The Journal of Open Source Software

AutoGAMESS: A Python package for automation of GAMESS(US) Raman calculations

Brian C. Ferrari[1]

1 Department of Physics, University of Central Florida, 4111 Libra Drive, Orlando FL 32816

DOI: 10.21105/joss.01612

Software

- Review
- Repository
- Archive

Submitted: 27 July 2019
Published: 03 September 2019

Summary

The *ab initio* quantum chemistry software GAMESS(US) (M. S. Gordon & Schmidt, 2005; M. W. Schmidt et al., 1993) is capable of calculating a variety of molecular properties. One of the many popular uses of GAMESS(US) is the prediction of properties of volatile and unstable species that have not been experimentally characterized or quantified before by chemists, physicists, astro-chemists and astro-physicists (Bennett, Ennis, & Kaiser, 2014; Burda, Pavelka, & Šimánek, 2004; Hickman, Miles, Hayden, & Talbi, 2005; Pacifici, Verdicchio, Lago, Lombardi, & Costantini, 2013). Applications of, and research done using, GAMESS(US) is not limited to uncharacterized species; it's also widely used in material characterization or material property prediction research. Research utilizing these types of *ab initio* calculations typically require calculations with multiple steps required to achieve each final result. For instance, a Raman activity prediction first requires a geometry optimization and Hessian calculation be performed on the molecule, making automation extremely beneficial. This leads to complicated and tedious workflows slowing a user's research.

Oftentimes single calculations of molecular properties are not reliable, resulting in publications requiring several calculations, each implementing either a different level of theory or basis set, for each property be done on each molecule. This has brought about a demand for high throughput data calculation packages which automate these *ab initio* calculations (Bhoorasingh, Slakman, Seyedzadeh Khanshan, Cain, & West, 2017; Kiran Mathew, 2017; Larsen et al., 2017); as well as workflow management systems (Krogel, 2016) and data parsers (O'boyle, Tenderholt, & Langner, 2008). Specific packages have also been developed to compliment GAMESS(US) (Allouche, 2011; Bode & Gordon, 1998; DerMardirossian & Bokoch, 2005; Schaftenaar & Noordik, 2000; J. Schmidt & Polik, 2013). However, because these programs are largely visualization and graphical programs, there is still a need for packages that automate GAMESS(US) Raman calculations. Automation is essential to generate large databases of Raman data, which could have further applications for machine learning of Raman data. As it stands, the automation of Raman calculations is either not being done, or being implemented individually by each research group utilizing the GAMESS(US) software. This slows scientific progress down, and an automation software written in a language extremely simple and well adopted by scientists, such as Python, is an attractive solution to the problem.

AutoGAMESS provides an open source, Python-based software for automating conversion between optimization calculations to Hessian calculations and then to Raman calculations. It also offers automation of data collection from the output files, for quick tabular data readouts of each calculation. AutoGAMESS has currently been used for a study presented at the 30th Annual Conference on Computation Physics CCP2018 that will be published in the conference proceedings. AutoGAMESS is also currently being used in multiple other computational chemistry projects, soon to be published by scientists at the University of Central Florida.

AutoGAMESS provides the following public interfaces:

- `new_project`: Builds a directory tree for housing input/output files with spreadsheets for collected data.
- `input_builder`: Builds optimization input files based on text file specifications
- `opt2hes`: Converts optimization input files into Hessian input files
- `hes2Raman`: Converts Hessians input files into Raman input files
- `sort_logs`: Sorts GAMESS(US) output files
- `fill_spreadsheet`: Fills in Excel Spreadsheets with data collected from log files
- `get_data`: Collects data from output files
- `make_plot`: Makes a vibrational frequency vs. IR/Raman intensity line plot

Capabilities

AutoGAMESS is capable of initializing an entire directory with well-organized subdirectories for housing all input and output files. This main directory will also contain spreadsheets that AutoGAMESS is later capable of filling with the data collected from parsing the output files. Once a main directory is initialized, input files can be generated for GAMESS(US) optimization calculations. Requiring only a simple CSV file as input, AutoGAMESS' `input_builder` function can generate thousands of files with any form of internal GAMESS(US) level of theory and both internal and external basis sets. External basis sets must be a part of EMSL Basis Set Exchange (Benjamin P. Pritchard, 2019; Feller, 1996; Schuchardt et al., 2007). This requirement is due to the integration of the EMSL basis set exchange Python package into AutoGAMESS. After the user has run a geometry optimization calculation, AutoGAMESS is able to quickly get the required data from the output to modify the geometry optimization input file into a Hessian calculation input file. Similarly, after a Hessian calculation AutoGAMESS can use the output to quickly generate a Raman calculation input file. Once all calculations a user desires to run have been completed, AutoGAMESS can sort the output files, then parse files for specific molecular properties and fill in the spreadsheets that had been generated initially. All Hessian and Raman data is pulled directly from output files, while geometry optimization properties, such as bond lengths and angles, are calculated by AutoGAMESS. Bond lengths are calculated by using the simple Euclidean distance formula,

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2},$$

while bond angles are calculated by first performing a translation on the Cartesian coordinates, of the general form, $P(x, y, z) \rightarrow P'(x - a, y - b, z - c)$, where (a, b, c) are the coordinates of a central atom. Then the angle between two atoms with a third as the origin is found using the equation

$$A = \arccos\left(\hat{V_1} \cdot \hat{V_2}\right)$$

where A is the bond angle and $\hat{V_1}$ and $\hat{V_2}$ are the normalized position vectors for each atoms location in relation to the modified origin.

AutoGAMESS is also capable of generating line plots of the vibrational frequency vs. IR/Raman intensity. Generated plots will be titled with the molecule name in the file and the theory/basis set used for the calculation. Each symmetry group will be plotted in a different color, from either a default or user-specified color list. The spectral line (sum of line broadening) will also be plotted in red with 50% transparency. Figure 1 shows an example line plot of H_2O using B3LYP/CCD for the calculation.

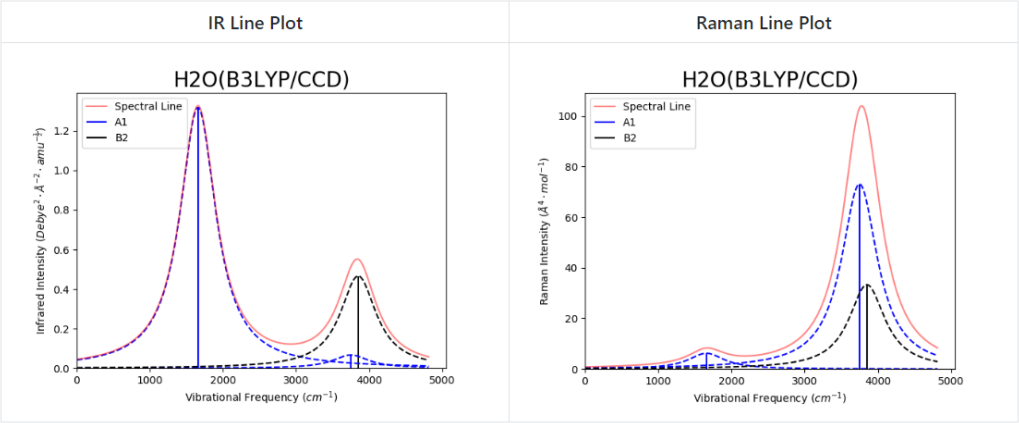

Figure 1: Line Plots Example

AutoGAMESS can also generate scaling factors for vibrational frequencies using the least squares method by Scott & Radom (1996). This method involves minimizing the residuals, Δ, given by

$$\Delta = \sum_{i}^{\text{all}} (\lambda \omega_i^{\text{theory}} - \nu_i^{\text{expt}})^2,$$

resulting in

$$\lambda = \frac{\sum_i^{\text{all}} \omega_i^{\text{theory}} \nu_i^{\text{expt}}}{\sum_i^{\text{all}} (\omega_i^{\text{theory}})^2},$$

where ω_i^{theory} and ν_i^{expt} are the ith theoretical harmonic and ith experimental fundamental frequencies (in cm^{-1}) and λ is the scaling factor. The root mean square error is then given by

$$\text{rms} = (\frac{\sum_i^n \Delta_{\text{min}}}{n})^{\frac{1}{2}},$$

where n is the number of frequency modes for the molecule and Δ_{min} is the minimized residual for each particular mode.

Use of AutoGAMESS

AutoGAMESS was developed to be versatile in its usability; several examples (in both shell and Python scripts) can be found in the software's GitHub repository.

Availability

This software is distributed under the MIT License and can be installed through Python's `pip` install command.

```
python -m pip install autogamess --user
```

Dependencies

AutoGAMESS requires all the following Python packages:

- Python-3.x
- NumPy (T. E. Oliphant, 2006; Walt, Colbert, & Varoquaux, 2011)
- SciPy (Jones, Oliphant, Peterson, & others, 2001)
- Pandas (McKinney, 2010)
- basis_set_exchange
- PeriodicElements
- openpyxl
- Matplotlib (Hunter, 2007)

Acknowledgements

The author would like to thank Dr. Christopher J. Bennett, Remington Cantelas and Sarah Swiersz for helpful discussions while developing the package.

References

Allouche, A.-R. (2011). Gabedit—a graphical user interface for computational chemistry softwares. *Journal of Computational Chemistry*, *32*(1), 174–182. doi:10.1002/jcc.21600

Benjamin P. Pritchard, B. D., Doaa Altarawy. (2019). A new basis set exchange: An open, up-to-date resource for the molecular sciences community. *Manuscript in Preparation*.

Bennett, C. J., Ennis, C. P., & Kaiser, R. I. (2014). Experimental studies on the formation of D_2O and D_2O_2 by implantation of energetic D^+ ions into oxygen ices. *The Astrophysical Journal*, *782*(2), 63. doi:10.1088/0004-637x/782/2/63

Bhoorasingh, P. L., Slakman, B. L., Seyedzadeh Khanshan, F., Cain, J. Y., & West, R. H. (2017). Automated transition state theory calculations for high-throughput kinetics. *The Journal of Physical Chemistry A*, *121*(37), 6896–6904. doi:10.1021/acs.jpca.7b07361

Bode, B. M., & Gordon, M. S. (1998). Macmolplt: A graphical user interface for GAMESS. *Journal of Molecular Graphics and Modelling*, *16*(3), 133–138. doi:10.1016/S1093-3263(99)00002-9

Burda, J. V., Pavelka, M., & Šimánek, M. (2004). Theoretical model of copper Cu(I)/Cu(II) hydration. DFT and ab initio quantum chemical study. *Journal of Molecular Structure: THEOCHEM*, *683*(1-3), 183–193. doi:10.1016/j.theochem.2004.06.013

DerMardirossian, C., & Bokoch, G. M. (2005). GDIs: Central regulatory molecules in Rho GTPase activation. *Trends in Cell Biology*, *15*(7), 356–363. doi:10.1016/j.tcb.2005.05.001

Feller, D. (1996). The role of databases in support of computational chemistry calculations. *Journal of Computational Chemistry*, *17*(13), 1571–1586. doi:10.1002/(SICI)1096-987X(199610)17:13<1571::AID-JCC9>3.0.CO;2-P

Gordon, M. S., & Schmidt, M. W. (2005). Chapter 41 - advances in electronic structure theory: GAMESS a decade later. In *Theory and Applications of Computational Chemistry* (pp. 1167–1189). Amsterdam: Elsevier. doi:10.1016/B978-044451719-7/50084-6

Hickman, A., Miles, R., Hayden, C., & Talbi, D. (2005). Dissociative recombination of e + $HCNH^+$: Diabatic potential curves and dynamics calculations. *Astronomy & Astrophysics*, *438*(1), 31–37. doi:10.1051/0004-6361:20052658

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi:10.1109/MCSE.2007.55

Jones, E., Oliphant, T., Peterson, P., & others. (2001). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/

Kiran Mathew, A. F., Joseph H. Montoya. (2017). Atomate: A high-level interface to generate, execute, and analyze computational materials science workflows. *Computational Materials Science*, *139*, 140–152. doi:10.1016/j.commatsci.2017.07.030

Krogel, J. T. (2016). Nexus: A modular workflow management system for quantum simulation codes. *Computer Physics Communications*, *198*, 154–168. doi:10.1016/j.cpc.2015.08.012

Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., et al. (2017). The atomic simulation environment—a Python library for working with atoms. *Journal of Physics: Condensed Matter*, *29*(27), 273002. doi:10.1088/1361-648x/aa680e

McKinney, W. (2010). Data structures for statistical computing in python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 51–56).

Oliphant, T. E. (2006). *A guide to NumPy* (Vol. 1). Trelgol Publishing USA.

O'boyle, N. M., Tenderholt, A. L., & Langner, K. M. (2008). Cclib: A library for package-independent computational chemistry algorithms. *Journal of Computational Chemistry*, *29*(5), 839–845. doi:10.1002/jcc.20823

Pacifici, L., Verdicchio, M., Lago, N. F., Lombardi, A., & Costantini, A. (2013). A high-level ab initio study of the $N_2 + N_2$ reaction channel. *Journal of Computational Chemistry*, *34*(31), 2668–2676. doi:10.1002/jcc.23415

Schaftenaar, G., & Noordik, J. H. (2000). Molden: A pre-and post-processing program for molecular and electronic structures. *Journal of Computer-Aided Molecular Design*, *14*(2), 123–134. doi:10.1023/A:1008193805436

Schmidt, J., & Polik, W. (2013). WebMO enterprise, version 13.0. *WebMO LLC*. Retrieved from http://www.webmo.net

Schmidt, M. W., Baldridge, K. K., Boatz, J. A., Elbert, S. T., Gordon, M. S., Jensen, J. H., Koseki, S., et al. (1993). General atomic and molecular electronic structure system. *Journal of Computational Chemistry*, *14*(11), 1347–1363. doi:10.1002/jcc.540141112

Schuchardt, K. L., Didier, B. T., Elsethagen, T., Sun, L., Gurumoorthi, V., Chase, J., Li, J., et al. (2007). Basis set exchange: A community database for computational sciences. *Journal of Chemical Information and Modeling*, *47*(3), 1045–1052. doi:10.1021/ci600510j

Scott, A. P., & Radom, L. (1996). Harmonic vibrational frequencies: An evaluation of Hartree–Fock, Møller–Plesset, quadratic configuration interaction, density functional theory, and semiempirical scale factors. *The Journal of Physical Chemistry*, *100*(41), 16502–16513. doi:10.1021/jp960976r

Walt, S. van der, Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22–30. doi:10.1109/MCSE.2011.37