**PyConSolv 0.9.2**

**User Manual**

Contents

[1. Introduction 1](#_Toc129901150)

[2. Installation 1](#_Toc129901151)

[2.1. Software Requirements 1](#_Toc129901152)

[2.2. Python Requirements 1](#_Toc129901153)

[2.3. Setting up the python environment 2](#_Toc129901154)

[3. Usage 2](#_Toc129901155)

[4. Methodology 3](#_Toc129901156)

[4.1. Program structure 3](#_Toc129901157)

[4.2. Charge Fitting 3](#_Toc129901158)

[4.3. MCPB.py 3](#_Toc129901159)

[4.4. Amber20 3](#_Toc129901160)

[4.5. Solvents 3](#_Toc129901161)

[5. Keywords 4](#_Toc129901162)

[5.1. Keywords for the ConfGen.PyConSolv.run() method 4](#_Toc129901163)

[5.2. Keywords for the pyconsolv.restart file 4](#_Toc129901164)

# Introduction

PyConSolv is a python-based software package meant to simplify the setup for molecular dynamics (MD) simulations for transition metal containing catalysts as well as analyse the resulting trajectories and generate conformers. It interfaces and relies on freely available software packages to execute a state-of-the-art parametrization process.

# Installation

## Software Requirements

To function, PyConSolv requires the following software packages to be installed on your system:

* [ORCA 5.0.x](https://orcaforum.kofo.mpg.de/index.php) (Tested with ORCA 5.0.4)
* [AmberTools](https://ambermd.org/AmberTools.php) (Tested with version 20-22)
* [Multiwfn](http://sobereva.com/multiwfn/) (Tested with version 3.8)

## Python Requirements

To install PyConSolv on your system, Python 3.10 is needed, as well as the RDKit package. RDKit will be installed automatically by PyConSolv.

* Python >= 3.10
* RDKit >= v2022.09.5

## Setting up the python environment

It is recommended to create a virtual environment for PyConSolv, to avoid compatibility problems. Installation is performed as shown in code snipped 1, below:



Code snippet 1 Conda environment creation.

# Usage

To begin the parametrization, an XYZ (XMol format) structure of your complex should be prepared and located in an empty folder. In case a solvent which has not been parametrized is required ([see solvent keyword](#_Keywords_for_the)), an empty folder should be created and an XYZ structure of your desired solvent should be placed inside. The recommended folder structure looks as shown in figure 1, below.



Figure 1 initial folder structure example

Once the desired folder structure is complete, PyConSolv can be started on the input structure.

## Command line usage

The PyConSolv parametrization process can be started by calling the pyconsolv script. For the execution of the script an XYZ input file must be provided.

A number of optional parameters can be passed to the function. For details of the available keywords, please see the [kewords](#_Keywords) section. While default values are provided, it is recommended to change the QM optimization method, as well as the basis set and dispersion corrections to match your needs. Likewise, it is very important to assign a suitable total charge for your complex.

A critical aspect is the solvent. One of the pre-parametrized solvents can be used or, as mentioned above, any user-provided solvent. A simple example is provided in code snipped 2.



Code snippet 2 Simple example on how to run pyconsolv from the command line.

## Usage within python

The ConfGen module must be imported from the PyConSolv package to make the parametrization functions available. To begin the parametrization process, execute the run function of the PyConSolv object.

A number of parameters can be passed to the run function. For details of the available keywords, please see the [kewords](#_Keywords) section. While default values are provided, it is recommended to change the QM optimization method, as well as the basis set and dispersion corrections to match your needs. Likewise, it is very important to assign a suitable total charge for your complex.

A critical aspect is the solvent. One of the pre-parametrized solvents can be used or, as mentioned above, any user-provided solvent. A simple example is provided in code snipped 3.



Code snippet 3 Simple example of parametrization for a neutral compound, while changing settings for the QM method and requesting acetonitrile as a solvent. In this case, the QM method will be DFT with the BP86 functional, def2-SVP basis set, D4 dispersion corrections, using implicit solvation with acetonitrile. The calculation will be run on 8 CPU cores.

The output from PyConSolv will guide the user through the parametrization progress. There are a few steps which must take place for the parametrization, which are explained under [methodology](#_Program_structure).

While the parametrization is running, a more complex folder structure is created, with each critical step being performed in a separate folder, to allow the user to investigate the output of each step in detail as well as use the input/output files for other tasks, unrelated to PyConSolv. The final folder structure looks as shown in figure 2. For a detailed explanation of the content of the files, please consult the [methodology](#_Program_structure_and) section.



Figure 2 Folder structure and important files after parametrization is complete

# Methodology

## Program structure and files

PyConsolv follow a workflow that consists of multiple steps:

1. Folder structure setup

During this step, the folder structure required for PyConSolv is created and the input file is checked for compatibility. If required, the input file is re-organized in such a way as to allow for an error-free parametrization, by grouping the ligand structures together. ORCA input files are also created.

1. Geometry optimization and frequency calculation

Structure optimization and a subsequent frequency calculation is performed using ORCA

1. Fragment splitting

The individual fragments are detected and split up in preparation for parametrization using a depth first search algorithm.

1. Fragment charge assignment

A UI window pops up with a Lewis structure representation of each individual substructure, where the user must provide a charge for the ligand/metal.

1. Generation of initial forcefield parameter files

Each ligand is parametrized using antechamber, with the previously provided total charge.

1. RESP charge calculation

Using Multiwfn 3.8, the RESP charges are [calculated](#_Charge_Fitting).

1. MCPB.py execution

MCPB.py input file is created and the script is then executed, in order to determine the metal parameters.

1. Solvent box construction

A solvent box is constructed, using the desired solvent and counterion (if applicable).

1. Equilibration setup

An equilibration workflow is prepared, along with all the needed input files. Using the GPU accelerated implementation of pmemd (Amber), the system is then equilibrated.

1. Simulation setup

The equilibrated system is copied to the simulation folder and some files required for a final analysis are created. A script is provided for starting the simulation using pmemd (Amber)

1. Analysis

The analysis of a simulation can be performed by calling the pyconsolv script with the -a option and providing the simulation basename as input (e.g. “pyconsolv sim-01 -a”)

## Charge Fitting

Multiwfn 3.81 is used to compute the charges for the parametrizations, as follows.

Nuclear + Electronic Merz-Kollman RESP1 type atomic charge mode is used with 6 points/Å2. All atoms are used for fitting 4 layers, with the scaling factors: 1.4,1.6,1.8,2.0. Automatic radii are utilized for fitting, missing radii are taken from UFF and scaled by 1/1.2. For fitting, the tightness parameter is 0.1, the restraint strengths for the 2 stages are set to 0.0005 and 0.0010, respectively. The maximum number of iterations is 50 and the convergence threshold is set to 10-6. Charge equivalence constraints are enabled for CH2 and CH3 groups. No charge constraint settings are used and the connectivity is determined automatically.

These settings can be changed by the end user, by stopping the parametrization and modifying the multiwfn.input file created in the orca frequency calculation folder, followed by re-running pyconsolv.

## MCPB.py

## Amber20

## Solvents

# Keywords

## Keywords for PyConSolv

* **-c, charge** : charge of the complete system; type int; default 0
* **-m, method** : ORCA 5 method line; type string; default 'PBE0'
* **-b , basis** : Basis set for ORCA calculations; type string; default 'def2-SVP'
* **-d, dsp** : Dispersion corrections; type string; default 'D4'
* **-p, cpu** : number of CPU cores to be used; type int; default 12
* **-s, solvent** : solvent to be used for MD simulation; type string; default 'Water'.

The method, basis and dsp keywords conform to the ORCA 5 naming conventions. In theory any input that would be suitable for an orca optimization can be entered here.

Solvents can be either one of the pre-parametrized ones: Water, Acetonitrile, Acetone, Benzene, Cyclohexane, Chloroform, CCl4, CH2Cl2, DMF, DMSO, Ethanol, Hexane, Methanol, Ammonia, Octanol, THF, Toluene or can take the value 'custom'. When the value 'custom' is passed to pyconsolv, a solvent parametrization dialogue is started. Here, the user is required to input the absolute path to the location of the custom solvent XYZ file and subsequently provide information about the permittivity and refraction index of the solvent.

## Keywords for the pyconsolv.restart file

Once PyConSolv is running, a pyconsolv.restart file is created in the folder where your input XYZ file is located. This file contains a keyword, which represents the last successful step of the parametrization. The values taken can be setup, orca, antechamber, frcmod, multiwfn, mcpb, tleap, equilibration or DONE. If you want to restart the parametrization at a specific point, you may edit this file and restart the PyConSolv process.