

---

# **PYQCTools Documentation**

***Release 1.0***

**Enrico Ronca**

July 17, 2016



CONTENTS

<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	Tools for Omega Space Green's Functions . . . . .	3
1.2	Tools for Real-Time Green's Functions . . . . .	3
1.3	Tools for Integrals Dumping . . . . .	4



PYQCTools is a collection of python scripts useful to dump quantum chemistry integrals and to perform data post-processing for different quantum chemistry methods.

PYQCTools requires the following prerequisites to work:

- Python 2.6, 2.7, 3.2, 3.3, 3.4
- Numpy 1.6.2 or higher
- Scipy 0.10 or higher (0.12.0 or higher for python 3.3, 3.4)
- [PySCF](#) for Integrals Calculation in Integrals Dumpings scripts.



## CONTENTS

## 1.1 Tools for Omega Space Green's Functions

**gf\_trace.py:** Calculate the Density of States (DOS) value from an  $\omega$ -dependent Green's Function. It makes the trace of the Green's Function associated with a certain frequency value.

**Example:**

```
from PYQCTools.Omega_GF import gf_trace
```

```
gf_trace.run(green.txt, omega_value)
```

`green.txt`: formatted text file containing the Green's function, `omega_value`: double frequency value.

## 1.2 Tools for Real-Time Green's Functions

**rtgf.py:** Calculate the Density of States (DOS) values during a time propagation. It makes the trace of time-dependent Green's Functions calculated along a time propagation and return the DOS values as a function of time both for the real and for the imaginary part of the Green's Function.

**Example:**

```
from PYQCTools.RT_GF import rtgf
```

```
rtgf.run(prop_time, time_step, scratch)
```

`prop_time`: double value of the full propagation time (period), `time_step`: double value of the time-step, `scratch`: directory containing text files of the real (`green.$t.$t.txt`) and imaginary (`green.30000+$t.30000+$t.txt`) Green's Functions, where `$t` indicate the specific time-step.

The script save two output files, `rt_real.txt` and `rt_imag.txt`, containing the real and imaginary part of the time-dependent DOS respectively.

**fft.py:** Perform the fourier transform of the time-dependent Density of States (DOS). It reads the `rt_real.txt` and `rt_imag.txt` generated by the `rtgf.py` script and produces the `ldos.out` and `real_part.txt` files containing the imaginary and real parts of the *omega*-dependent DOS respectively.

**Example:**

```
from PYQCTools.RT_GF import fft
```

```
fft.run(broad, rem_add)
```

`broad`: double value of the imaginary broadening, `rem_add`: string specifying if we are working with the addition or removal part of the Green's Function. It can assume only the values 'add' or 'rem'.

**rt1rdm\_builder.py: Build the Real and Imaginary parts of a Real-Time 1RDM at every time step starting from DMRG data**

It opportunely combines the 1RDM components read from files `onepdm.$t.$t.txt` where `$t` is of the order of 1,2,3,..., 100001,10002,100003,... and 200001,200002,200003,... for the Real-Real, Imag-Real and Imag-Imag respectively.

**Example:**

```
from PYQCTools.RT_GF import rt1rdm_builder

rt1rdm_builder.run(prop_time, time_step, scratch)
```

`prop_time`: double value of the full propagation time (period), `time_step`: double value of the time-step, `scratch`: directory containing text files of the 1RDM components at every time-step.

**extrapolation.py: Perform linear prediction to extend the total propagation time of a time propagation.**

It reads `N` points of time-dependent DOS inside the files `rt_real.txt` and `rt_imag.txt` and use the last `N/2` data to predict the following `N` points.

**Example:**

```
from PYQCTools.RT_GF import extrapolation

extrapolation.run(full_range)
```

`full_range`: boolean variable. If it is true it returns the full range of calculated and predicted values, if it is false it returns only the predicted values.

The script produces `new_full_data.out` files if `full_range = true` otherwise it produces `predicted.out` output files.

**iter\_extrapolation.py: Perform an iterative linear prediction to extend the total propagation time of a time propagation**

It reads 4 points of the time-dependent DOS inside the files `rt_real.txt` and `rt_imag.txt` and use the last 2 of them to predict the following `N` points.

**Example:**

```
from PYQCTools.RT_GF import iter_extrapolation

iter_extrapolation.run(N)
```

`N`: integer variable specifying the total number of points that need to be predicted.

The script produces `new_full_real.out` and `new_full_imag.out` output files with the real and imaginary parts of the extended time-dependent DOS respectively.

## 1.3 Tools for Integrals Dumping

**Integrals\_dump.py: It dumps 1 and 2-electron integrals in the MO basis inside a CASCI space in FCIDUMP format.**

The PySCF input to calculate integrals is already included in the script.

**DipoleIntegrals\_dump.py: It dumps dipole integrals in the MO basis in a CASCI space in FCIDUMP format.**

The PySCF input to calculate integrals is already included in the script.



**LowdinOrtho\_Integrals.py:** It dumps 1 and 2-electron integrals in the Localized basis obtained by Lowdin Orthogonalization.

The PySCF input to calculate integrals is already included in the script.

hubbard\_1d: It dumps 1 and 2-electron integrals got the 1D Hubbard model.

**Example:**

```
from PYQCTools.Integrals_dump import hubbard_1d
```

```
hubbard_1d.run(nsites, t, U, output)
```

nsites: Number of sites, t: Hopping constant, U: Coupling constant, output: Output file name.

You can also download the [PDF version](#) of this manual.