JCTC Journal of Chemical Theory and Computation

pubs.acs.org/JCTC Article

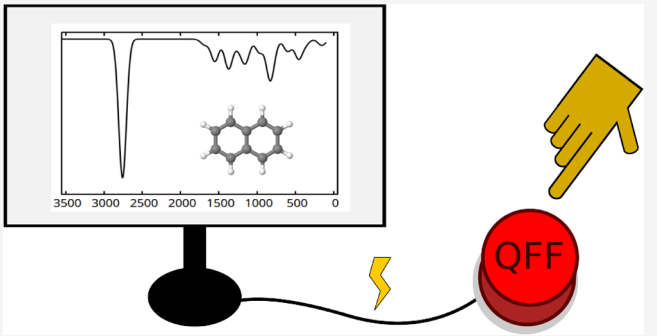
pbqff: Push-Button Quartic Force Fields

Brent R. Westbrook* and Ryan C. Fortenberry

Cite This: J. Chem. Theory Comput. 2023, 19, 2606−2615 Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: PBQFF is an open-source program for fully automating the production of quartic force fields (QFFs) and their corresponding anharmonic spectroscopic data. Rather than being a monolithic piece of code, it consists of several key modules including a generic interface to quantum chemistry codes and, notably, queuing systems; a molecular point group symmetry library; an internal-to-Cartesian coordinate conversion module; a module for the ordinary least-squares fitting of potential energy surfaces; and an improved second-order rotational and vibrational perturbation theory package for asymmetric and symmetric tops that handles type-1 and -2 Fermi resonances, Fermi resonance polyads, and Coriolis resonances. All of these pieces are written in Rust, a modern, safe, and performant programming language that has much to offer for scientific programming. This work introduces PBQFF and its surrounding ecosystem, in addition to reporting new anharmonic vibrational data for *c*-(C)C_3H_2 and describing how the components of PBQFF can be leveraged in other projects.



1. INTRODUCTION

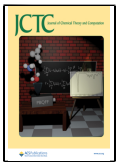
Quartic force fields (QFFs) are fourth-order Taylor series expansions of the internuclear potential energy portion of the Watson Hamiltonian,[1] typically written as shown in eq 1.

$$V = \frac{1}{2}\sum_{ij} F_{ij}\Delta_i\Delta_j + \frac{1}{6}\sum_{ijk} F_{ijk}\Delta_i\Delta_j\Delta_k + \frac{1}{24}\sum_{ijkl} F_{ijkl}\Delta_i\Delta_j\Delta_k\Delta_l \quad (1)$$

In line with this definition, QFFs are approximations to the potential function used to produce theoretical, anharmonic spectroscopic data including fundamental vibrational frequencies, vibrationally averaged principal rotational constants, and quartic and sextic distortion coefficients.[2,3] QFFs are often combined with both rotational and vibrational perturbation theory at second order (VPT2) to obtain these spectroscopic data,[4−8] but they can also provide the base potential energy surface (PES) for variational approaches such as vibration configuration interaction (VCI) or vibrational self-consistent field (VSCF)[9,10] after a Morse-cosine coordinate transformation.[11,12] A fairly simple input is required to generate all of these constants: an equilibrium molecular geometry, and the second-, third-, and fourth-order force constants (FCs), denoted *F* in eq 1. Simple, of course, does not mean easy, and the immense cost of QFFs comes in computing these FCs or derivatives, the number of which increases geometrically with the number of atoms in a molecule. Despite the great cost, QFFs are still much less expensive[10,13] than alternatives like global or even semiglobal PESs. These more extensive energy surfaces may require orders of magnitude more single-point energy computations, but these approaches may be required for loosely bound or "floppy" systems with PESs that are poorly approximated by a QFF. As a case in point, the C_4 semiglobal PES computed by Wang et al.[14] required 2914 single-point energy computations, while a recent VPT2 QFF study on the same molecule required only 233.[15] Clearly, the cost is reduced dramatically when the underlying electronic structure method has analytic derivatives available, but in most practical cases the derivatives are approximated numerically.

This numerical approximation comes in two major forms. In the first version, the single-point energies can be fit to a system of polynomial equations generated by the Taylor series expansion in eq 1 using ordinary least-squares. The FCs are, then, the coefficients that minimize the sum of squared residuals for the model. In the other formulation, the FCs are computed directly using central finite differences. The major benefit of the fitting approach is that, with the matrix form of the solution in hand, Newton's method can be used to find a stationary point of the potential energy function. This stationary point can be used both to verify that the input

Received: January 31, 2023
Published: April 20, 2023



https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

geometry is a minimum and to refit the FCs determined by the fitted function. This typically leads to small displacements from the input geometry, but this refitting is especially valuable when the QFF is composed of composite energies, for which a direct geometry optimization is unavailable.[12] On the other hand, the fitting process itself, and especially the iterative stationary point determination, can become expensive as the size of the molecule in question increases. The fact that the finite difference approach computes the force constants directly is a benefit in this regard. Additionally, if memory usage is a concern, the finite difference formulation can be adapted to accumulate each single-point energy directly into its corresponding FC, avoiding the overhead of storing a potentially large number of single-point energies.

Another fork in the procedure comes in the types of coordinates displaced to construct the QFF. Traditionally, the most efficient QFFs are performed in symmetry-internal coordinates (SICs). As their name suggests, SICs are linear combinations of simple internal coordinates such as bond stretches, bends, and torsions.[12] Their efficiency comes from the definition of SICs as nonmass-weighted approximations to normal coordinates. Despite not including the mass-weighting, well-chosen SICs typically align well with the actual normal coordinates, allowing the QFF to take advantage of the reduced coordinate space without the trouble of obtaining actual normal coordinates in advance. Of course, this reduced coordinate space refers to the reduction from $3N$ Cartesian coordinates to $3N - 6$ or $3N - 5$ internal coordinates.

However, SICs are not without their shortcomings. The computational efficiency comes at the cost of requiring the user to generate a set of unique coordinates that align well with the actual normal coordinates. As such, we have more recently[16] taken advantage of the simpler definition of Cartesian coordinates to run QFFs on molecules for which the development of an SIC system was too complicated. In this case, the equilibrium geometry is displaced directly along the Cartesian x, y, and z axes. Because of the much larger number of coordinates involved in Cartesian QFFs, they have been combined with the finite difference scheme of evaluating FCs. Together, these two factors increase the computational cost to about an order of magnitude more than SICs, even when molecular symmetry can be taken into account. For a concrete example, the three SICs required for the C_{2v} water molecule (symmetric and antisymmetric O−H stretches, and the bend) require 69 single-point energies to generate a QFF. Increasing the number of atoms to four in another C_{2v} molecule, formaldehyde, raises the number of single-point energy computations by nearly an order of magnitude to 413. Stepping to a C_{2v} five-atom system in cyclopropenylidene (c-C_3H_2) requires 1585 single-point energies.[17] Using $3N$ Cartesian coordinates instead of $3N - 6$ SICs increases the number of single-point energies in each case to 1780, 5252, and 11952 points, respectively. Still, when this computational cost can be paid, it can enable the application of QFFs to molecules for which SICs cannot readily be derived, notably for large molecules or those with non-Abelian point group symmetry.[16]

Regardless of the derivative scheme or coordinate type, when the single-point energies comprising a QFF are computed with a high-level quantum chemical method such as CCSD(T)-F12b[18−20] and triple-ζ basis sets, accuracy of within 5−7 cm^{-1} can be achieved for VPT2 based on such QFFs relative to gas-phase experimental values.[21−25] Similarly, experimental agreement to within about 60 MHz can be attained in the vibrationally averaged principal rotational constants.[26,27] Even better agreement (1−2 cm^{-1}; 7 MHz) can be reached through the use of more expensive, composite methods that include corrections like complete basis set extrapolations, corrections for core correlation, scalar relativistic effects, or higher-order electron correlation.[1,26] The QFF procedure itself is agnostic to these concerns and merely requires the generation of some set of single-point energies that can be turned into a local PES.

The sheer number of these single-point energies required for molecules larger than about five atoms means that these energies cannot practically be run serially within the confines of a conventional quantum chemistry program, necessitating some form of external automation.[10,28,29] Traditionally, our group has handled this issue by dividing each single-point energy computation into its own input file and submitting the individual computations to a queue managed by software like Slurm[30] or OpenPBS.[31] Again, for small molecules, the individual input files and corresponding queue submission scripts can readily be generated and submitted by scripts. However, as the number of single-point energies eclipses the number of jobs allowed (or desirable) in the queue, the process becomes more complicated, and the user has to monitor the available queue space and submit more jobs as it becomes available. Additionally, transient queue failures or errors in the computations may leave gaps in the completed single-point energies, requiring further manual intervention to detect and correct the affected files.

In addition to these difficulties, our traditional QFF procedure relies on several standalone programs written in Fortran 77. For SIC QFFs, the internal coordinate transformations are handled by the INTDER program,[32] the fitting of the PES is the domain of ANPASS,[33] and the final spectroscopic data are produced by SPECTRO.[5] Each connection between these programs (and in some places within a program) is mediated by the reading and writing of files to disk, which leads to a degradation in both performance and precision. Further, the brittle input formats for these programs mean that user errors in converting between them can cause silent and devastating errors. Some of this can be papered over with scripts that help to automate postprocessing, but no solutions to date have been totally satisfactory, especially for new users. As such, a program that can continuously monitor the status of the queue, collect output from the quantum chemistry program of choice, create new input files as needed, and finally compute the corresponding anharmonic rovibrational spectra would dramatically ease this process for the user.

2. THE PBQFF PROGRAM

These problems, (1) generating displaced geometries for QFFs, (2) building, running, and monitoring the corresponding single-point energy computations, and (3) collecting the results and transforming them into force constants and spectroscopic data, are what the program described herein and its subcomponents are designed to handle, with an additional emphasis on reproducibility and scientific rigor attained by ease of use. That program is PBQFF, and the name is an acronym for "push-button quartic force fields." As the name suggests, the goal of this program is to make the technology of QFFs easy to use and, thus, accessible to a wider audience of experts and nonexperts alike. In contrast to the aforementioned procedure, which often required hours of training and pages of

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

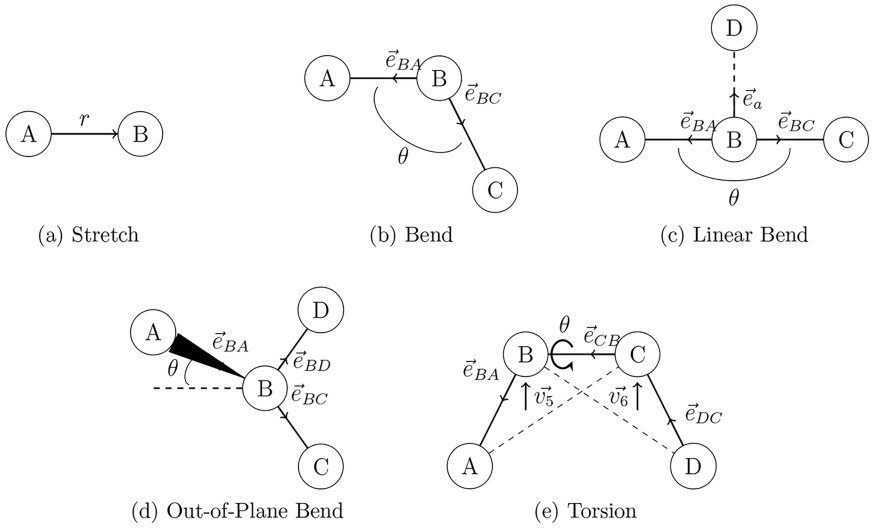


Figure 1. SICs supported by INTDER. A, B, C, and D denote real atoms, except in the case of (c), where D is a dummy atom.

notes to reproduce spectral data for a simple molecule like water, PBQFF dramatically streamlines the process. It allows the user to provide a simple configuration file written in Tom's Obvious Minimal Language (TOML),[34] literally push a button to initiate the QFF [in the optional graphical user interface (GUI)], and come back to anharmonic spectroscopic data. These data can then be passed to another simple tool that formats them for typesetting with LATEX. Underlying this functionality is a constellation of standalone libraries that can be readily composed to handle the problems above.

Rust has been selected as the language of choice for this suite of programs because of its strong dedication to safety and correctness at the level of language design, as well as its commitment to "blazingly fast" performance in line with other systems programming languages like C and C++.[35] Additionally, Rust offers many of the niceties of modern programming languages by default, such as unit and integration testing, benchmarking tools, an integrated package manager and build tool (named cargo), and extensive text editor support. The first two of these are invaluable in the production of scientific code where the importance of correctness and reproducibility are paramount. Rust ensures these features with a rigorous compiler that tracks memory ownership and lifetimes of variables to prevent common bugs in other languages like use-after-frees or dereferencing null pointers. These restrictions and the closely related careful accounting of the types that are safe to send between threads means that Rust code can truly be fearlessly parallelized. Further, cargo, the integrated package manager and build tool, can be used not only to facilitate easier builds, but also to achieve so-called "hermetic" builds where all versions of dependencies can be set explicitly to ensure reproducible build artifacts. On top of all of this, Rust is an expressive programming language, allowing for the direct translation of scientific and mathematical concepts and invariants into computer language, while preserving clean and easy-to-use application programming interfaces (APIs).

2.1. Normal Coordinates. PBQFF itself is both a binary and a library that addresses problem (1); namely, PBQFF is focused primarily on driving the coordinate transformations that convert displacements in the aforementioned coordinate systems into input for quantum chemistry programs. In addition to the SIC and Cartesian coordinate systems described above, PBQFF contains our first implementation of normal coordinate QFFs. This approach utilizes the infrastructure for Cartesian coordinate QFFs, but truncated to the second order, yielding a harmonic force field (HFF). This Cartesian HFF is used to generate the harmonic FC matrix, and the eigenvectors of the mass-weighted form of this matrix correspond to the normal coordinates for the molecule in question. These normal coordinates then serve as the basis for a full QFF, which captures the convenience of automatic coordinate determination from Cartesian coordinates and the computational efficiency of treating only $3N - 6$ internal coordinates.

For small molecules of about three or four atoms, the additional cost of the HFF is somewhat prohibitive, but for these molecules, SIC systems are typically straightforward anyway. Returning to the example of cyclopropenylidene, the initial HFF requires only 171 points compared to the 1585 points for the normal coordinate QFF, giving a ratio of about 11%. For a C_{2v}, nine-atom molecule $[CH_2(NH_2)_2]$[36] the ratio is 742 to 71769, or down to about 1%. Hence, the cost diminishes for the types of large molecules to which these coordinates will be applied. The one downside of using normal coordinates in this way is that the QFF is no longer fully isotope-independent. However, since this scheme preserves the full eigenvector matrix, including the translational and rotational degrees of freedom, isotope-independent Cartesian FCs can be recovered as described by Mackie et al.,[37] although this has not yet been implemented.

2.2. Symmetry and Internal Coordinates. The aforementioned normal and Cartesian coordinate schemes are implemented directly in PBQFF. In contrast, the SIC coordinates rely on a reimplementation of the INTDER program.[32] INTDER handles the transformation of SIC displacements to Cartesian coordinates for input to quantum chemistry (QC) programs and the corresponding transformation of SIC FCs to Cartesian coordinates for use in our VPT2 software. Figure 1 shows the simple-internal coordinates available in this version of INTDER. In all three coordinate types, PBQFF takes advantage of the SYMM package to account for molecular point group symmetry. SYMM provides an interface for obtaining the point group of a molecule, as well as the symmetries of displaced geometries within a given point group. This latter feature is especially

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

```
pub trait Program {
    fn filename(&self) -> String;
    fn outfile(&self) -> String;
    fn infile(&self) -> String;
    fn set_filename(&mut self, filename: &str);
    fn template(&self) -> &Template;
    fn extension(&self) -> String;
    fn charge(&self) -> isize;
    fn write_input(&mut self, proc: Procedure);
    fn read_output(filename: &str) -> Result<ProgramResult, ProgramError>;
    fn associated_files(&self) -> Vec<String>;
    fn new(filename: String, template: Template,
           charge: isize, geom: Geom) -> Self;
    fn build_jobs(moles: &Vec<Geom>, dir: &'static str, start_index: usize,
                  coeff: f64, job_num: usize, charge: isize, tmpl: Template)
                  -> Vec<Job<Self>> where Self: Sized;
}
```

Figure 2. Program interface.

```
pub trait SubQueue<P>
where
    P: Program + Clone + Serialize + for<'a> Deserialize<'a>,
{
    const SCRIPT_EXT: &'static str;
    fn dir(&self) -> &str;
    fn submit_command(&self) -> &str;
    fn chunk_size(&self) -> usize;
    fn job_limit(&self) -> usize;
    fn sleep_int(&self) -> usize;
    fn stat_cmd(&self) -> String;
    fn status(&self) -> HashSet<String>;
    fn no_del(&self) -> bool;
}

pub trait Queue<P>: SubQueue<P> + Submit<P>
where
    P: Program
        + Clone
        + Send
        + std::marker::Sync
        + Serialize
        + for<'a> Deserialize<'a>,
{
    fn write_submit_script(&self, infiles: &[String], filename: &str);
}
```

Figure 3. Queue interface.

useful for automatically determining symmetry-equivalent structures in order to eliminate redundant computations in a QFF. For SICs and fitted normal coordinates, this elimination is tightly coupled to the TAYLOR package, which implements the lazy Cartesian product algorithm presented by Thackston and Fortenberry[38] used to generate the matrices needed for the least-squares fitting and the corresponding geometrical displacements.

2.3. psqs. For actually interfacing with QC programs and cluster management or queuing systems, PBQFF relies on the PSQS library. This name is another acronym standing for "ProgramS and QueueS." PSQS is the primary handler of problem (2): building, running, and monitoring running QC programs, and the output gathering portion of problem (3). In addition to providing concrete implementations for the Molpro[39] and Mopac[40] QC programs and the OpenPBS[31] and Slurm[30] queuing systems, PSQS exposes interfaces for `Programs` and `Queues`, which allow users of the library to define their own types to be used with the core queue management functionality.

Figure 2 shows the set of methods required for a QC program to implement the Program trait (interface in many other languages). Nearly all of these are trivial to implement, and a default implementation of `build_jobs` is provided

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

```
fn drain(
    &self,
    dir: &str,
    jobs: Vec<Job<P>>,
    dst: &mut [f64],
    check_int: usize,
) -> Result<f64, ProgramError> where Self: std::marker::Sync;
```

Figure 4. Drain method.

that relies on the other, simpler methods. The main work is in `write_input` and `read_output`, which tell PSQS how to interact with the QC program.

Figure 3 shows the corresponding methods for a queuing system to implement the `Queue` trait. In this case, the methods are separated into two traits to allow more code to be reused between implementations. In particular, the `SubQueue` trait contains the methods that are independent of the generic `P: Program` bound in the trait definition. This trait can be implemented generically for all `P`, while the `Queue` trait itself may need to be implemented specifically for a given QC program. Regardless, most of these methods are trivial to implement and allow users of PSQS to take advantage of its abilities without having to modify PSQS itself.

Using these methods as building blocks, the `Queue` trait provides default implementations of methods that run, monitor, and collect output of geometry optimizations and single-point energy computations. It does this while monitoring the status of the queue (`status` method), dividing large numbers of jobs into smaller "chunks" that can be submitted to the queue together, and obeying the set `job_limit` to avoid overloading the queue. Figure 4 shows the signature of the `drain` method on `Queue`, which provides the interface for single-point energy computations. A user just needs to provide the directory in which to run the calculations, a vector of `Jobs` produced by the `build_jobs` method on a `Program`, and a destination array for the results. The `check_int` parameter represents the interval at which checkpoint files are to be written. These checkpoints allow the whole draining process to resume from any point if it gets interrupted. Furthermore, the methods for generating these checkpoints are composed entirely of calls to the other methods already described, meaning they are essentially free from the perspective of the programmer.

2.4. Anharmonic Spectra. Once the single-point energies have been computed and collected by PSQS, the second phase of problem (3) kicks in: transforming these energies to FCs and then to spectroscopic constants. In the case of the SICs and the fitted normal coordinates, there is an intermediate least-squares fitting (and refitting) step handled by a package called ANPASS, which is another reimplementation of the original Fortran version under the same name.[33] In the Cartesian and finite difference normal coordinate implementation, the FCs are, again, computed directly from the energies using finite differences. For SICs, in particular, there is the additional, aforementioned step of converting the SIC FCs to Cartesian coordinates via INTDER. In any case, the resulting Cartesian FCs are fed into the VPT2 and rotational perturbation theory subprogram, called SPECTRO. Like INTDER and ANPASS, this is also a reimplementation of the original Fortran version.[5] However, the SPECTRO implementation presented herein contains some additional niceties, such as fully automatic degenerate mode alignment for symmetric tops and JavaScript Object Notation (JSON) output for easy interface with postprocessing scripts. SPECTRO handles the rotational and vibrational perturbation theory that transform the FCs from the QFF into constants like harmonic and anharmonic vibrational frequencies, vibrationally averaged and singly vibrationally excited principal rotational constants, and quartic and sextic centrifugal distortion constants. It does all of this while handling individual type-1 and −2 Fermi resonances, their combination into Fermi resonance polyads[41] (for asymmetric tops), and Coriolis resonances. Accounting for all of these resonance effects has been shown to be crucial for obtaining accurate spectral data.[41,42]

2.5. Auxiliary Packages. In addition to these core packages for computing QFFs, several others take advantage of the modularity of these libraries to provide useful functionality. For example, the SUMMARIZE package uses the JSON serialization facilities of the SPECTRO package itself to collect multiple PBQFF/SPECTRO output files and compose them into tables in several formats including plain text, comma-separated value (CSV), JSON, Emacs org-mode, and LATEX. Similarly, the SCOPE package takes this JSON output and converts it to a format compatible with the Molden[43] molecule viewer for visualizing the vibrational modes computed by SPECTRO. Figure 5 shows the dependency graph between the core packages like PBQFF, PSQS, INTDER, and SPECTRO, as well as how these auxiliary packages depend on this core.

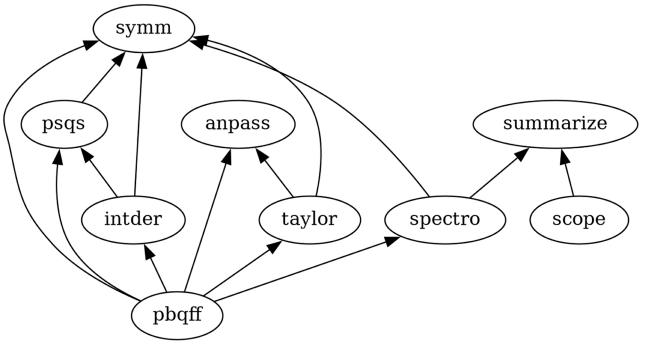


Figure 5. PBQFF ecosystem

The modularity of the code allows the individual pieces to be easily composed and reused, raising the level of abstraction at which the user has to think about QFFs. As noted for PBQFF itself, many of these packages exist as both a library and a standalone executable, lending composability to program authors as well as end users. For example, the executable forms of INTDER, ANPASS, and SPECTRO can be used as drop-in

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

```
geometry = """
O
H 1 OH
H 1 OH 2 HOH

OH = 1.0
HOH = 109.5
"""
optimize = true
charge = 0
step_size = 0.005
coord_type = "sic"
program = "mopac"
queue = "slurm"
sleep_int = 2
job_limit = 2048
chunk_size = 1
template = "scfcrt=1.D-21 aux(precision=14) pm6 threads=1"
check_int = 0
```

Figure 6. Example PBQFF input file.

replacements for their Fortran predecessors without the queuing features of PBQFF. On the other side, the primarily executable-focused SUMMARIZE can be loaded as a library for use in more specific data analysis programs. While the coordination provided by PBQFF is often the most convenient approach, this allows users with more established workflows to capture benefit from these packages as well.

2.6. Interface. As mentioned above, the primary interface for PBQFF is a single configuration file written in the TOML format. An example of such an input file for an SIC QFF on water is shown in Figure 6. This example uses the Slurm queuing system and the MOPAC QC program.

As an alternative to direct manipulation of this input file, a graphical interface called QFFBUDDY is also provided, as shown in Figure 7. In addition to enforcing inclusion of all required keywords, QFFBUDDY provides useful functionality like the loading of QC program templates and extracting optimized geometries from QC output files. Once the input has been assembled in this way, QFFBUDDY supports both generating a TOML configuration file as depicted above and running the QFF directly from the interface. With this latter feature, the user never has to interface directly with the underlying queuing system or even a terminal emulator.

2.7. Open source. Like many other chemistry programs including Psi4,[44] NWChem,[45] and GAMESS,[46] to name just a few, PBQFF and its component crates are fully open-source and publicly available on GitHub. Although contributions have thus far been limited to our group, we hope this publication will bring wider attention and usage to these programs. As such, we welcome bug reports, code contributions, feature requests, and integrations with other packages from the community as a whole. Links to all of the source code mentioned herein can be found in Table A1 of Appendix A.

3. ILLUSTRATIVE EXAMPLES

3.1. As an Executable. As an application of the PBQFF binary and the normal coordinate formulation therein, the vibrational frequencies of $c-(C)C_3H_2$ have been revisited. Previous work[22] reported positive anharmonicities in ν_7-ν_{10}, as well as ν_{12}, casting some doubt on the accuracy of the anharmonic treatment despite the high level of theory employed. As shown in Table 1, the present computations in normal coordinates appear to correct this issue. The CCSD(T)-F12b/cc-pVDZ-F12[18−20,47] level of theory is denoted F12-DZ therein, while TZ-cCR corresponds to the CCSD(T)-F12b/cc-pCVTZ-F12 level of theory, with additional corrections for scalar relativity.[26] While the earlier authors suggest that these positive anharmonicities are negligible, the new computations indicate that there may have been an underlying issue with their SIC definitions, which the automated normal coordinates in PBQFF have corrected. The only exceptions to this trend are in ν_{12} of the F12-DZ results and, surprisingly, in ν_{11} of the TZ-cCR results. The former is somewhat expected due to this being a pseudolinear bend, as indicated by the original authors.[22,48] The latter is more surprising, albeit smaller in magnitude, and may even suggest a lingering issue at the TZ-cCR level. Regardless, with an intensity of only 8 km mol^{-1}, this mode is less likely to play a role in experimental or astronomical observation of this molecule. Still, the change of coordinate systems leads to substantial differences in the anharmonic frequencies. In particular, ν_{10}, which was previously reported to have a double-harmonic intensity of 67 km mol^{-1}, shifts by over 100 cm^{-1} from 731.2 to 628.0 cm^{-1}. Such a large difference would surely inhibit experimental verification of the earlier results. In general, this highlights the reproducibility and accuracy gains of the PBQFF suite of programs compared to more manual and error-prone alternatives.

3.2. As a Library. Our recent work on the reparameterization of semiempirical methods[49] provides a perfect example of the composability of the PBQFF ecosystem. The essential problem in that work is to minimize the difference between experimental and computed anharmonic vibrational frequencies by adjusting the parameters in the PM6 semiempirical model.[50] The minimization algorithm of choice is the Levenberg−Marquardt algorithm,[51,52] which at its core necessitates the calculation of a Jacobian or gradient matrix describing how the frequencies change with respect to

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

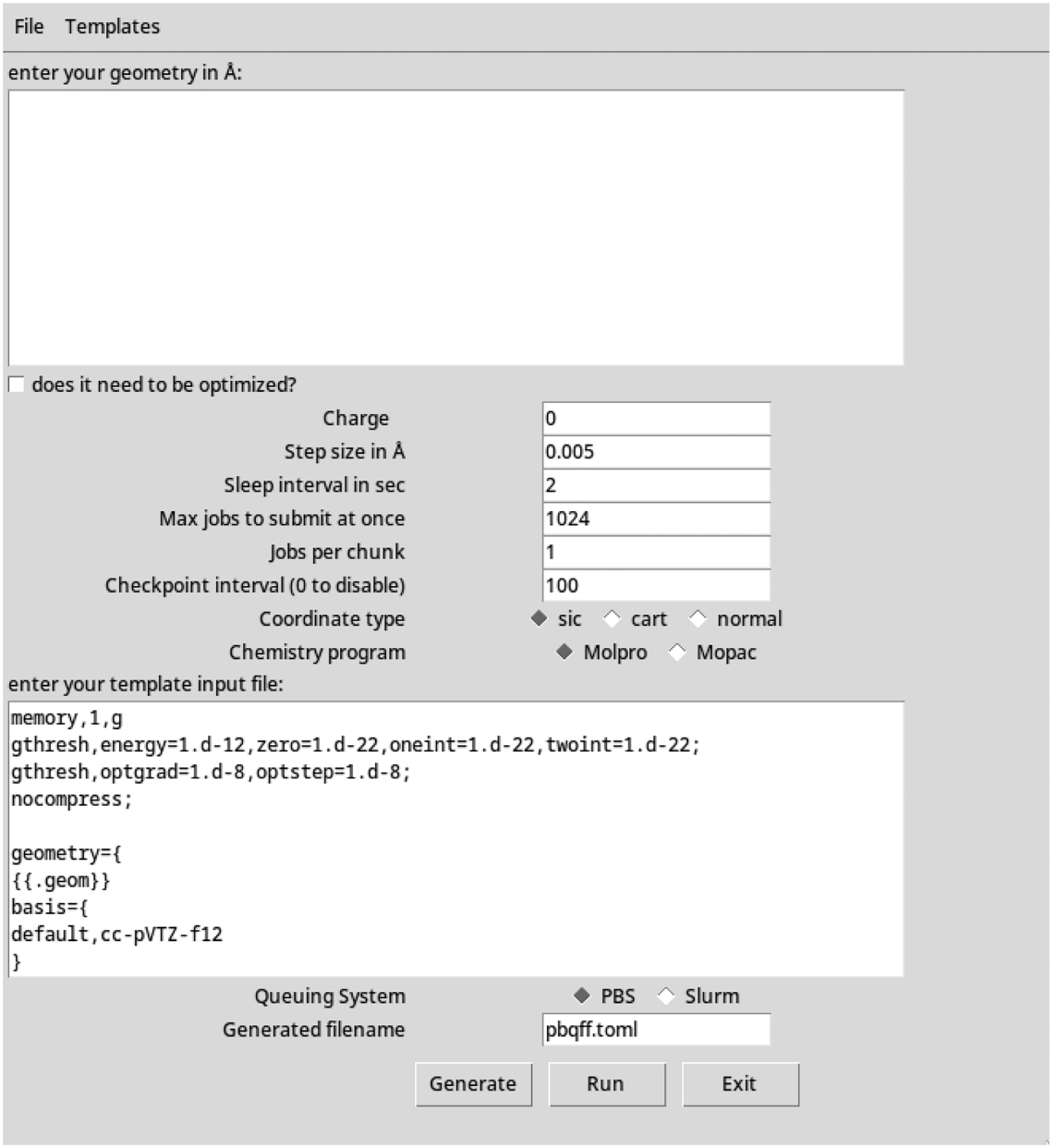


Figure 7. QFFBUDDY GUI.

variations in the parameters. This matrix can be assembled using finite differences of the anharmonic vibrational frequencies output by PBQFF, and the performance of the resulting parameters can similarly be assessed by another QFF run automatically through PBQFF.

While this is a somewhat naive approach, the fact that the current version of our code takes a better-parallelized approach is actually indicative of the flexibility of the PBQFF ecosystem. Namely, PBQFF and its constituent libraries allow for rapid prototyping of QFF-reliant programs (e.g., the naive approach above), while also exposing more primitive, but concomitantly more powerful, underlying functionality. When needed, this underlying functionality can be leveraged incrementally to tackle problems in a progressively more refined way. Further, when parallel computation is needed, as is often the case in modern high-performance computing, Rust carefully restricts the kinds of data and operations that can be performed across thread boundaries. While these restrictions may sound onerous at first, they help to prevent common issues that arise when doing parallel programming in other languages and are enforced automatically by the type system and the compiler. Additionally, when these constraints are obeyed, libraries like RAYON[53] provide a nearly seamless means for transforming sequential operations into parallel operations. RAYON is used in both PSQS and our semiempirical reparameterization code to process QC program output and run SPECTRO in parallel, respectively. Taking advantage of these improvements has allowed our reparameterization code to scale to optimizations requiring more than half a million single-point energies per Jacobian evaluation.

4. FUTURE WORK

Along the lines of the prototyping discussion in the previous section, a major path for future work is providing bindings to the functionality of PBQFF and its associated packages in a

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

Table 1. Vibrational Frequencies of c-(C)C_3H_2 (in cm^{-1})

Mode	Symm.	F12-DZ	TZ-cCR	Prev.[a]
ω_1	a_1	3342.5	3346.9	3353.9
ω_2	b_2	3294.0	3299.1	3307.4
ω_3	a_1	1826.3	1836.9	1841.7
ω_4	a_1	1564.8	1573.1	1584.7
ω_5	b_2	993.7	995.8	998.1
ω_6	a_1	975.2	974.9	973.2
ω_7	a_1	824.5	829.0	826.6
ω_8	a_2	779.5	786.6	793.6
ω_9	b_2	761.2	768.9	759.4
ω_{10}	b_1	633.0	638.5	645.4
ω_{11}	b_1	349.4	354.7	356.0
ω_{12}	b_2	156.4	161.7	157.2
ν_1	a_1	3203.1	3212.5	3214.1
ν_2	b_2	3160.4	3169.0	3171.8
ν_3	a_1	1807.5	1823.6	1826.8
ν_4	a_1	1545.5	1558.5	1564.0
ν_5	b_2	961.4	961.9	973.6
ν_6	a_1	949.5	947.4	941.6
ν_7	a_1	809.9	818.5	828.4
ν_8	a_2	744.9	743.3	816.3
ν_9	b_2	711.1	718.2	797.1
ν_{10}	b_1	611.9	628.0	731.2
ν_{11}	b_1	328.9	358.2	323.5
ν_{12}	b_2	172.0	125.9	195.1

[a]CcCR results from ref 22.

scripting language like Python. Such an integration would allow users to take advantage of the performance and safety guarantees of Rust while iterating more quickly on exploratory programs. To return to the reparameterization example, this would have allowed us to combine the convenient queue and program abstractions of PBQFF with the user-friendly linear algebra and optimization routines of NumPy[54] and SciPy,[55] while still likely transitioning to a full Rust implementation as efficiency became more important. Such bindings may also make it easier to integrate with other QC programs with Python interfaces, such as Psi4.[44]

Other routes of improvement include expanding PSQS to support more QC and queuing packages. Again, PSQS exposes the `Queue` and `Program` traits, which allow users to add their own implementations, but providing more concrete implementations out of the box would only make things easier for more users. PSQS could also be adapted to collect more information from QC output files. Thus far, it has been limited to optimized geometries and single-point energies, but it could be readily expanded to collect any data presented in the output files. Of particular interest may be analytic gradients or higher derivatives, which could also be leveraged within PBQFF, or properties like dipole moments, which could be used to construct dipole moment surfaces for the computation of anharmonic vibrational intensities.

5. CONCLUSION

Rust is an excellent language for developing safe, performant, and reusable scientific code. PBQFF and its related ecosystem of libraries and programs leverage the strengths of Rust to produce fully automated QFFs at massive scales while taking full advantage of all of the available, widely distributed computational resources. Further, several of these component libraries are useful in their own right, especially SPECTRO, which offers extensive Fermi and Coriolis resonance corrections for asymmetric and symmetric top molecules to increase the accuracy of its produced rovibrational spectral data. While much of this technology has been available for decades, it has never before been presented in such a reliable, composable, and accessible package for end users and programmers alike, as reflected by the examples reported herein. In many cases, it has also never been published as open source software, and we hope this publication will bring new users and contributors to the PBQFF ecosystem as a whole. Finally, an illustrative example for the c-(C)C_3H_2 molecule shows that the use of PBQFF can alleviate potential errors in QFF VPT2 computations, leading to corrections to several previously predicted fundamental frequencies for this molecule.

APPENDIX A: REPOSITORIES

This appendix contains Table A1 as mentioned in the text.

Table A1. Software Packages and Their git Repositories

Package	URL
PBQFF	https://github.com/ntBre/rust-pbqff
SYMM	https://github.com/ntBre/symm
PSQS	https://github.com/ntBre/psqs
ANPASS	https://github.com/ntBre/rust-anpass
SUMMARIZE	https://github.com/ntBre/summarize
INTDER	https://github.com/ntBre/intder
TAYLOR	https://github.com/ntBre/taylor
SPECTRO	https://github.com/ntBre/spectro
SCOPE	https://github.com/ntBre/scope

AUTHOR INFORMATION

Corresponding Author

Brent R. Westbrook − *Department of Chemistry & Biochemistry, University of Mississippi, University, Mississippi 38677-1848, United States;* orcid.org/0000-0002-6878-0192; Email: bwestbr2@go.olemiss.edu

Author

Ryan C. Fortenberry − *Department of Chemistry & Biochemistry, University of Mississippi, University, Mississippi 38677-1848, United States;* orcid.org/0000-0003-4716-8225

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.3c00129

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The present work is supported by NASA Grant NNX17AH15G and NSF Grant OIA-1757220. The Mississippi Center for Supercomputing Research (MCSR) provided the computational resources. The authors would also like to acknowledge the late Timothy J. Lee, Jan M. L. Martin of the Weizmann Institute of Science, and Xinchuan Huang of the SETI Institute for their contributions to the original SPECTRO code. Finally, the authors would like to thank Prof. Ashley Ringer McDonald of California Polytechnic State University-San Luis Obispo for help in formulating the manuscript.

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

REFERENCES

(1) Fortenberry, R. C.; Lee, T. J. Computational Vibrational Spectroscopy for the Detection of Molecules in Space. *Ann. Rep. Comput. Chem.* **2019**, *15*, 173−202.
(2) Watson, J. K. G. In *Vibrational Spectra and Structure*; During, J. R., Ed.; Elsevier: Amsterdam, 1977; pp 1−89.
(3) Papousek, D.; Aliev, M. R. *Molecular Vibration-Rotation Spectra*; Elsevier: Amsterdam, 1982.
(4) Willetts, A.; Handy, N. C.; Green, W. H.; Jayatilaka, D. Anharmonic Corrections to Vibrational Transition Intensities. *J. Phys. Chem.* **1990**, *94*, 5608−5616.
(5) Gaw, J. F.; Willets, A.; Green, W. H.; Handy, N. C. In *Advances in Molecular Vibrations and Collision Dynamics*; Bowman, J. M., Ratner, M. A., Eds.; JAI Press, Inc.: Greenwich, CT, 1991; pp 170−185.
(6) Barone, V. Anharmonic vibrational properties by a fully automated second-order perturbative approach. *J. Chem. Phys.* **2005**, *122*, 014108.
(7) Franke, P. R.; Stanton, J. F.; Douberly, G. E. How to VPT2: Accurate and Intuitive Simulations of CH Stretching Infrared Spectra Using VPT2+K with Large Effective Hamiltonian Resonance Treatments. *J. Phys. Chem. A* **2021**, *125*, 1301−1324.
(8) Yang, Q.; Bloino, J. An Effective and Automated Processing of Resonances in Vibrational Perturbation Theory Applied to Spectroscopy. *J. Phys. Chem. A* **2022**, *126*, 9276−9302.
(9) Bowman, J. M. Self-consistent field energies and wavefunctions for coupled oscillators. *J. Chem. Phys.* **1978**, *68*, 608.
(10) Puzzarini, C.; Bloino, J.; Tasinato, N.; Barone, V. Accuracy and Interpretability: The Devil and the Holy Grail. New Routes across Old Boundaries in Computational Spectroscopy. *Chem. Rev.* **2019**, *119*, 8131−8191.
(11) Fortenberry, R. C.; Huang, X.; Yachmenev, A.; Thiel, W.; Lee, T. J. On the Use of Quartic Force Fields in Variational Calculations. *Chem. Phys. Lett.* **2013**, *574*, 1−12.
(12) Fortenberry, R. C.; Lee, T. J. *Vibrational Dynamics of Molecules*; World Scientific: 2022; Chapter 7, pp 235−295.
(13) Yang, Q.; Mendolicchio, M.; Barone, V.; Bloino, J. Accuracy and Reliability in the Simulation of Vibrational Spectra: A Comprehensive Benchmark of Energies and Intensities Issuing From Generalized Vibrational Perturbation Theory to Second Order (GVPT2). *Front. Astron. Space Sci.* **2021**, DOI: 10.3389/fspas.2021.665232.
(14) Wang, X.; Huang, X.; Bowman, J. M.; Lee, T. J. Anharmonic Rovibrational Calculations of Singlet Cyclic C_4 Using a New ab initio Potential and a Quartic Force Field. *J. Chem. Phys.* **2013**, *139*, 224302.
(15) Sehring, C. M.; Palmer, C. Z.; Westbrook, B. R.; Fortenberry, R. C. The Spectral Features and Detectability of Small, Cyclic Silicon Carbide Clusters. *Front. Astron. Space Sci.* **2022**, DOI: 10.3389/fspas.2022.1074879.
(16) Westbrook, B. R.; Valencia, E. M.; Rushing, S. C.; Tschumper, G. S.; Fortenberry, R. C. Anharmonic Vibrational Frequencies of Ammonia Borane (BH_3NH_3). *J. Chem. Phys.* **2021**, *154*, 041104.
(17) Fortenberry, R. C.; Novak, C. M.; Layfield, J. P.; Matito, E.; Lee, T. J. Overcoming the Failure of Correlation for Out-of-Plane Motions in a Simple Aromatic: Rovibrational Quantum Chemical Analysis of *c*-C_3H_2. *J. Chem. Theory Comput.h* **2018**, *14*, 2155−2164.
(18) Raghavachari, K.; Trucks, G. W.; Pople, J. A.; Head-Gordon, M. A Fifth-Order Perturbation Comparison of Electron Correlation Theories. *Chem. Phys. Lett.* **1989**, *157*, 479−483.
(19) Adler, T. B.; Knizia, G.; Werner, H.-J. A Simple and Efficient CCSD(T)-F12 Approximation. *J. Chem. Phys.* **2007**, *127*, 221106.
(20) Knizia, G.; Adler, T. B.; Werner, H.-J. Simplified CCSD(T)-F12 Methods: Theory and Benchmarks. *J. Chem. Phys.* **2009**, *130*, 054104.
(21) Huang, X.; Valeev, E. F.; Lee, T. J. Comparison of One-Particle Basis Set Extrapolation to Explicitly Correlated Methods for the Calculation of Accurate Quartic Force Fields, Vibrational Frequencies, and Spectroscopic Constants: Application to H_2O, N_2H^+, NO_2^+, and C_2H_2. *J. Chem. Phys.* **2010**, *133*, 244108.
(22) Agbaglo, D.; Lee, T. J.; Thackston, R.; Fortenberry, R. C. A Small Molecule with PAH Vibrational Properties and a Detectable Rotational Spectrum: *c*-$(C)C_3H_2$, Cyclopropenylidenyl Carbene. *Astrophys. J.* **2019**, *871*, 236.
(23) Agbaglo, D.; Fortenberry, R. C. The Performance of CCSD(T)-F12/aug-cc-pVTZ for the Computation of Anharmonic Fundamental Vibrational Frequencies. *Int. J. Quantum Chem.* **2019**, *119*, No. e25899.
(24) Agbaglo, D.; Fortenberry, R. C. The Performance of Explicitly Correlated Wavefunctions [CCSD(T)-F12b] in the Computation of Anharmonic Vibrational Frequencies. *Chem. Phys. Lett.* **2019**, *734*, 136720.
(25) Westbrook, B. R.; Fortenberry, R. C. Anharmonic Frequencies of $(MO)_2$ & Related Hydrides for M = Mg, Al, Si, P, S, Ca, & Ti and Heuristics for Predicting Anharmonic Corrections of Inorganic Oxides. *J. Phys. Chem. A* **2020**, *124*, 3191−3204.
(26) Watrous, A. G.; Westbrook, B. R.; Fortenberry, R. C. F12-TZ-cCR: A Methodology for Faster and Still Highly-Accurate Quartic Force Fields. *J. Phys. Chem. A* **2021**, *125*, 10532−10540.
(27) Puzzarini, C.; Stanton, J. F. Connections between the accuracy of rotational constants and equilibrium molecular structures. *Phys. Chem. Chem. Phys.* **2023**, *25*, 1421.
(28) Smith, D. G. A.; Lolinco, A. T.; Glick, Z. L.; Lee, J.; Alenaizan, A.; Barnes, T. A.; Borca, C. H.; et al. Quantum Chemistry Common Driver and Databases (QCDB) and Quantum Chemistry Engine (QCEngine): Automation and interoperability among computational chemistry programs. *J. Chem. Phys.* **2021**, *155*, 204801.
(29) Bloino, J.; Barone, V. A Second-Order Perturbation Theory Route to Vibrational Averages and Transition Properties of Molecules: General Formulation and Application to Infrared and Vibrational Circular Dichroism Spectroscopies. *J. Chem. Phys.* **2012**, *136*, 124108.
(30) Yoo, A. B.; Jette, M. A.; Grondona, M. *SLURM: Simple Linux Utility for Resource Management. Job Scheduling Strategies for Parallel Processing*; Berlin, Heidelberg, 2003; pp 44−60.
(31) Allen, W. D. *Program INTDER*, Stanford University, Stanford, CA, 2005.
(32) Allen, W. D.; Császár, A. G.; Szalay, V.; Mills, I. M.; Horner, D. A. *INTDER 2005 is a General Program Written by W. D. Allen and Coworkers, which Performs Vibrational Analysis and Higher-Order Non-Linear Transformations*; 2005 (accessed 2021-09-15).
(33) Taylor, P. R. *ANPASS is a Program for Multi-Dimensional Polynomial Least-Squares Fitting and Stationary Point Determination*; 1982.
(34) *Tom's Obvious Minimal Language*. https://toml.io/en/ (accessed 2023-01-25.
(35) *Rust Programming Language*. https://www.rust-lang.org/ (accessed 2022-11-03).
(36) Marks, J. H.; Wang, J.; Fortenberry, R. C.; Kaiser, R. I. Preparation of Methanediamine $(CH_2(NH_2)_2)$ − A Precursor to Nucleobases in the Interstellar Medium. *Proc. Natl. Acad. Sci. U.S.A.* **2022**, *119*, No. e2217329119.
(37) Mackie, C. J.; Candian, A.; Huang, X.; Lee, T. J.; Tielens, A. G. G. M. Linear Transformation of Anharmonic Molecular Force Constants between Normal and Cartesian Coordinates. *J. Chem. Phys.* **2015**, *142*, 244107.
(38) Thackston, R.; Fortenberry, R. C. An Efficient Algorithm for the Determination of Force Constants and Displacements in Numerical Definitions of a Large, General Order Taylor Series Expansion. *J. Math. Chem.* **2018**, *56*, 103−119.
(39) Werner, H.-J.; Knowles, P. J.; Manby, F. R.; Black, J. A.; Doll, K.; Heßelmann, A.; Kats, D.; et al. *MOLPRO*, Version 2020.1, a Package of *ab Initio* Programs; 2020; see http://www.molpro.net.
(40) Stewart, J. J. P. *MOPAC2016*; Stewart Computational Chemistry: 2016.
(41) Martin, J. M. L.; Taylor, P. R. Accurate *ab Initio* Quartic Force Field for *trans*-HNNH and Treatment of Resonance Polyads. *Spectrochim. Acta, Part A* **1997**, *53*, 1039−1050.

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615

(42) Martin, J. M. L.; Lee, T. J.; Taylor, P. R.; François, J.-P. The Anharmonic Force Field of Ethylene, C_2H_4, by Means of Accurate *ab Initio* Calculations. *J. Chem. Phys.* **1995**, *103*, 2589−2602.
(43) Schaftenaar, G.; Noordik, J. H. Molden: A Pre- and Post-Processing Program for Molecular and Electronic Structures. *J. Comput.-Aided Mol. Design* **2000**, *14*, 123−134.
(44) Turney, J. M.; Simmonett, A. C.; Parrish, R. M.; Hohenstein, E. G.; Evangelista, F. A.; Fermann, J. T.; Mintz, B. J.; et al. PSI4: An Open-Source *Ab Initio* Electronic Structure Program. *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2012**, *2*, 556−565.
(45) Aprà, E.; Bylaska, E. J.; de Jong, W. A.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Valiev, M.; et al. NWChem: Past, present, and future. *J. Chem. Phys.* **2020**, *152*, 184102.
(46) Barca, G. M. J.; Bertoni, C.; Carrington, L.; Datta, D.; De Silva, N.; Deustua, J. E.; Fedorov, D. G.; et al. Recent developments in the general atomic and molecular electronic structure system. *J. Chem. Phys.* **2020**, *152*, 154102.
(47) Peterson, K. A.; Adler, T. B.; Werner, H.-J. Systematically Convergent Basis Sets for Explicitly Correlated Wavefunctions: The Atoms H, He, B-Ne, and Al-Ar. *J. Chem. Phys.* **2008**, *128*, 084102.
(48) Fortenberry, R. C.; Huang, X.; Francisco, J. S.; Crawford, T. D.; Lee, T. J. Quartic Force Field Predictions of the Fundamental Vibrational Frequencies and Spectroscopic Constants of the Cations $HOCO^+$ and $DOCO^+$. *J. Chem. Phys.* **2012**, *136*, 234309.
(49) Westbrook, B. R.; Layfield, J. P.; Lee, T. J.; Fortenberry, R. C. Reparameterized Semi-Empirical Methods for Computing Anharmonic Vibrational Frequencies of Multiply-Bonded Hydrocarbons. *Electronic Structure* **2022**, *4*, 045003.
(50) Stewart, J. J. P. Optimization of Parameters for Semiempirical Methods V: Modification of NDDO Approximations and Application to 70 Elements. *J. Mol. Model.* **2007**, *13*, 1173−1213.
(51) Levenberg, K. A. Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quart. Appl. Math.* **1944**, *2*, 164−168.
(52) Marquardt, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Indust. Appl. Math.* **1963**, *11*, 431−441.
(53) *Rayon: A data parallelism library for Rust*. https://github.com/rayon-rs/rayon (accessed 2023-01-25).
(54) Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357−362.
(55) Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17*, 261−272.

https://doi.org/10.1021/acs.jctc.3c00129
J. Chem. Theory Comput. 2023, 19, 2606−2615