation of two or more independent interacting moieties (typically two molecules). This find the collection of disconnected molecules in a given chemical compound.
- Intramolecular fragmentation: Separation of two or more moieties which belong to the same molecule, chain or system. This breaks a molecule into smaller parts, each of which is a collection of subset of atoms, in a given molecule defined by its atoms and connections.

Although the fragmentation of a single molecule is a priori simple, the number of possible fragments increases exponentially with the number of atoms and is unviable for large systems. The AGS tool systematically explores, analyses and creates inter- and intra-molecular fragments keeping the chemical meaning, and provide with a full fragmentation scheme.

To achieve this, the tool reads the input molecular structure and identifies which type (inter- or intra-molecular) of fragmentation is to be done. This is done by estimating the intermolecular distances between two fragments using van der Waals radii (Bondi radii). Once the fragmentation type is identified, the intramolecular fragmentation will be done exploring each atom and its connection with its neighbors using covalent radii. This provides the connectivity between atoms within one fragment (molecule), which in a subsequent step will be used to propose fragmentation schemes and levels of fragmentations. This ensures retaining the chemical meaning in the fragmented subsystems.

The `KraChem` library is based on an approach of analysing chemical molecules as graphs and arrays, and is developed using two main underlying packages:

- *numpy*: a Python library adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- *networkx*: a Python library for studying graphs and networks, which is used to encode molecules as graphs, and graph routines are defined/used to manipulate the molecules.

Moreover, routines for plotting graphs and certain analysis are implemented using packages for creating subsets and manipulating paths and directories. These include:

- *matplotlib*: a plotting library for the python programming language and its numerical mathematics extension numpy, where we have some features for plotting figures
- *itertools*: this module implements a number of iterator building blocks inspired by constructs from APL, Haskell, and SML. We use this package for combining parts of a chemical compound or atoms in a molecule into a collection of subsets
- *pathlib*: this module offers classes representing filesystem paths with semantics appropriate for different operating systems, where we can access and manipulate files in folders

2.1. Molecule Class in KraChem

The `KraChem` library is implemented using Object-Oriented Programming (OOP), composed of the main class called *Molecule* and its methods and attributes. Figure 1 depicts a concise structure of the features implemented in this package.

The root of this library structure contains the *Molecule Class* where a molecule is initialised by inputting the file path. An optional parameter is the chemical molecule name, in order to determine the name of the result folders. A category of *Molecular Geometry* is available to extract all the atoms, connections and bonding types (single, double or triple). A *Molecule to Graph* functionality is available to encode a given molecule as a graph.

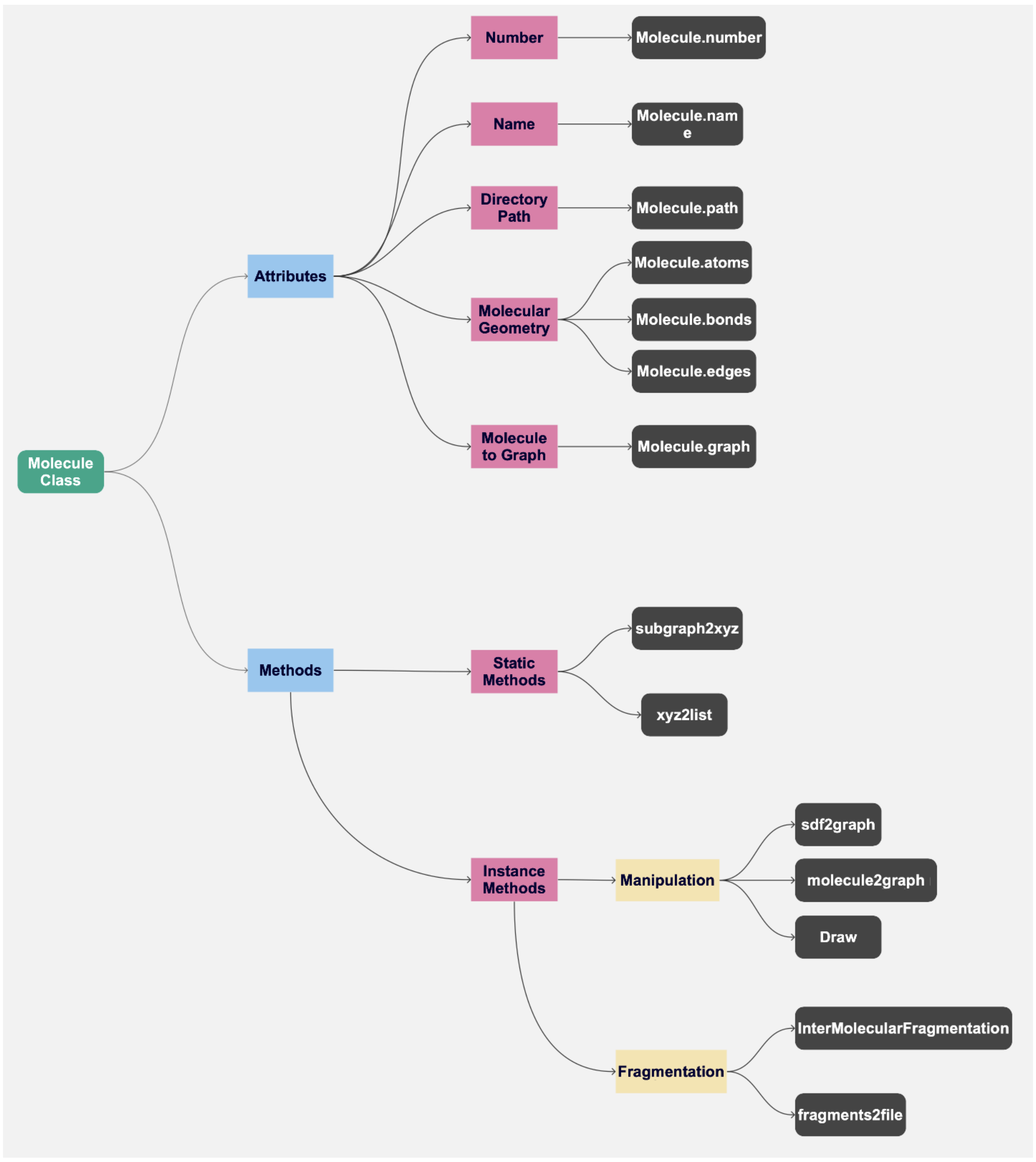

Figure 1*: `KraChem` package structure*

Under the first branch of the structure, all `Attributes` which can be associated with an initialised molecule are available. These include:

- **Number**: The attribute *Molecule.number* to initialise several molecules and keep tracking by ordering them.
- **Name**: The attribute *Molecule.name* to track and name files and folders after fragmenting them.
- **Directory Folder**: The attribute *Molecule.path* stores the path of a chemical molecule or compound to easily access sub-folders in other routines.
- **Molecular Geometry**: There are three attributes available for the molecule as a graph. They are:
  - *Molecule.atoms*: an array that contains a sequence of atoms as strings

- *Molecule.bonds*: an array that stores the bonding type of each connection between atoms (single, double, triple)
- *Molecule.edges*: an array of connections between atoms, considering atoms as nodes, written as tuples

• **Molecule to Graph**: The attribute *Molecule.graph* provides an object created from the package `networkx` which has the structure of a graph on which to perform analysis and operations.

The second branch of the structure, the `Methods` is composed of `instance` and `static` types of methods for manipulation and fragmentation routines.

- **Static Methods**:
  - *subgraph2xyz*: This is a routine which takes a `subgraph` as input, which can be a fragment from our molecule. The result is extraction of the `xyz` information over this fragment.
  - *xyz2list*: This routine converts an `xyz` file representation into a Python list.
- **Instance Methods**:
  - *sdf2graph*: This roune extracts the information from an *sdf* file into atoms, bonds and connections.
  - *molecule2graph*: Given the information of atoms, bonds and connections of a molecule, this routine can convert it into a *networkx* object graph, which may be used for initialization.
  - *Molecule.Draw()*: The *Molecule* object has all the information of the molecule graph, and can use this function to plot the graph.
  - *InterMolecularFragmentation*: This function may be used to extract all non-connected components of a graph – intermolecular fragments – and has the attribute *Molecule.components*.
  - *fragments2file*: Given the components of a molecule, this routine can be used to write them as files. A file is created is created for each fragment and all of their combinations.

2.2. Accessing and using KraChem

The `KraChem` library is available on the NEASQS GitHub as a repository at https://github.com/NEASQC/qfrag/tree/master (Chagas et al., 2023).

A user guide is available in the form of a Jupyter notebook at https://github.com/NEASQC/qfrag/blob/master/krachem_user_guide.ipynb (Chagas et al., 2023).

A set of input molecules that can be used as examples are provided at https://github.com/NEASQC/qfrag/tree/master/input_molecules (Chagas et al., 2023).

Bibliography

Chagas, B., Sánchez-Sanz, G., de Rituerto, P. L. M., & Kannan, V. (2023). NEASQC D4.6 Automatic Subspace Generation (AGS) v1.0. https://github.com/NEASQC/qfrag/tree/master