JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim

Application Note

PyPE_RESP: A Tool to Facilitate and Standardize Derivation of RESP Charges

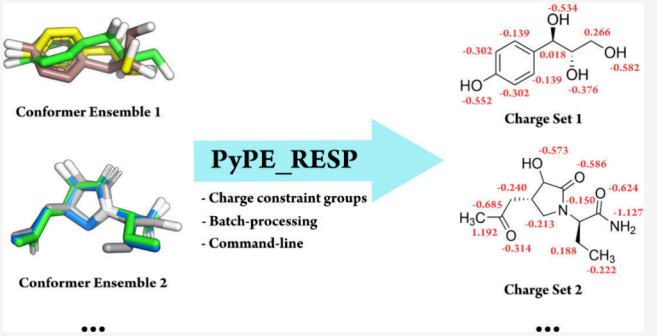
Marco Lapsien, Michele Bonus, Lianne Gahan, Adélaïde Raguin, and Holger Gohlke*

Cite This: J. Chem. Inf. Model. 2025, 65, 4251−4256

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: We introduce PyPE_RESP, a tool to facilitate and standardize partial atomic charge derivation using the Restrained Electrostatic Potential (RESP) approach. PyPE_RESP builds upon the open-source Python package RDKit for chemoinformatics and the AMBER suite for molecular simulations. PyPE_RESP provides an easy setup of multiconformer and multimolecule RESP fitting while allowing a comprehensive definition of charge constraint groups through multiple methods. As a command line tool, PyPE_RESP can be integrated into batch processes. The software enables the derivation of partial atomic charges for additive and polarizable force fields. It outputs constraint group and non-constraint group charges to give an immediate overview of the fit result. PyPE_RESP will be distributed with AmberTools and compatible with most computing platforms.



INTRODUCTION

The accurate representation of electrostatic interactions is essential for developing, extending, and applying empirical force fields used in molecular dynamics simulations.[1] Considering the prominent use of widespread additive force fields such as from the AMBER family[2−4] or CHARMM family,[5,6] and the ongoing development of polarizable force fields including the atomic multipole-optimized energetics for biomolecular applications (AMOEBA),[7,8] polarizable Gaussian Multipole (pGM),[9−11] and OPLS5[12] force fields, this statement holds regardless of the type of force field being developed. Although various methods for deriving partial atomic charges have been described,[13] the restrained electrostatic potential (RESP) approach is still the main charge derivation method used in modern AMBER force fields.[14,15] This can be attributed to its capability to reproduce the quantum mechanical molecular electrostatic potential (ESP) while producing chemically sensible charge distributions.[16] Though RESP was initially developed to obtain fixed atom-centered partial charges for additive force fields, it has been further evolved into the RESP-ind and RESP-perm models implemented in PyRESP[17] with induced and permanent dipole moments for polarizable force fields. A major drawback of the RESP method lies in its setup requirements, as multiple preparation steps must be carried out before atomic charges are obtained for the system of interest. These steps include:

(i) Extraction of electrostatic potential (ESP) information from Gaussian[18] output files (via espgen[19])

(ii) Generation of input files for PyRESP (via PyRESP_GEN[20])

(iii) Insertion of charge constraint and atom equivalency information into PyRESP input files.

While tasks (i) and (ii) can be automated by dedicated software, task (iii) requires either manual editing of input files or creating custom scripts to adhere to the RESP input file format. It additionally requires the manual definition of charge constraint groups (CCGs), meaning a group of atoms whose charge is fixed during fitting by referencing the respective atom indices. The latter can be tedious and error-prone when PyRESP needs to be invoked for multiple molecules. Further complications arise when multiple conformers of a given molecule are considered. Currently, PyRESP_GEN is capable of producing only PyRESP input files for one structure at a time, which therefore involves additional file management by the user to obtain a concatenated PyRESP input file. This also applies to the ESP information, which must be concatenated into a single file. Finally, the obtained RESP charges must be manually input into a structure file (e.g., mol2 format), as charges will only be written to a parameter file created by PyRESP. These concerns have already been tackled to some extent by the development of the RESP plugin for the Psi4 quantum chemistry package[21,22] or the RESP ESP charge derive (R.E.D.) server.[23] Although the R.E.D. server covers

Received: January 9, 2025
Revised: March 17, 2025
Accepted: April 14, 2025
Published: April 26, 2025

https://doi.org/10.1021/acs.jcim.5c00041
J. Chem. Inf. Model. 2025, 65, 4251−4256

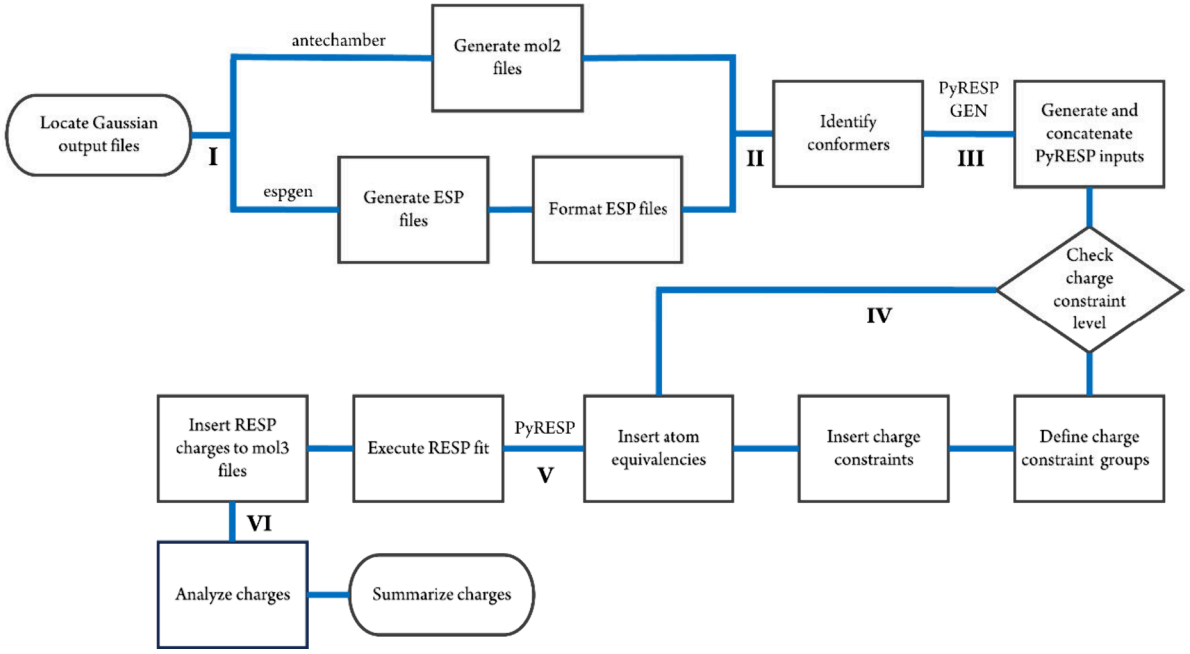


Figure 1. Flowchart highlighting the most important steps in the PyPE_RESP workflow. After each Gaussian output file is located, mol2 and ESP files are created simultaneously (I). Then, the number of conformers per unique molecule is identified by comparing the International Chemical Identifier (InChI) of each molecule (II). First-stage and second-stage PyRESP input files are created by executing PyRESP_GEN and subsequently concatenated to yield two PyRESP input files (III). Depending on the charge constraint level chosen, CCGs are defined based on the information supplied on the command line and appended to the PyRESP input files. The atom equivalency information (i.e., declaring that atoms of all conformers are treated equally) is also appended to the PyRESP input files (IV). Next, PyRESP is executed to start the fitting process (V). The resulting partial atomic charges are written to a newly generated mol3 file for each molecule together with the molecule's connectivity points, meaning atoms connected to, but not part of, a constraint group determined by PyPE_RESP. The derived charges are analyzed based on the mol3 files by utilizing the atom indices of each constraint group to match the corresponding atom charges (VI). Additionally, the sum of charges for each constraint group and the sum of charges of each atom not belonging to the constraint group are calculated. This information is written in a log file and a csv file.

tasks (i) and (ii) well and offers various functionalities, such as integrated geometry optimization and ESP calculation using Gaussian, its syntax for defining CCGs can be overwhelming if many structures need to be parametrized. This is also reflected in the server's naming convention for output files, which can impede the identification of a desired molecular fragment. Additionally, the use of an external server always carries a risk of experiencing periods of downtime and requires the uploading of research data. Furthermore, antechamber,[19] a tool within the AMBER molecular dynamics suite, is capable of rapidly deriving RESP charges but is limited to single conformer fitting. Therefore, the conformational bias of RESP charges is neglected in this approach.[24]

To overcome these issues, we developed PyPE_RESP, a Python-based piPEline to conveniently and systematically execute multiconformer and multimolecule RESP fitting within the realm of AMBER, using Gaussian for ESP computation. PyPE_RESP will be distributed under a GPL license together with AmberTools[19] (https://ambermd.org/AmberTools.php).

WORKFLOW DESCRIPTION

The PyPE_RESP workflow minimizes the user's manual work while providing flexible setups and reliable and reproducible results of charge fitting. A flowchart of the main steps of PyPE_RESP is presented in Figure 1. PyPE_RESP builds upon existing tools (antechamber,[19] espgen, PyRESP_GEN,[20] and PyRESP[17]) and manages their in- and outputs, enabling it to be adaptable toward specific properties of each system to be parametrized and forming an "all-in-one" solution for RESP charge derivation. The only input files required are the Gaussian output files containing the ESP information for each conformer of each molecule. These files were gathered in a dedicated directory. All other specifications for the RESP run are managed by flags submitted via the PyPE_RESP command line interface (CLI) or a PyPE_RESP input file. The latter is convenient for the repetitive processing of multiple molecules. Most flags in the CLI utilize PyPE_RESP's inherent enumeration of CCGs, which is illustrated in Figure 2.

Definition of Charge Constraint Groups. During the electrostatic parametrization of fragments derived from large molecules, they are often equipped with a capping group that mimics the chemical environment in the assembled molecule. Since these capping groups are only attached temporarily, charge constraints are applied to account for their removal after partial atomic charges have been derived. PyPE_RESP indexes each CCG (starting from 1) in the order they are defined under the `--constraint` flag. No additional specification is needed if a fit without charge constraints is desired, since PyPE_RESP will automatically carry out unconstrained RESP fitting if no CCGs were specified.

PyPE_RESP accepts two ways for defining CCGs under the `--constraint` flag. First, groups of atom indices were supplied to be constrained. The atom indices can be taken directly from the Gaussian output files that PyPE_RESP uses as inputs. Still, a visual inspection of the molecule's three-dimensional structure is most likely required to determine all indices of atoms of a group constraint. While visualization software exists that can read Gaussian output files, common

https://doi.org/10.1021/acs.jcim.5c00041
J. Chem. Inf. Model. 2025, 65, 4251−4256

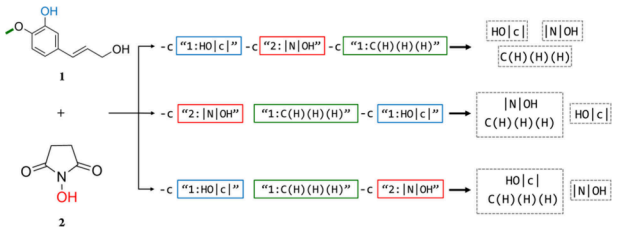


Figure 2. Illustration of the definitions of intra- and intermolecular CCGs in PyPE_RESP using arbitrary constraints. CCGs are defined using the `-c/--constraint` flag and are encapsulated in quotation marks. Three cases are shown: 1. Each CCG is defined separately, leading to three intramolecular CCGs (top). 2. The methyl group of compound **1** (green) and the hydroxyl group of compound **2** (red) are associated together, whereas the hydroxyl group of compound **1** (cyan) defines a separate CCG, leading to one intermolecular and one intramolecular CCG (middle). 3. The methyl group of compound **1** (green) and the hydroxyl group of compound **1** (cyan) are associated together, while the hydroxyl group of compound **2** (red) forms a separate CCG (bottom). The latter example leads to a faulty definition of an intermolecular CCG between the hydroxyl and methyl groups of compound **1**, which will force PyPE_RESP to exit and therefore underlines the importance of adhering to the PyPE_RESP syntax.

molecule visualization software, such as VMD[25] or PyMOL,[26] cannot do this without further modification. To overcome this, PyPE_RESP allows carrying out a conversion run if the `--mol2_only` flag is supplied. In this case, PyPE_RESP will generate mol2 files for each Gaussian output file found using the antechamber and exit afterward. Second, defining CCGs is based on matching substructures of molecules generated from SMARTS patterns by utilizing the RDKit Open-source cheminformatics package (Figure 2).[27] Both ways follow the same syntax where the molecule to assign the CCG is specified, followed by a colon and the SMARTS pattern to be used for substructure matching, or the space-separated atom indices. In both cases, the CCG must be enclosed in quotation marks to be read as a single argument. Molecules are indexed starting from 1 and can be referenced by their index to define CCGs. Indexing is either determined automatically based on the ascending alphabetical order of filenames or by specifying wildcard patterns matching the respective filenames, which will be resolved using Python's glob module. PyPE_RESP accepts only unambiguous SMARTS patterns and raises an error if multiple substructure matches are found for a given pattern.

The use of SMARTS patterns often requires less initial setup than defining CCGs by atom indices but can create long and complex character sequences to match a specific functional group. To verify whether a pattern matches one or multiple substructures, the `--test_smarts` flag can be utilized. Then a test run is carried out, in which CCGs defined under the `--constraint` flag via SMARTS patterns are matched against the respective target molecule and the number of substructure matches, along with the atom indices corresponding to each match per CCG, is returned. Any CCG defined via atom indices will be ignored in this case. If a SMARTS pattern is found to be ambiguous, a warning is displayed. Additionally, an incorrectly specified SMARTS string would force the program to exit.

SMARTS patterns often include atoms that are required to ensure unambiguity but are not meant to be part of the final CCG. To neglect these atoms later during fitting, "SMARTS slicing" can be applied by enclosing the parts of the SMARTS pattern that are neglected in pipe symbols (Figure 2). PyPE_RESP recognizes these symbols and removes the atom indices corresponding to the enclosed section after the full pattern has been used for substructure matching.

Setting Fit Properties. Besides defining single CCGs, one needs to consider how intermolecular charge constraints are defined. This is also accomplished using the `--constraint` flag (Figure 2). The flag can be used multiple times and accepts up to two arguments corresponding to two CCGs. If a single CCG is defined, it is treated as an intramolecular CCG, whereas two CCGs from different molecules are treated as intermolecular CCGs.

The assignments of the constraint group and molecular charges are another crucial step in RESP charge derivation. While molecular charges do not need to be defined as they are taken from the respective Gaussian input file, constraint group charges are set to zero by default but can be assigned by specifying the group charge during the definition of the respective constraint group (see Examples section).

Besides the traditional RESP method, PyPE_RESP also allows using the RESP-ind and RESP-perm models that are integrated into the PyRESP software. The associated flags (`--ptype` and `--polariz`) are identical to those used in PyRESP and PyRESP_GEN. These flags do not need to be specified if the usual point-charge model ("chg") is chosen.

PyRESP execution can be prohibited by supplying the `--no_pyresp` flag, which stops PyPE_RESP after the generation of PyRESP input files to allow for analysis by the user. PyPE_RESP also provides insights and statistics for each run carried out. A log file is created that documents the key steps during the generation of the PyRESP input files. It includes the number of unique molecules found and the number of conformers per molecule detected. Moreover, a second log file is created after the successful execution of PyRESP, which contains the fit statistics provided by PyRESP and highlights the sum of charges for each CCG, the sum of charges of all atoms that do not belong to the respective constraint group, and the difference between these two sums. This information is also stored in a csv file to facilitate integration into data analysis software for further inspection

EXAMPLES

Each one of the following example cases (Figure 3) will be carried out as a multiconformer RESP fit. Refer to the Supporting Information for details about conformer generation. The described examples focus on using the PyPE_RESP CLI instead of PyPE_RESP input files. Detailed information about the PyPE_RESP input file syntax can be obtained via the

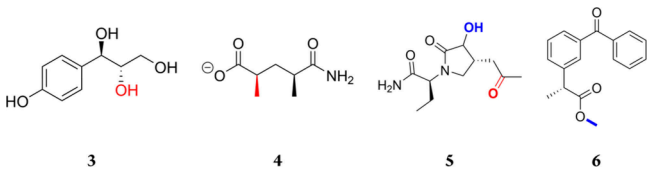


Figure 3. Example compounds illustrate the use of PyPE_RESP. **3**: (1*R*,2*S*)-1-(4-hydroxyphenyl)propane-1,2,3-triol. **4**: (2*R*,4*S*)-4-carbamoyl-2,4-dimethylbutanoic acid. **5**: (2*S*)-2-((4*R*)-3-hydroxy-2-oxo-4-(2-oxopropyl)pyrrolidin-1-yl)butanamide. **6**: (2*R*)-methyl-2-(3-benzoylphenyl)propanoate. Intramolecular CCGs are highlighted in red, whereas groups that are part of an intermolecular charge constraint are shown in blue.

https://doi.org/10.1021/acs.jcim.5c00041
J. Chem. Inf. Model. 2025, 65, 4251−4256

`--input_help` flag, and a template input file can be generated via the `--template` flag. Input files, including Gaussian output files and PyPE_RESP input files, for each example are provided in the Supporting Information. Note that all CCGs were chosen to highlight PyPE_RESP's functionality and not based on chemical relevance.

CASE A: Using SMARTS Patterns to Define CCGsg. The benefits of SMARTS slicing to define an intramolecular CCG can be illustrated using the hydroxyl group bound to the carbon atom at position two of molecule **3** (Figure 3). Since this compound possesses multiple hydroxyl groups, the unambiguous SMARTS pattern to identify the group of interest must include the neighboring aliphatic carbon atom, as well as the aromatic carbon attached to it. This results in the following SMARTS pattern: HOCCc. The three carbon atoms in this sequence are not meant to be part of the final CCG and therefore must be neglected during fitting using SMARTS slicing. To achieve this, the three carbon atoms must be enclosed in pipe symbols. The whole pattern must also be enclosed in quotation marks to prevent interpretation by the shell. Thus, the full command to identify the 2-OH group for an intramolecular CCG is `pype-resp.py --constraint "1:HO|CCc|"`.

Notably, the hydrogen atom is not enclosed in square brackets, unlike what is required by the SMARTS language convention. PyPE_RESP automatically adds the brackets, although they will be recognized if added by the user. Identification of the target molecule by its index (1) was chosen here for convenience, as it is the only molecule to be parametrized. A wildcard pattern to identify the molecule would also work.

Figure S1 highlights how for some atoms of **3** the derived charges vary between conformers, motivating the use of a multiconformer fit.

CASE B: Defining Intramolecular CCGs Using Atom Indices and the RESP-ind Model. Besides the definition of CCGs via SMARTS patterns, PyPE_RESP also allows the definition of CCGs using atom indices, if desired. To illustrate this, the methyl group next to the carboxylic acid group of molecule **4** (Figure 3) is used to define an intramolecular CCG. PyPE_RESP offers two possibilities to identify atom indices. One option is to carry out a test run by supplying the `--test_smarts` flag with the SMARTS pattern C(H)(H)(H)|C(O)(O)|, which retrieves the atom indices corresponding to the target methyl group. The indices of the group can then be used to define a CCG using the `--constraint` flag in a production run. Alternatively, one can carry out a PyPE_RESP run with the `--mol2_only` flag to generate mol2 files, which can then be visualized to obtain the atomic indices of the methyl group atoms. Both approaches reveal the atoms with indices 11, 21, 22, and 23 to be part of the target functional group. Atomic partial charges for the compound can be obtained using the RESP-ind model, for which the "ind" keyword must be supplied under the `--ptype` flag. The use of the RESP-ind and RESP-perm models requires information about individual atomic polarizabilities. The absolute or relative path to a file containing this information can be supplied under the `--polariz` flag (an example is provided in the Supporting Information). Finally, the charge of the constraint group for the methyl moiety is set to −1 by adding the colon-separated charge after the definition of the constraint group. The full command is `pype-resp.py --constraint "1:11 21 22 23:-1" --ptype ind --polariz Polarizability_info.txt`

Figure S2 indicates for **4** that PyPE_RESP-derived charges agree with those from the RESP algorithm and PyRESP.

CASE C: Defining Intermolecular Constraints by SMARTS Matching and Wildcard Patterns. If multiple molecules need to be parametrized, it is reasonable to identify them via wildcard patterns instead of their CCG indices to avoid mismatches. Intermolecular charge constraints are defined by supplying two CCGs under one `--constraint` flag. Taking molecule **5** and molecule **6** (Figure 3), the hydroxyl group attached to the pyrrolidine ring of **5** is constrained together with the methyl group of **6**. The carbonyl carbon and carbonyl oxygen of **5** additionally form an intramolecular CCG, whose charge is arbitrarily set to −2. This results in the definition of three CCGs, with two being constrained together. Finally, the RESP-perm model is invoked by supplying the option "perm" under the−ptype flag. The typical command that achieves this is `pype-resp.py --constraint "butanamide:OH" "propanoate:C(H)(H)(H)|O|:0" --constraint "butanamide:|[C;H3]|C = O:-2" --ptype perm --polariz Polarizability_info.txt`

This example also illustrates how the charge for intermolecular constraint groups can be set. The syntax is identical to the one used for intramolecular CCGs, but the total charge of the intermolecular constraint will be set to the charge specified for the second CCG. As mentioned above, if the charge is supposed to be zero, no specification is needed, and it is specified only here for illustration purposes. Finally, note that wildcard patterns identifying the respective molecules do not contain asterisks. These are added automatically by PyPE_RESP during the globing process.

DISCUSSION

We present PyPE_RESP to facilitate and standardize the derivation of atomic partial charges following the RESP procedure. PyPE_RESP uses the RDKit open-source chemoinformatics package and those built into PyPE_RESP of AmberTools to derive charges for additive or polarizable force fields, while providing multiple means to define CCGs in nontrivial cases. PyPE_RESP improves the reproducibility of parametrization tasks, which would otherwise be carried out by hand, and provides informative postrun statistics that allow evaluating the fitting process. While most of these features are also implemented in the R.E.D. server, PyPE_RESP offers the benefit of being executed locally. This circumvents depending on an external server, which eliminates issues arising from downtime periods as well as the need to upload scientific data and makes the resubmission of jobs easier if minor mistakes in the command or input file were detected in a prior run. The runtime of PyPE_RESP is comparable to that of PyRESP (Figure S3).

We acknowledge that the syntax required by PyPE_RESP can be overwhelming initially. However, in the context of force field design, or comprehensive molecular design and analysis studies, where numerous compounds must be parametrized, the benefits of PyPE_RESP's systematic approach stand out. Similarly, creating SMARTS patterns to target components of complex molecules such as multiring systems can be troublesome, e.g., if they are intertwined. In such a case, an inexperienced user may need to go through a lengthy trial-and-error process to match the desired substructure. Such scenarios

https://doi.org/10.1021/acs.jcim.5c00041
J. Chem. Inf. Model. 2025, 65, 4251−4256

can be prevented, however, by falling back to the definition of CCGs via atom indices, further highlighting that PyPE_RESP's functionality can be adapted according to system complexity. Although PyPE_RESP offers a range of options to cover the needs of most RESP charge derivation procedures, it does not yet account for niche cases where additional fine-tuning of parameters or reorientation of molecules before fitting is required.

Finally, PyPE_RESP is compatible with most operating systems. This includes common Linux distributions, MacOS, and Windows 11 using the Windows Subsystem for Linux (WSL).

■ ASSOCIATED CONTENT

Data Availability Statement

The Gaussian output files per conformer for each molecule discussed in the Examples section, along with PyPE_RESP input files and all files generated by PyPE_RESP for each example case, are provided as Supporting Information.

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.5c00041.

Details on the generation of conformer ensembles and ESP calculation for each compound discussed in the Examples section (PDF)

Gaussian output files per conformer for each molecule discussed in the Examples section; PyPE_RESP input files and all files generated by PyPE_RESP for each example case (ZIP)

■ AUTHOR INFORMATION

Corresponding Author

Holger Gohlke − *Institute for Pharmaceutical and Medicinal Chemistry & Bioeconomy Science Center (BioSC), Heinrich Heine University Düsseldorf, 40225 Düsseldorf, Germany; Institute of Bio- and Geosciences (IBG-4: Bioinformatics), Forschungszentrum Jülich, 52425 Jülich, Germany;* orcid.org/0000-0001-8613-1447; Email: gohlke@uni-duesseldorf.de

Authors

Marco Lapsien − *Institute for Pharmaceutical and Medicinal Chemistry & Bioeconomy Science Center (BioSC), Heinrich Heine University Düsseldorf, 40225 Düsseldorf, Germany*

Michele Bonus − *Institute for Pharmaceutical and Medicinal Chemistry & Bioeconomy Science Center (BioSC), Heinrich Heine University Düsseldorf, 40225 Düsseldorf, Germany;* orcid.org/0000-0003-4411-7342

Lianne Gahan − *Institute for Computational Cell Biology & Bioeconomy Science Center (BioSC), Heinrich Heine University Düsseldorf, 40225 Düsseldorf, Germany*

Adélaïde Raguin − *Institute for Computational Cell Biology & Bioeconomy Science Center (BioSC), Heinrich Heine University Düsseldorf, 40225 Düsseldorf, Germany*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.5c00041

Author Contributions

H.G. designed the study; M.L. wrote the software code; M.B. and L.G. supported software design; M.L., A.R., and H.G. wrote the manuscript; M.B. and L.G. revised the manuscript; H.G. and A.R. secured funding.

Funding

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy-EXC 2048/1-project (ID 390686111) and a grant from the Ministry of Innovation, Science, and Research of North-Rhine Westfalia (NRW) within the framework of the NRW Strategieprojekt BioSC (No. 313/323-400-002 13) by the BOOST FUND 2.0 project "OptiCellu".

Notes

The authors declare no competing financial interest.

The used software RDKit (https://www.rdkit.org/), Open Babel (https://openbabel.org/index.html), XTB (https://github.com/grimme-lab/xtb), CREST (https://crest-lab.github.io/crest-docs/), CENSO (https://github.com/grimme-lab/CENSO), and Gaussian (https://gaussian.com/) are available according to the provided web resources. PyPE_RESP will be distributed with AmberTools (https://ambermd.org/AmberTools.php).

■ ACKNOWLEDGMENTS

We are grateful for the computational infrastructure and support provided by the "Zentrum für Informations- und Medientechnologie" (ZIM) at Heinrich Heine University Düsseldorf and the computing time provided by the John von Neumann Institute for Computing (NIC) to H.G. on the supercomputer JUWELS at Jülich Supercomputing Centre (JSC) (user IDs: VSK33, lignin). We acknowledge the developers of AMBER and the RDKit cheminformatics package for making their software available publicly.

■ ABBREVIATIONS

AMBER	Assisted Model Building with Energy Refinement
AMOEBA	Atomic Multipole Optimized Energetics for Biomolecular Applications
CCG	Charge constraint group
CLI	Command Line Interface
ESP	Electrostatic Potential
InChI	International Chemical Identifier
R.E.D.	RESP ESP charge Derive
RESP	Restrained Electrostatic Potential
SMARTS	SMILES Arbitrary Target Specification
WSL	Windows Subsystem for Linux

■ REFERENCES

(1) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Oxford University Press, 2017.

(2) Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. Development and testing of a general amber force field. *J. Comput. Chem.* **2004**, *25* (9), 1157−1174.

(3) Tian, C.; Kasavajhala, K.; Belfon, K. A. A.; Raguette, L.; Huang, H.; Migues, A. N.; Bickel, J.; Wang, Y.; Pincay, J.; Wu, Q.; et al. ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. *J. Chem. Theory Comput.* **2020**, *16* (1), 528−552.

(4) Galindo-Murillo, R.; Robertson, J. C.; Zgarbová, M.; Šponer, J.; Otyepka, M.; Jurečka, P.; Cheatham, T. E., III. Assessing the Current State of Amber Force Field Modifications for DNA. *J. Chem. Theory Comput.* **2016**, *12* (8), 4114−4127.

(5) Vanommeslaeghe, K.; Hatcher, E.; Acharya, C.; Kundu, S.; Zhong, S.; Shim, J.; Darian, E.; Guvench, O.; Lopes, P.; Vorobyov, I.; et al. CHARMM general force field: A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields. *J. Comput. Chem.* **2010**, *31* (4), 671−690.

(6) Vanommeslaeghe, K.; MacKerell, A. D. CHARMM additive and polarizable force fields for biophysics and computer-aided drug design. *Biochimica et Biophysica Acta (BBA) - General Subjects* **2015**, *1850* (5), 861−871.
(7) Zhang, C.; Lu, C.; Jing, Z.; Wu, C.; Piquemal, J.-P.; Ponder, J. W.; Ren, P. AMOEBA Polarizable Atomic Multipole Force Field for Nucleic Acids. *J. Chem. Theory Comput.* **2018**, *14* (4), 2084−2108.
(8) Ponder, J. W.; Wu, C.; Ren, P.; Pande, V. S.; Chodera, J. D.; Schnieders, M. J.; Haque, I.; Mobley, D. L.; Lambrecht, D. S.; DiStasio, R. A., Jr.; et al. Current Status of the AMOEBA Polarizable Force Field. *J. Phys. Chem. B* **2010**, *114* (8), 2549−2564.
(9) Zhao, S.; Wei, H.; Cieplak, P.; Duan, Y.; Luo, R. Accurate Reproduction of Quantum Mechanical Many-Body Interactions in Peptide Main-Chain Hydrogen-Bonding Oligomers by the Polarizable Gaussian Multipole Model. *J. Chem. Theory Comput.* **2022**, *18* (10), 6172−6188.
(10) Zhao, S.; Cieplak, P.; Duan, Y.; Luo, R. Assessment of Amino Acid Electrostatic Parametrizations of the Polarizable Gaussian Multipole Model. *J. Chem. Theory Comput.* **2024**, *20* (5), 2098−2110.
(11) Zhao, S.; Cieplak, P.; Duan, Y.; Luo, R. Transferability of the Electrostatic Parameters of the Polarizable Gaussian Multipole Model. *J. Chem. Theory Comput.* **2023**, *19* (3), 924−941.
(12) Damm, W.; Dajnowicz, S.; Ghoreishi, D.; Yu, Y.; Ganeshan, K.; Madin, O.; Rudshteyn, B.; Hu, R.; Wu, M.; Shang, Y. OPLS5: Addition of Polarizability and Improved Treatment of Metals. *ChemRxiv* **2024**, DOI: 10.26434/chemrxiv-2024-2svr7.
(13) Wiberg, K. B.; Rablen, P. R. Atomic Charges. *Journal of Organic Chemistry* **2018**, *83* (24), 15463−15469.
(14) Bayly, C. I.; Cieplak, P.; Cornell, W.; Kollman, P. A. A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: the RESP model. *J. Phys. Chem.* **1993**, *97* (40), 10269−10280.
(15) Case, D. A.; H. M. A; Belfon, K.; Ben-Shalom, I. Y.; Berryman, J. T.; Brozell, S. R.; Cerutti, D. S.; Cheatham, T. E.; Cisneros, G. A.; Cruzeiro, V. W. D.; Darden, T. A.; Forouzesh, N.; Giambaşu, G.; Giese, T.; Gilson, M. K.; Gohlke, H.; Goetz, A. W.; Harris, J.; Izadi, S.; Izmailov, S.A.; Kasavajhala, K.; Kaymak, M.C.; King, E.; Kovalenko, A.; Kurtzman, T.; Lee, T. S.; Li, P.; Lin, C.; Liu, J.; Luchko, T.; Luo, R.; Machado, M.; Man, V.; Manathunga, M.; Merz, K. M.; Miao, Y.; Mikhailovskii, O.; Monard, G.; Nguyen, H.; O'Hearn, K. A.; Onufriev, A.; Pan, F.; Pantano, S.; Qi, R.; Rahnamoun, A.; Roe, D. R.; Roitberg, A.; Sagui, C.; Schott-Verdugo, S.; Shajan, A.; Shen, J.; Simmerling, C.L.; Skrynnikov, N. R.; Smith, J.; Swails, J.; Walker, R. C.; Wang, J.; Wang, J.; Wei, H.; Wu, X.; Wu, Y.; Xiong, Y.; Xue, Y.; York, D. M.; Zhao, S.; Zhu, Q.; Kollman, P. A. *Amber 2023*; AMD, 2023.
(16) Wang, J.; Cieplak, P.; Kollman, P. A. How well does a restrained electrostatic potential (RESP) model perform in calculating conformational energies of organic and biological molecules? *J. Comput. Chem.* **2000**, *21* (12), 1049−1074.
(17) Zhao, S.; Wei, H.; Cieplak, P.; Duan, Y.; Luo, R. PyRESP: A Program for Electrostatic Parameterizations of Additive and Induced Dipole Polarizable Force Fields. *J. Chem. Theory Comput.* **2022**, *18* (6), 3654−3670.
(18) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; et al. *Gaussian 16* Rev. C.01; Gaussian Inc., 2016.
(19) Case, D. A.; Aktulga, H. M.; Belfon, K.; Cerutti, D. S.; Cisneros, G. A.; Cruzeiro, V. W. D.; Forouzesh, N.; Giese, T. J.; Götz, A. W.; Gohlke, H.; et al. AmberTools. *J. Chem. Inf. Model.* **2023**, *63* (20), 6183−6191.
(20) Zhu, Q.; Wu, Y.; Zhao, S.; Cieplak, P.; Duan, Y.; Luo, R. Streamlining and Optimizing Strategies of Electrostatic Parameterization. *J. Chem. Theory Comput.* **2023**, *19* (18), 6353−6365.
(21) Alenaizan, A.; Burns, L. A.; Sherrill, C. D. Python implementation of the restrained electrostatic potential charge model. *Int. J. Quantum Chem.* **2020**, *120* (2), No. e26035.
(22) Parrish, R. M.; Burns, L. A.; Smith, D. G. A.; Simmonett, A. C.; DePrince, A. E., III; Hohenstein, E. G.; Bozkaya, U.; Sokolov, A. Y.; Di Remigio, R.; Richard, R. M.; et al. Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability. *J. Chem. Theory Comput.* **2017**, *13* (7), 3185−3197.
(23) Vanquelef, E.; Simon, S.; Marquant, G.; Garcia, E.; Klimerak, G.; Delepine, J. C.; Cieplak, P.; Dupradeau, F.-Y. R.E.D. Server: a web service for deriving RESP and ESP charges and building force field libraries for new molecules and molecular fragments. *Nucleic Acids Res.* **2011**, *39* (suppl_2), W511−W517.
(24) Williams, D. E. Alanyl dipeptide potential-derived net atomic charges and bond dipoles, and their variation with molecular conformation. *Biopolymers* **1990**, *29* (10−11), 1367−1386.
(25) Humphrey, W.; Dalke, A.; Schulten, K. VMD - Visual Molecular Dynamics. *J. Mol. Graphics* **1996**, *14*, 33−38.
(26) Schrodinger, L. L. C. *The PyMOL Molecular Graphics System*, Version 1.8; 2015.
(27) Landrum, G. *RDKit: Open-source cheminformatics*; https://www.rdkit.org (accessed August 28th, 2024). 2010.

https://doi.org/10.1021/acs.jcim.5c00041
J. Chem. Inf. Model. 2025, 65, 4251−4256