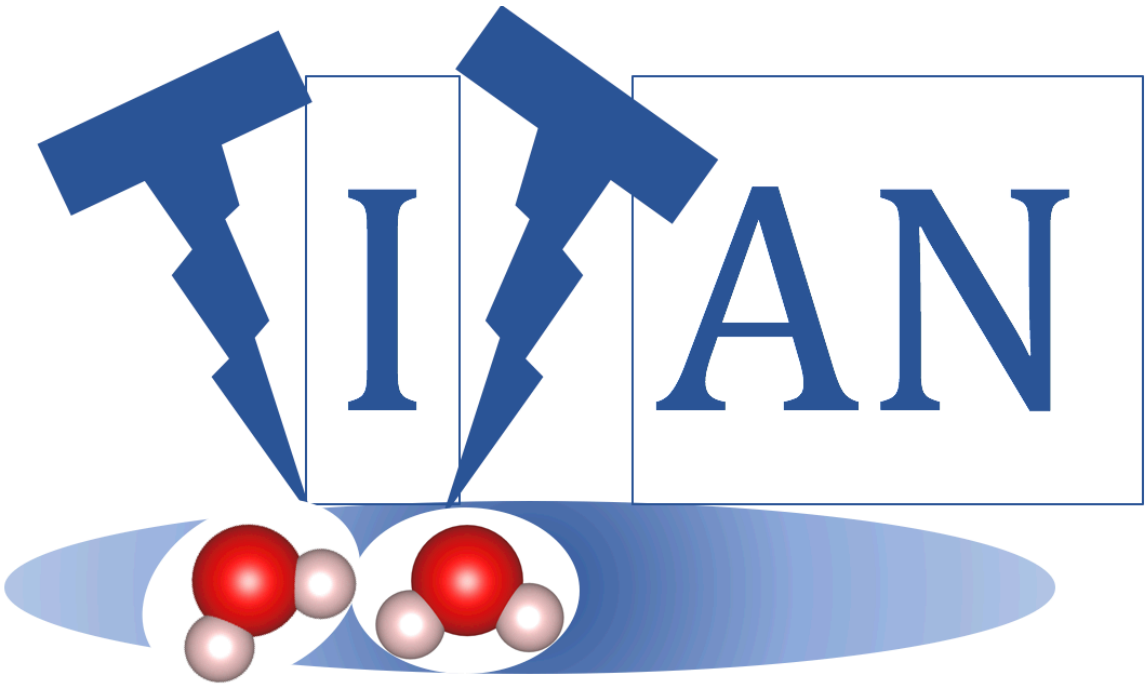

—

v2.0

A code for the generation and manipulation of electric fields

Authors:

Thijs Stuyver
Jing Huang
Dibyendu Mallick
Sason Shaik

8 – May – 2020

Table of Contents

1. Introduction

This section presents an overview of the TITAN package with all its features and acknowledges the funding and collaborators who have contributed to the code.

1.1. What is TITAN?

TITAN is an open-source software package capable of (a) generating external electric fields by creating collections of point-charges around chemical systems, and (b) quantifying electric fields at any location based on a charge distribution, e.g. the local electric fields (LEF) experienced at the active site of a (bio)chemical system, caused by the charged functional groups in distant regions of this system.

More specifically, two types of point charge distributions can be generated. On the one hand, it is possible to create two circular plate charge distributions on opposite sides of the active site of the system under consideration to generate a uniform oriented electric field. On the other hand, one can create a chiral charge distribution in the shape of spiral line with mixed positive and negative charges.

The quantification of electric fields can either start from an existing charge distribution, or alternatively, one can provide the structure of a chemical system (e.g. a protein), which is subsequently transformed into a charge distribution.

Additionally, a number of tools are included in the TITAN code as well, capable of generating AMBER[1] and CHARMM[2] libraries with non-standard residues and of determining differential charge distributions.

TITAN is distributed under the conditions of the GPL License version 3 (GPLv3).

1.2. What are the features in the current version?

- Generate a uniform oriented electric field with the help of circular plate charge distributions; output in AMBER or CHARMM format. Alternatively, the output can also be generated in Cartesian coordinates (compatible with GAUSSIAN09[3] and TURBOMOLE[4]; *vide infra*).

- Generate a chiral (either left- or right-handed) charge distribution in the shape of a spiral line; output in Cartesian coordinates (compatible with GAUSSIAN09 and TURBOMOLE; *vide infra*).
- Calculate the strength of the local electric field present within a chemical system: as input, one can either start from a .txt. file, a .pdb-format (the residues in the .pdb-file need to be compatible with either the AMBER or CHARMM force-field; non-standard residues can be included in the library with the help of one of the tools included in the package), or one can start from a GAUSSIAN09 .log-format (currently only methods to extract charges obtained through natural population analysis (NBO) have been implemented).

1.3. Limitations

The TITAN code does not include any Graphical User Interface (GUI). However, many of the output-files generated can be visualized by freely available graphic software packages. As an example, the generated .pdb files with the charge distributions can be visualized by either AVOGADRO,[5] CHIMERA,[6] VMD,[7] etc. The code is written in the standard PYTHON language and should thus run smoothly on either LINUX, WINDOWS or MAC computers. TITAN – v2.0 is compatible with PYTHON3. An older version of the code compatible with PYTHON2 (TITAN v1.3) can be downloaded from the website (*vide infra*).

1.4. Acknowledgements, contact and citation

TITAN has been developed within the Shaik group at the Hebrew University in Jerusalem, Israel. The code builds on a collection of algorithms which had been developed throughout the years by a variety of former research associates of Prof. Shaik. The first code for generating uniform external electric field was written by K. B. Cho. Subsequently it was applied by W. Z. Lai, C. Hui, and K. B. Cho to P450cam. Later D. Usharani and C. Li made some improvements to these scripts. H. Hirao wrote a script to quantify local electric fields. These individual scripts were organized, rewritten and a number of new features were included, resulting in TITAN – v1. The main developers of this original version of the code were Thijs Stuyver and Jing Huang. Subsequently, the code was rewritten from scratch and streamlined by Thijs Stuyver according to the principles of object-oriented programming to improve the versatility of the code and the

user-friendliness. Support throughout the project has been provided by Dibyendu Mallick and Prof. Sason Shaik.

If you wish to make use of the TITAN code in your research project, please include the following citation in the resulting scientific publications:

Thijs Stuyver, Jing Huang, Dibyendu Mallick, Sason Shaik, *J. Comput. Chem.* **2020**, *41*, 74-82.

Questions, suggestions, bugs and more can be sent to either of the following contributors:

- Prof. Sason Shaik: Director, The Lise Meitner-Minerva Center for Computational Quantum Chemistry, Institute of Chemistry, The Hebrew University of Jerusalem, Givat Ram Campus, Jerusalem 91904, ISRAEL

  E-mail: sason.shaik@gmail.com
  FAX: +972-2-6584680
  Phone: +972-2-6585909
  http://yfaat.ch.huji.ac.il/sason.html

- Thijs Stuyver: Post-doctoral research associate within the Shaik group, Institute of Chemistry, The Hebrew University of Jerusalem, Givat Ram Campus, Jerusalem 91904, ISRAEL

  E-mail: Thijs.Stuyver@huji.ac.il

1.5. How to obtain the TITAN code?

The TITAN code is an open-source package distributed by the Shaik group from the Hebrew University of Jerusalem for free. The current version of the package can be downloaded from:

https://titan-code.github.io

2. Installation

To properly install TITAN, the pip package is needed. Pip is readily distributed together with PYTHON and should thus be present on your machine; if this is not the case, the information on how to download pip can be found at https://pip.pypa.io/en/stable/installing/.

A link to download the latest version of TITAN can be found online at https://titan-code.github.io. Once the code has been downloaded and unpacked, the package can be installed by entering the main folder (TITAN-2.0.x) through the terminal and typing:

pip install .

The successful installation of the code can be tested by entering the 'unittest' folder situated in the main folder and introducing the following command in the terminal:

python TITAN_unittest.py

As soon as the code has been installed successfully, the individual modules are importable in PYTHON, so that they can be incorporated into new .py-scripts or they can be used interactively within the PYTHON interpreter.

Alternatively, one can use the stand-alone scripts, which are located in the 'stand-alone scripts' folder situated in the main folder. The main such script, 'run_titan.py', makes use of input-files (templates can be found in the 'templates' folder; examples can be found in the 'examples' folder). 'run_titan.py' starts by importing all the modules needed from the titan package and can thus be copied and moved to any location on the machine without its functioning being impacted. To execute 'run_titan.py', the following command needs to be entered into the terminal:

*Python THE-PATH-TO -RUN_TITAN/*run_titan.py TITAN_INPUT_NAME.inp

Next to 'run_titan.py', also 'run_differential_charge_distribution.py' and 'compile_custom_library.py' are included in the 'stand-alone scripts' folder. 'run_differential_charge_distribution.py' can be run through the following command:

Python THE-PATH-TO-RUN_DIFFERENTIAL_CHARGE_DISTRIBUTION/run_differential_charge_distribution.py name_distribution1 name_distribution2 name_output_file

Note that the .txt extensions of the files containing the charge distributions should not be included in the respective arguments. 'compile_custom_library' is an interactive script, i.e. it will request its input from the user, and can simply be run through the following command:

Python THE-PATH-TO -COMPILE_CUSTOM_LIBRARY/compile_custom_library.py *library_type*

The "library_type" keyword can either be "AMBER" or "CHARMM" (*vide infra*). In order to use the 'new_library.json' file generated by this script instead of one of the existing library files in the package ('amber_library.json' or 'charmm_library.json'), the existing library file needs to be replaced by this new file in the 'titan' folder. The new_library.json file should subsequently be named exactly the same as the library file it is intended to replace. Do not forget to re-install the package once this is done, i.e. enter the command '*python TITAN_unittest.py*' in the terminal while being located in the main folder, to update the .json files stored in the TITAN-package in your version of PYTHON.

3. Overview of the code

The current version of the TITAN code can perform 3 main types of operations:

- Circular plate charge distribution generation (CPC)
- Spiral line charge distribution generation (SL)
- Quantification of an electric field (QUANT)

Additionally, the code is capable of constructing differential charge distributions and compiling custom AMBER and CHARMM libraries (*vide infra*). As already indicated, the code can either be executed with the help of stand-alone scripts, or the modules can be imported individually into new PYTHON-scripts and/or into an instance of the PYTHON interpreter. First, we will provide an overview of the input-based approach, making use of the stand-alone script 'run_titan.py'.

3.1. Run_titan.py – input-file based execution of the TITAN code

For each of the three main calculation types identified above, templates of the corresponding input file can be found in the 'templates'-directory of the package. Below, an overview of the different keywords for each of the types of operations is provided. Note that while each operation requires every keyword listed in their respective subsection to be specified (unless indicated otherwise), there is no inherent order required in their listing in the input-file.

3.1.1. CPC

INPUT:
A .inp-file (template-file = TITAN_CPC.inp)

OUTPUT:

- *"OUTFORMAT = CHARMM"*: NAME.info and NAME.pdb
- *"OUTFORMAT = AMBER"*: NAME.info, NAME.pdb, NAME.frcmod, NAME.lib and leap.in

- *"OUTFORMAT = GAUSSIAN"*: NAME.info and NAME.txt

All these files are combined in a new directory, made within the active directory, and is called *"CHARMM_FORMAT_CPC"*, *"AMBER_FORMAT_CPC"* or *"GAUSSIAN_FORMAT_CPC"* respectively.

KEYWORDS:

- *"TYPE"*: This keyword indicates the type of operation that is requested. For a CPC-operation, this keyword is set to *"CPC"*.
- *"R"*: This keyword indicates the increase in radius between each successive ring in the circular plate charge distribution (cf. Fig. 1). The unit of this keyword is either Angstrom or Bohr (cf. *"UNIT"*).
- *"N"*: This keyword indicates the total number of circular rings in the plate.
- *"DIS"*: This keyword indicates the distance between the center of the plate and the point of at which the electric field should be calculated. The unit of this keyword is either Angstrom or Bohr (cf. *"UNIT"*).
- *"NAME"*: This keyword indicates the name for the output files (*NAME.info*, *NAME.pdb* etc.).
- *"OUTFORMAT"*: This keyword indicates the format of the output
  - *Options: "CHARMM", "AMBER"* or *"GAUSSIAN"*
- *"POINT1_X"*: This keyword indicates the x-coordinate of the first point of the vector along which the electric field will be aligned (e.g. the Fe=O bond in the active site of a P450 enzyme). Furthermore, POINT1 is the point at which the electric field strength is set, *vide infra*.
- *"POINT1_Y"*: This keyword indicates the y-coordinate of the first point of the vector along which the electric field will be aligned.
- *"POINT1_Z"*: This keyword indicates the z-coordinate of the first point of the vector along which the electric field will be aligned.
- *"POINT2_X"*: This keyword indicates the x-coordinate of the second point of the vector along which the electric field will be aligned.
- *"POINT2_Y"*: This keyword indicates the y-coordinate of the second point of the vector along which the electric field will be aligned.

- *"POINT2_Z"*: This keyword indicates the z-coordinate of the second point of the vector along which the electric field will be aligned.
- *"FIELD_STRENGTH"*: This keyword indicates the desired field strength (in a. u.) at POINT1.
- *"UNIT"*: This keyword indicates the spatial unit to be used.
  - *Options: "ANS"* or *"BOHR"*

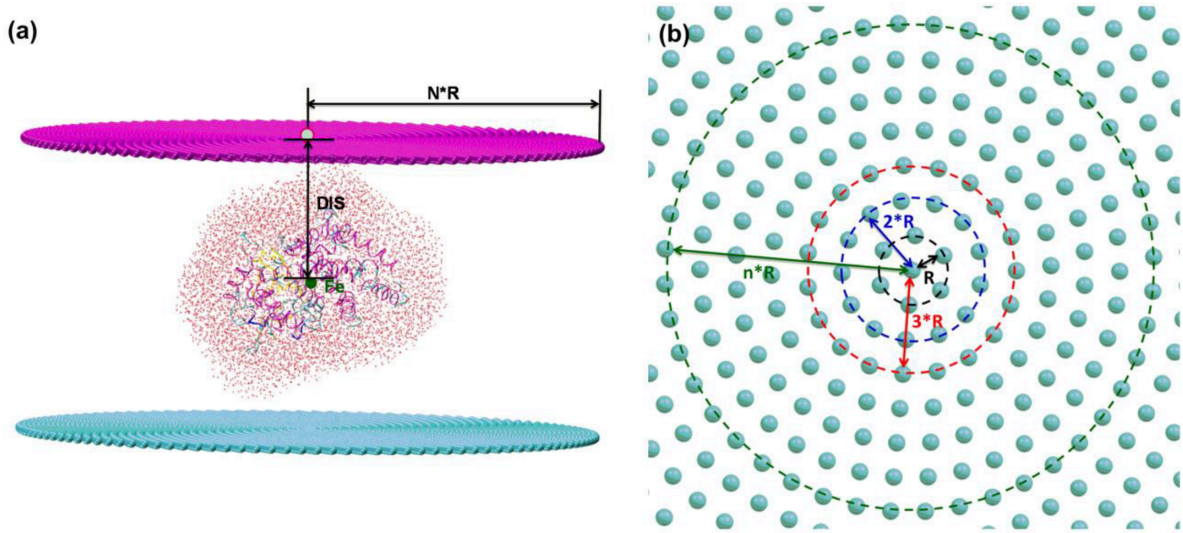

Figure 1. (a) The charges distribution generated in two circular plates with the keywords "R", "N" and "DIS"; (b) The relationship between "R", "N" and the position of the point charges in the circular plate.

The template input file for this kind of calculation, as it can be found in the 'templates'-directory, is shown below:

```
# TYPE OF OPERATION (CPC/SL/QUANT)
TYPE = CPC

# "R" IS THE GAP BETWEEN CIRCULAR RING (ANGSTROM)
R = 2.8

# "N" IS THE NUMBER OF CIRCULAR RINGS
N = 33

# THE DISTANCE BETWEEN THE CENTRE OF THE PLATE AND THE POINT OF CHOICE (ANGSTROM)
DIS = 46.8279
```

```
# THE NAME FOR THE OUTPUT FILES
# FOR EXAMPLE, IF WE SET THE “NAME” KEYWORD TO "chrg",
# IT MEANS THAT THE OUTPUT WILL BE chrg.info AND chrg.pdb
NAME = chrg

# THE FORMAT OF THE OUTPUT (“CHARMM”, “AMBER”, or “GAUSSIAN”)
OUTFORMAT = AMBER

# DEFINE THE COORDINATES OF THE VECTOR (POINT1 -> POINT2)
# WITH WHICH THE ELECTRIC FIELD HAS TO BE ALIGNED
# FOR EXAMPLE, THE Fe=O VECTOR IN THE ACTIVE SITE OF A P450 ENZYME

POINT1_X = 0.000
POINT1_Y = 0.000
POINT1_Z = 0.000

POINT2_X = 5.000
POINT2_Y = 9.000
POINT2_Z = 6.000

# THE FIELD STRENGTH DESIRED AT POINT1 (a.u.)
FIELD_STRENGTH = 0.0025

# THE SPATIAL UNIT (ANS OR BOHR)
UNIT = ANS
```

3.1.2. SL

INPUT:

A .inp-file (template-file = TITAN_SL.inp)

OUTPUT:

NAME.txt, NAME.pdb and NAME.info. All these files are combined in new directory, made within the active directory, and is called *"GAUSS_FORMAT_CPC"*.

KEYWORDS:

- *"TYPE"*: This keyword indicates the type of operation that is requested. For an SL-operation, this keyword is set to *"SL"*.
- *"RADIUS"*: This keyword indicates the increase in radius between each successive ring in the spiral-line distribution (cf. Fig. 2). The unit of this keyword is either Angstrom or Bohr (cf. *"UNIT"*).
- *"STEP"*: This keyword indicates the pitch between two neighboring atoms (cf. Fig. 2). The unit of this keyword is either Angstrom or Bohr (cf. *"UNIT"*).
- *"N"*: This keyword indicates the total numbers of atoms included in one full circle.
- *"NAME"*: This keyword indicates the name for the output files (*NAME.info, NAME.pdb* etc.).
- *"POINT_X"*: This keyword indicates the x-coordinate of the point at which the electric field will be calculated.
- *"POINT_Y"*: This keyword indicates the y-coordinate of the point at which the electric field will be calculated.
- *"POINT_Z"*: This keyword indicates the z-coordinate of the point at which the electric field will be calculated.
- *"CHIRALITY"*: This keyword indicates the chirality of the spiral line.
  - *Options: "RIGHT-HAND" or "LEFT-HAND"*
- *"NAME"*: This keyword indicates the name for the output files (*NAME.info*, *NAME.txt* and *NAME.pdb*).
- *"SEQUENCE"*: This keyword indicates the sequence of the charges in the spiral line. The sequence has to consist of a succession of *"P"* and *"N"* fragments (the number of *"P"* and *"N"* fragments should be equal in order to make sure that the net charge of the spiral line is zero).

- *"FATOM"*: This keyword indicates the number of dummy atoms included in a single *"P"* or *"N"* fragment.
- *"CHARGE"*: This keyword indicates the charge of each dummy atom.
- *"UNIT"*: This keyword indicates the spatial unit to be used (Angstrom or Bohr).
  - *Options: "ANS"* or *"BOHR"*

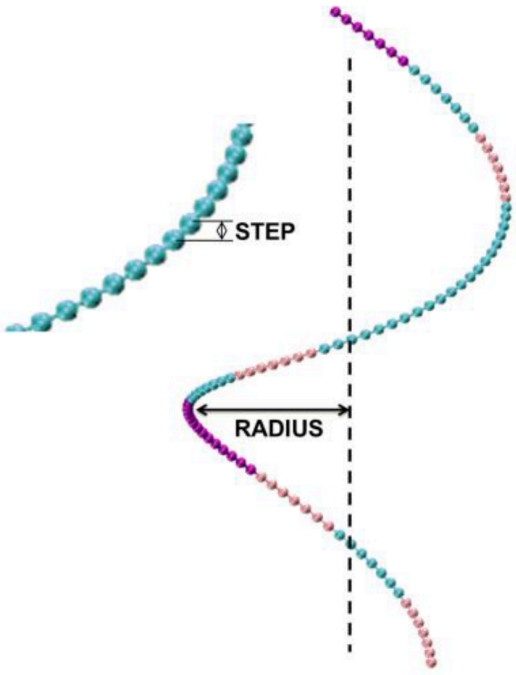

Figure 2. The point charges in the shape of a spiral line.

The template input file for this kind of calculation, as it can be found in the 'templates'-directory, is shown below:

```
# TYPE OF OPERATION (CPC/SL/QUANT)
TYPE = SL

# THE RADIUS OF SPIRAL LINE (ANGSTROM)

RADIUS = 10.0

# THE PITCH BETWEEN TWO NEIGHBORING ATOMS (ANGSTROM)

STEP = 0.01
```

```
# THE NUMBER OF ATOMS INCLUDE IN ONE CIRCLE

N = 700

# THE POSITION FOR THE ELECTRIC FIELD CALCULATION
# THE ELECTRIC FIELD WILL BE OBTAINED AT (POINT_X, POINT_Y, POINT_Z)

POINT_X = 0.00
POINT_Y = 0.00
POINT_Z = 0.00

# CHIRALITY OF THE SPIRAL LINE

#   CHIRALITY   |   NOTE
# ---------------------------------------
#  RIGHT-HAND   |  right-handed helices
#   LEFT-HAND   |  left-handed helices
# ---------------------------------------

CHIRALITY = RIGHT-HAND

# THE NAME FOR OUTPUT FILES
# FOR EXAMPLE, IF WE SET THE “NAME” KEYWORD TO "chrg",
# IT MEANS THE OUTPUT WOULD BE chrg.txt, chrg.info AND chrg.pdb

NAME = SL_test

# THE "SEQUENCE" IS DEFINED AS A SUCCESSION OF
# POSITIVE AND NEGATIVE CHARGE FRAGMENTS, WITH EACH FRAGMENT
# REPRESENTED BY "P" AND "N" RESPECTIVELY. THE NUMBER OF "P" AND "N"
# SHOULD BE EQUAL IN ORDER TO MAKE SURE THE NET CHARGE OF THE SPIRAL LINE IS
ZERO.
# "FATOM" CORRESPONDS TO THE NUMBER OF POINTS IN EACH FRAGMENT.
# "CHARGE" CORRESPONDS TO THE CHARGE OF EACH DUMMY ATOM.

FATOM = 10

SEQUENCE =
PPNNPPNPNPNPNNPPNNPNNPPNNPNPNPNPPPPPNPNNPNPNNPNNPNPPNNPPNPNPNPN
NPPNNPNNPPNNPNPNPNPPPPPNPNNPNPNNPNNPNPPNNPPNPNPNPNNPPNNPNNPPNN
PNPNPNPNPPPPPNPNNPNPNNPNNPNPPNNPPNPNPNPNNPPNNPNNPPNNPNPNPNPNPPPPP
NPNNPNPNNPNNPNPPNNPPNPNPNPNNPPNNPNNPPNNPNPNPNPNPPPPPNPNNPNPNNPNN
PNPPNNPPNPNPNPNNPPNNPNNPPNNPNPNPNPNPPPPPNPNNPNPNNPNNPNPPNNPPNPNPN
PNNPPNNPNNPPNNPNPNPNPNPPPPPNPNNPNPNNPNNPN
```

CHARGE = 0.1

```
# THE SPATIAL UNIT (ANS OR BOHR)
UNIT = ANS
```

3.1.3. QUANT

INPUT:

A .inp-file (template-file = TITAN_QUANTIFICATION_xxx.inp), together with either a .txt file (in case *"FILE = TXT"*), a .pdb file (in case *"FILE = PDB"*), or a GAUSSIAN .log file (in case *"FILE = LOG"*) as starting point for the electric field calculation, [1] a DIRECTION_FILE.txt file (in case *"DIRECTION = SELECT"*).

OUTPUT:

NAME.ef (contains the output of the electric field calculation), NAME.txt (in case *"FILE = PDB"* or *"FILE = LOG"*; this file contains the selected charge distribution extracted from the .pdb file, which was used for the electric field calculation), NAME.save (in case *"FILE = LOG"*, this file contains the list of the selected atoms with their coordinates and assigned charge).

KEYWORDS:

- *"TYPE"*: This keyword indicates the type of operation that is requested. For a QUANT-operation, this keyword is set to *"QUANT"*.
- *"FILE_TYPE"*: This keyword indicates the starting point of the electric field calculation, i.e. the type of input file.
  - *Options: "TXT", "PDB" or "LOG"*
- *"UNIT"*: This keyword indicates the spatial unit to be used.
  - *Options: "ANS"* or *"BOHR"*

[1] Note that in the MM region of a QM/MM calculation, i.e. the protein matrix of an enzyme, is generally modified close to the link atoms, which impacts the corresponding charge distribution. As such, in order to probe the actual role played by the electrostatics of a protein matrix on the active site in such calculations, it is advisable to quantify the electric field starting from this modified charge distribution so as to retain full consistency. ChemShell produces this modified point charge distribution in a format that is compatible with the charge distribution .txt-input of TITAN.

- *"NAME"*: This keyword indicates the name of the input file containing the starting point of the electric field calculation (either *NAME.txt*, *NAME.pdb* or *NAME.log*).
- *"CHARGE_SELECT"*: This keyword indicates whether all the point charges in the charge distribution are taken into account during the electric field quantification or not (in case *"CHARGE_SELECT = PART"*, the keyword *"CHARGE_SEQ"* has to be specified as well).
  - *Options: "ALL" or "PART"*
- *"CHARGE_SEQ"*: This keyword indicates which charges are taken into account during the electric field quantification. The sequence is defined as a combination of ranges, denoted by "*R(begin,end)"* (e.g. *"R(3,10)"*), and points, denoted by *"P(number)"* (e.g. *"P(300)"*). Consult the template below for a full example of such a sequence-string.
- *"DIRECTION"*: This keyword indicates whether the direction vector V, along which the oriented electric field is calculated, will be introduced manually or by selecting atoms in an additional input file.
  - *Options: "SELECT" or "MANUAL"*
- *"V1X"*: This keyword indicates the x-coordinate of the first point of the vector V along which the oriented electric field will be calculated (e.g. the Fe=O bond in the active site of a P450 enzyme). This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"V1Y"*: This keyword indicates the y-coordinate of the first point of the vector V along which the oriented electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"V1Z"*: This keyword indicates the z-coordinate of the first point of the vector V along which the oriented electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"V2X"*: This keyword indicates the x-coordinate of the first point of the vector V along which the oriented electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"V2Y"*: This keyword indicates the y-coordinate of the first point of the vector V along which the oriented electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"V2Z"*: This keyword indicates the z-coordinate of the first point of the vector V along which the oriented electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.

- *"POINT_X"*: This keyword indicates the x-coordinate of the point at which the electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"POINT_Y"*: This keyword indicates the y-coordinate of the point at which the electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"POINT_Z"*: This keyword indicates the z-coordinate of the point at which the electric field will be calculated. This keyword is only relevant in case *"DIRECTION = MANUAL"*.
- *"DIRECTION_FILE"*: This keyword indicates the coordinate .txt-file from which the parameters for the direction vector V and the location for the electric field evaluation will be read. This keyword is only relevant in case *"DIRECTION = SELECT"*.
- *"ATOM1"*: This keyword indicates the number of the first atom defining vector V: ATOM1(V1X, V1Y, V1Z). This keyword is only relevant in case *"DIRECTION = SELECT"*.
- *"ATOM2"*: This keyword indicates the number of the second atom defining vector V: ATOM2(V2X, V2Y, V2Z). This keyword is only relevant in case *"DIRECTION = SELECT"*.
- *"ATOM_CENTER"*: This keyword indicates the number of the atom defining the location at which the electric field will be calculated: ATOM_CENTER(POINT_X, POINT_Y, POINT_Z). This keyword is only relevant in case *"DIRECTION = SELECT"*.
- *"TYPE_OF_CHARGES"*: This keyword indicates the type of charges to be read in from the GAUSSIAN .log file. This keyword is only relevant in case *" FILE_TYPE = LOG"*.
  - *Options: "NBO"*
- *"FORCE_FIELD"*: This keyword indicates the force field of choice which will be used to generate the charge distribution from the .pdb-file. This keyword is only relevant in case *"FILE_TYPE = PDB"*.
  - *Options: "AMBER" or "CHARMM"*
- *"N_TERMINAL"*: This keyword indicates the number of the residue in the .pdb-file which corresponds to the N-terminal of the peptide. This keyword is only relevant in case *"FILE_TYPE = PDB"*.
- *"C_TERMINAL"*: This keyword indicates the number of the residue in the .pdb-file which corresponds to the C-terminal of the peptide. This keyword is only relevant in case *"FILE_TYPE = PDB"*.

- *"ASPP"*: This keyword indicates the numbers of the residues in the .pdb-file which corresponds to protonated ASP-residues. This keyword is only relevant in case *"FILE_TYPE = PDB"* and *"FORCE_FIELD = CHARMM"*.
- *"GLUP"*: This keyword indicates the numbers of the residues in the .pdb-file which corresponds to protonated GLU-residues. This keyword is only relevant in case *"FILE_TYPE = PDB"* and *"FORCE_FIELD = CHARMM"*.
- *"DISU"*: This keyword indicates the numbers of the residues in the .pdb-file which corresponds to CYS-residues involved in a disulfide bridge, i.e. unprotonated CYS-residues. This keyword is only relevant in case *"FILE_TYPE = PDB"* and *"FORCE_FIELD = CHARMM"*.

The complete template input file for this kind of calculation, as it can be found in the 'templates'-directory, is shown below:

```
# TYPE OF OPERATION (CPC/SL/QUANT)
TYPE = QUANT

# INPUT FILE TYPE: TXT FILE (TXT), PDB FILE (PDB), OR GAUSSIAN LOG FILE (LOG) AS
STARTING POINT
FILE_TYPE = PDB

# INPUT UNIT (BOHR OR ANS)
UNIT = ANS

# THE CHARGE DISTRIBUTION WILL BE READ FROM NAME.txt, NAME.pdb or NAME.log
# THE RESULT WILL BE WRITTEN IN NAME.ef

NAME = opt

#  CHARGE_SELECT    |                NOTE
#-------------------------------------------------------------------
#     "ALL"       | ALL THE POINT CHARGES IN THE CHARGE DISTRIBUTION
#                 | ARE SELECTED FOR THE ELECTRIC FIELD CALCULATION.
#-------------------------------------------------------------------
#     "PART"      | A PART OF CHARGES ARE SELECTED FROM THE CHARGE DISTRIBUTION.
#                 | (IN THIS CASE, THE "CHARGE_SEQ" KEYWORD NEEDS TO BE DEFINED)
#-------------------------------------------------------------------

CHARGE_SELECT = PART
```

```
# THE “CHARGE_SEQ" KEYWORD IS USED TO SELECT THE POINT CHARGES FOR
CHARGE_SELECT = "PART"
# IT IS NOT NECESSARY TO SET THE "CHARGE_SEQ" KEYWORD WHEN CHARGE_SELECT =
"ALL"
# FOR EXAMPLE, ”CHARGE_SEQ = R(3,10)+P(20)+P(3000)+R(400,403)+P(50)" MEANS:
# THE POINT CHARGES (PC) FROM NO. 3 TO NO. 10, THE PC NO. 20, THE PC NO. 3000, THE
PC FROM
# NO. 400 TO NO. 403 AND PC NO. 50 ARE SELECTED FOR THE EF CALCULATIONS.

CHARGE_SEQ = R(2,20)+P(35)

# THE DIRECTION VECTOR V CAN BE DEFINED IN TWO WAYS: EITHER BY SELECTING ATOMS
IN A COORDINATE FILE (SELECT),
# OR BY INTRODUCING THE COORDINATES MANUALLY (MANUAL)

DIRECTION = MANUAL

#_____
____
# THE FOLLOWING PARAMETERS ARE ONLY RELEVANT IN CASE DIRECTION = “MANUAL”

# DIRECTION VECTOR V
# V = (V2X-V1X, V2Y-V1Y, V2Z-V1Z)
# THE VALUE OF THE EF ALIGNED WITH THIS VECTOR IS CALCULATED.

V1X = 0.0000
V1Y = 0.0000
V1Z = 0.0000
V2X = 0.0000
V2Y = 0.0000
V2Z = 3.1010

# THE VALUE OF THE EF IS CALCULATED AT (POINT_X,POINT_Y,POINT_Z).

POINT_X = 0.0000
POINT_Y = 0.0000
POINT_Z = 0.0000

#_____
____
# THE FOLLOWING PARAMETERS ARE ONLY RELEVANT IN CASE DIRECTION = “SELECT”


DIRECTION_FILE = coords_ans

# DIRECTION VECTOR V
# V = (V2X-V1X, V2Y-V1Y, V2Z-V1Z)
```

# THE NUMBER OF THE ATOMS DETERMINING V: ATOM1(V1X, V1Y, V1Z) AND ATOM2(V2X, V2Y, V2Z)
# SELECTED FROM DIRECTION_FILE
ATOM1 = 26
ATOM2 = 25

THE VALUE OF THE EF IS CALCULATED AT (POINT_X,POINT_Y,POINT_Z).
THE NUMBER OF THE ATOM DETERMINING THIS POINT SELECTED FROM DIRECTION_FILE (ATOM_CENTER)

ATOM_CENTER = 26

_____
_____
THE FOLLOWING PARAMETERS ARE ONLY RELEVANT IN CASE FILE = “LOG”

# TYPE OF CHARGES TO BE READ FROM THE LOG FILE (NBO)
TYPE_OF_CHARGES = NBO

_____
_____
THE FOLLOWING PARAMETERS ARE ONLY RELEVANT IN CASE FILE = “PDB”

# FORCE FIELD OF CHOICE (AMBER/CHARMM)
FORCE_FIELD = AMBER

THE RESIDUE NUMBER OF N TERMINAL AND C TERMINAL

FOR EXAMPLE:
IF NO. 4 RESIDUE IS THE N-TERMINAL OF THE PEPTIDE, PLEASE SET " N_TERMINAL = 4 "
USE THE COMMAND: " grep "HT1" PDB_FILE   " TO CONFIRM THE RESIDUE NUMBER OF N-TERMINAL.
IF NO. 500 RESIDUE IS THE C-TERMINAL OF THE PEPTIDE, PLEASE SET " C_TERMINAL = 500 "
USE THE COMMAND: " grep "OT1" PDB_FILE   " TO CONFIRM THE RESIDUE NUMBER OF C-TERMINAL.

N_TERMINAL = 1
C_TERMINAL = 152

_____
_____
THE FOLLOWING PARAMETERS ARE ONLY RELEVANT IN CASE FILE = “PDB” & FORCE = “CHARMM”

THE RESIDUE NUMBER OF ASPP, GLUP AND DISU

```
# IF THE RESIDUES OF THE PROTONATED ASP ARE E.G. NO. 235, 246 RESIDUES, PLEASE SET " ASPP = 235,246 "
# USE THE COMMAND: "  grep "HD2 ASP" PDB_FILE    " TO CONFIRM THE RESIDUE NUMBER OF PROTONATED ASP.
# IF THE RESIDUES OF THE PROTONATED GLU ARE E.G. NO. 250, 266 RESIDUES, PLEASE SET " GLUP = 250,266 "
# USE THE COMMAND: "  grep "HE2 GLU" PDB_FILE    " TO CONFIRM THE RESIDUE NUMBER OF PROTONATED GLU.
# IF NO. 300 CYS RESIDUE IS BONDED WITH NO. 340 CYS RESIDUE THROUGH A DISULFIDE BOND,
# THEN NO. 300 AND NO. 340 CYS ARE NOT PROTONATED. IN THIS CASE, PLEASE SET " DISU = 300,340 "
# USE THE COMMAND: "  grep "SG  CYS" PDB_FILE    " AND "  grep "HG1 CYS" PDB_FILE    "
# TO CONFIRM THE RESIDUE NUMBER OF UNPROTONATED CYS.

ASPP =
GLUP =
DISU =
```

3.2. Constructing differential charge distributions and compiling libraries with the help of stand-alone scripts

As already indicated in Section 2, next to ‘run_titan.py’, also ‘run_differential_charge_distribution.py’ and ‘compile_custom_library.py’ are included in the ‘stand-alone scripts’ folder. Detailed instructions on how to run these scripts can be found in this preceding Section.

‘run_differential_charge_distribution.py’ is the script that can be used to generate a differential charge distribution, i.e. the difference between two existing charge distributions. It requires 2 charge distributions in .txt-format as input. The names of these input-files should be provided as arguments when calling the script, together with the name of the .txt output file to which the resulting differential distribution should be written (note that the .txt extensions of the arguments should not be included in the arguments). To calculate the electric field associated to the resulting differential distribution, one can perform a QUANT calculation with the help of ‘run_titan.py’ on the produced .txt file.

Custom libraries, containing non-standard residues, can be constructed with the help of ‘compile_custom_library.py’. This script can read .lib parameter files in AMBER style, as well

as .txt parameter files in CHARMM style. The standard parameter files for the respective force fields can be found in the 'libraries' folder in the main directory. In order to include non-standard residues, these parameter files can either be expanded by adding new entries, or a new .lib or .txt file can be constructed, adhering to the regular AMBER/CHARMM parameter file lay-out. Upon executing the 'compile_custom_library.py' script, the user will be asked to provide the names of the AMBER/CHARMM parameter files to be included in the new library. Before ending its operation, the script automatically writes a dictionary containing all the extracted parameters to the output-file 'new_library.json'. In order to use the generated 'new_library.json' file within the TITAN code, the existing library file ('amber_library.json' or 'charmm_library.json') in the 'titan' folder needs to be replaced by this new file. The 'new_library.json' file should subsequently be named exactly the same as the library file it is intended to replace. Do not forget to re-install the package once this is done so that the .json files stored in the TITAN-package in your version of PYTHON are updated.

3.3. Importing individual modules of TITAN in PYTHON

Next to running TITAN with the help of stand-alone input files, one can also import the TITAN package in new PYTHON scripts and/or the PYTHON interpreter. In its redesigned form, the TITAN code contains a separate class for each calculation type, i.e. CPC calculations correspond to the CircularGenerate class, SL calculations correspond to the SpiralLineGenerate class, and QUANT corresponds to the Quantification class. The Quantification class is an abstract class which has been instantiated into the QuantificationTxt, QuantificationPdbAmber, QuantificationPdbCharmm and QuantificationLog classes. Upon importing TITAN, i.e. executing '*import titan*', all of the calculation classes mentioned are directly accessible (the docstrings for the main calculation classes are provided in Subsection 3.3.1):

- CircularGenerate(self, point1_X, point1_Y, point1_Z, point2_X, point2_Y, point2_Z, R, N, distance, field_strength, name, unit="ANS")

- SpiralLineGenerate(self, sequence, fatom, charge, radius, step, N, chirality, point_X, point_Y, point_Z, name, unit="ANS")
- QuantificationTxt(self, name, point_x, point_y, point_z, v1_x, v1_y, v1_z, v2_x, v2_y, v2_z, charge_seq="/", charge_select="ALL", unit="ANS")
- QuantificationPdbAmber(self, name, point_x, point_y, point_z, v1_x, v1_y, v1_z, v2_x, v2_y, v2_z, n_terminal, c_terminal, charge_seq="/", charge_select="ALL", unit="ANS")
- QuantificationPdbCharmm(self, name, point_x, point_y, point_z, v1_x, v1_y, v1_z, v2_x, v2_y, v2_z, n_terminal, c_terminal, charge_seq="/", charge_select="ALL", unit="ANS", aspp=None, glup=None, disu=None)
- QuantificationLog(self, name, point_x, point_y, point_z, v1_x, v1_y, v1_z, v2_x, v2_y, v2_z, charge_seq="/", charge_select="ALL", unit="ANS")

Each of these classes has a main method which is responsible for the execution of the calculation, giving rise to the same output as obtained with the help of the stand-alone 'run_titan.py' script. For all of the QUANT classes, this method is 'self.execute()'; for the SL class, this method is 'self.create_spiral_line_distribution()'; for the CPC class, this method is 'self.create_and_write_plates()'. For practical examples on how to import the individual classes/modules, setup specific calculation objects and execute them, one can consider the 'TITAN_unittest.py' script in the unittest folder.

As a final side-note, the workhorse underlying all the calculation classes mentioned above is a single object type: the ChargeDistribution class. This ChargeDistribution contains a single attribute, the point_charge_list, which is a list containing the point-charges making up the charge distribution under the form [[x, y, z], charge]. A variety of methods are connected to this class; a complete overview can be obtained by typing 'help(ChargeDistribution)' in the

PYTHON interpreter after loading this class from the '_general_charge_distribution_class.py' module within the titan package. The most important ones are:

- ***Self.construct_point_charge_list****(all_charge_list, charge_seq, charge_select ="ALL")*: selects point-charges from an external all_charge_list to be included in the electric field calculation and stores them in the point_charge_list
- ***Self.calculate_oriented_electric_field****(point_x, point_y, point_z, vector_x, vector_y, vector_z, unit="ANS")*: calculates the electric field exerted by the point_charges in all_charge_list along (vector_x, vector_y, vector_z)-direction at (point_x, point_y, point_z)
- ***Self.calculate_electric_field****(point_x, point_y, point_z, unit="ANS")*: calculates the electric field exerted by the charge distribution in all_charge_list at position (point_x, point_y, point_z)
- ***Self.write_point_charge_list_to_txt_file****(name)*: writes the point_charge_list to the name.txt output-file

3.3.1. Docstrings for the main calculation classes

```
class CircularGenerate():
    """
    A class to represent a uniform field generation with the help of
circular plates

    Attributes:
    ----------
    point1_X -> point2_Z : float
        The coordinates of the axis along which the uniform field
    R : float
        The increase in radius between each successive ring in the circular
plate charge distribution (in A)
    N : int
        The total number of circular rings in the plate
    distance : float
        The distance between the center of the plate and the point at which
the electric field
        should be calculated (in A)
    field_strength : float
        The desired field strength (in au)
    name : str
        The name for the output files
```

```
    unit : str
        The unit used in the calculation (angstrom = "ANS", bohr = "BOHR";
default = "ANS")

    Methods:
    --------
    create_and_write_plates : creates the plates and writes them to the
respective input files
    """
```

```
class SpiralLineGenerate():
    """
    A class to represent a spiral line charge distribution generation

    Attributes:
    -----------
    name : string
        The name of the output files (name.info, name.pdb, etc.)
    sequence: string
        The sequence of the charges in the spiral line. The sequence has to
consist of a succession of "P"
        and "N" fragments (the number of "P" and "N" fragments should be
equal in order to make sure that
        the net charge of the spiral line is zero)
    fatom : int
        The number of dummy atoms included in a single "P" or "N" fragment
    charge : int
        The charge of each dummy atom
    radius : float
        The radius of the spiral line distribution (in Angstrom)
    step : float
        The pitch between two neighboring atoms (in Angstrom)
    N : int
        The total numbers of atoms included in one full circle.
    chirality : string
        The chirality of the spiral line (options: "RIGHT-HAND" or "LEFT-
HAND")
    point_X : float
        The x-coordinate of the point at which the electric field will be
calculated
    point_Y : float
        The y-coordinate of the point at which the electric field will be
calculated
    point_Z : float
        The z-coordinate of the point at which the electric field will be
calculated
    unit : str
        The unit used in the calculation (angstrom = "ANS", bohr = "BOHR";
default = "ANS")

    Methods:
    --------

    create_plates : creates the plates and writes them to the respective
input files
    """
```

```
class QuantificationTxt(Quantification):
    """
    A class that represents a quantification calculation starting from a
.txt file

    Attributes:
    ----------
    name : string
        The name of the input and output files
    point_x : float
        The x-coordinate of the point at which the electric field will be
quantified
    point_y : float
        The y-coordinate of the point at which the electric field will be
quantified
    point_z : float
        The z-coordinate of the point at which the electric field will be
quantified
    v1_x : float
        The x-coordinate of the first point making up the direction vector
    v1_y : float
        The y-coordinate of the first point making up the direction vector
    v1_z : float
        The z-coordinate of the first point making up the direction vector
    v2_x : float
        The x-coordinate of the second point making up the direction vector
    v2_y : float
        The y-coordinate of the second point making up the direction vector
    v2_z : float
        The z-coordinate of the second point making up the direction vector
    charge_seq : string
        The charge sequence, in which the range from charge/atom x to y is
denoted by R(x,y) and individual
        charges/atoms are denoted by P, i.e. P(x); "+" signs are used to
catenate ranges and points (default = "/")
    charge_select: string
        Keyword indicating whether all charges/atoms or only part are
considered during the quantification
        (2 options: "ALL" or "PART"; default = "ALL")
    unit : string
        The unit used in the calculation (angstrom = "ANS", bohr = "BOHR";
default = "ANS")

    Methods:
    -------
    execute : executes the workflow associated to the quantification
calculation
    """
```

```
class QuantificationPdbAmber(QuantificationPdb):
    """
    A class that represents a quantification calculation starting from a
.pdb file (amber force field)

    Attributes:
    ----------
    name : string
        The name of the input and output files
```

```
    point_x : float
        The x-coordinate of the point at which the electric field will be
quantified
    point_y : float
        The y-coordinate of the point at which the electric field will be
quantified
    point_z : float
        The z-coordinate of the point at which the electric field will be
quantified
    v1_x : float
        The x-coordinate of the first point making up the direction vector
    v1_y : float
        The y-coordinate of the first point making up the direction vector
    v1_z : float
        The z-coordinate of the first point making up the direction vector
    v2_x : float
        The x-coordinate of the second point making up the direction vector
    v2_y : float
        The y-coordinate of the second point making up the direction vector
    v2_z : float
        The z-coordinate of the second point making up the direction vector
    n_terminal
        The residue number of the protein that constitutes the n-terminal;
use command: "  grep "HT1" PDB_FILE    "
        to confirm the residue number
    c_terminal : int
        The residue number of the protein that constitutes the c-terminal;
use command: "  grep "OT1" PDB_FILE    "
        to confirm the residue number
    charge_seq : string
        The charge sequence, in which the range from charge/atom x to y is
denoted by R(x,y) and individual
        charges/atoms are denoted by P, i.e. P(x); "+" signs are used to
catenate ranges and points (default = "/")
    charge_select: string
        Keyword indicating whether all charges/atoms or only part are
considered during the quantification
        (2 options: "ALL" or "PART"; default = "ALL")
    unit : string
        The unit used in the calculation (angstrom = "ANS", bohr = "BOHR";
default = "ANS")

    Methods:
    -------
    execute : executes the workflow associated to the quantification
calculation
    """
```

```
class QuantificationPdbCharmm(QuantificationPdb):
    """
    A class that represents a quantification calculation starting from a
.pdb file (amber force field)

    Attributes:
    ----------
    name : string
        The name of the input and output files
    point_x : float
```

```
        The x-coordinate of the point at which the electric field will be
quantified
    point_y : float
        The y-coordinate of the point at which the electric field will be
quantified
    point_z : float
        The z-coordinate of the point at which the electric field will be
quantified
    v1_x : float
        The x-coordinate of the first point making up the direction vector
    v1_y : float
        The y-coordinate of the first point making up the direction vector
    v1_z : float
        The z-coordinate of the first point making up the direction vector
    v2_x : float
        The x-coordinate of the second point making up the direction vector
    v2_y : float
        The y-coordinate of the second point making up the direction vector
    v2_z : float
        The z-coordinate of the second point making up the direction vector
    n_terminal : int
        The residue number of the protein that constitutes the n-terminal;
use command: "  grep "HT1" PDB_FILE    "
        to confirm the residue number
    c_terminal : int
        The residue number of the protein that constitutes the c-terminal;
use command: "  grep "OT1" PDB_FILE    "
        to confirm the residue number
    charge_seq : string
        The charge sequence, in which the range from charge/atom x to y is
denoted by R(x,y) and individual
        charges/atoms are denoted by P, i.e. P(x); "+" signs are used to
catenate ranges and points (default = "/")
    charge_select: string
        Keyword indicating whether all charges/atoms or only part are
considered during the quantification
        (2 options: "ALL" or "PART"; default = "ALL")
    unit : string
        The unit used in the calculation (angstrom = "ANS", bohr = "BOHR";
default = "ANS")
    aspp : list
        The list of protonated aspartate residues (default = None); use
command: "  grep "HD2 ASP" PDB_FILE    "
        to confirm the residue numbers
    glup : list
        The list of protonated glutamate residues (default = None); use
command: "  grep "HE2 GLU" PDB_FILE    "
        to confirm the residue numbers
    disu : list
        The list of cysteine residues linked through a disulfide bond
(default = None); if e.g. CYS 300 is bonded with
        CYS 340 this way, then set this keyword to [300, 340]. Use
commands: "  grep "SG  CYS" PDB_FILE    " and
        "  grep "HG1 CYS" PDB_FILE    " to confirm the residue numbers

    Methods:
    _______
    execute : executes the workflow associated to the quantification
calculation
    """
```

```
class QuantificationLog(Quantification):
    """
    A class that represents a quantification calculation starting from a
.log file

    Attributes:
    ----------
    name : string
        The name of the input and output files
    point_x : float
        The x-coordinate of the point at which the electric field will be
quantified
    point_y : float
        The y-coordinate of the point at which the electric field will be
quantified
    point_z : float
        The z-coordinate of the point at which the electric field will be
quantified
    v1_x : float
        The x-coordinate of the first point making up the direction vector
    v1_y : float
        The y-coordinate of the first point making up the direction vector
    v1_z : float
        The z-coordinate of the first point making up the direction vector
    v2_x : float
        The x-coordinate of the second point making up the direction vector
    v2_y : float
        The y-coordinate of the second point making up the direction vector
    v2_z : float
        The z-coordinate of the second point making up the direction vector
    charge_seq : string
        The charge sequence, in which the range from charge/atom x to y is
denoted by R(x,y) and individual
        charges/atoms are denoted by P, i.e. P(x); "+" signs are used to
catenate ranges and points (default = "/")
    charge_select: string
        Keyword indicating whether all charges/atoms or only part are
considered during the quantification
        (2 options: "ALL" or "PART"; default = "ALL")
    unit : string
        The unit used in the calculation (angstrom = "ANS", bohr = "BOHR";
default = "ANS")

    Methods:
    -------
    execute : executes the workflow associated to the quantification
calculation
    """
```

4. References

[1] Case, D. A.; Cheatham, T. E.; Darden, T.; Gohlke, H.; Luo, R.; Merz, K. M. Jr.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. *J. Comput. Chem.* **2005**, *26*, 1668-1668.

[2] Brooks, B. R., Brooks III, C. L., Mackerell Jr, A. D., Nilsson, L., Petrella, R. J., Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S. et al. *J. Comput. Chem.* **2009**, *30,* 1545-1614.

[3] Frish, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Mennucci, B.; Petersson, G. A. et al. *Gaussian 09, revision D.01*; Gaussian, Inc.: Wallingford, CT, 2013.

[4] Ahlrichs, R.; Bär, M.; Häser, M.; Horn, H.; Kölmel, C. *Chem. Phys. Lett.* **1989**, *162*, 165-169.

[5] Hanwell, M. D.; Curtis, D. E.; Lonie, D. C.; Vandermeersch, T.; Zurek, E.; Hutchison, G. R. *J. Cheminformatics* **2012**, *4*, 17.

[6] Pettersen, E. F.: Goddard, T. D.; Huang, C. C.; Couch, G. S.; Greenblatt, D. M.; Meng, E. C.; Ferrin, T. E. *J. Comput. Chem.* **2004**, *25*, 1605-1612.

[7] Humphrey, W.; Dalke, A.; Schulten, K. *J. Mol. Graph.* **1996**, *14*, 33-38.