Psi4 1.4: Open-Source Software for High-Throughput Quantum Chemistry

Daniel G. A. Smith,[1] Lori A. Burns,[2] Andrew C. Simmonett,[3] Robert M. Parrish,[2] Matthew C. Schieber,[2] Raimondas Galvelis,[4] Peter Kraus,[5] Holger Kruse,[6] Roberto Di Remigio,[7] Asem Alenaizan,[2] Andrew M. James,[8] Susi Lehtola,[9] Jonathon P. Misiewicz,[10] Maximilian Scheurer,[11] Robert A. Shaw,[12] Jeffrey B. Schriber,[2] Yi Xie,[2] Zachary L. Glick,[2] Dominic A. Sirianni,[2] Joseph Senan O'Brien,[2] Jonathan M. Waldrop,[13] Ashutosh Kumar,[8] Edward G. Hohenstein,[14] Benjamin P. Pritchard,[1] Bernard R. Brooks,[3] Henry F. Schaefer III,[10] Alexander Yu. Sokolov,[15] Konrad Patkowski,[13] A. Eugene DePrince III,[16] Uğur Bozkaya,[17] Rollin A. King,[18] Francesco A. Evangelista,[19] Justin M. Turney,[10] T. Daniel Crawford,[8, 1] and C. David Sherrill[2, a)]

[1)] *Molecular Sciences Software Institute, Blacksburg, Virginia 24061, USA*
[2)] *Center for Computational Molecular Science and Technology, School of Chemistry and Biochemistry, School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0400, United States*
[3)] *National Institutes of Health – National Heart, Lung and Blood Institute, Laboratory of Computational Biology, Bethesda, Maryland 20892, United States*
[4)] *Acellera Labs, C/Doctor Trueta 183, 08005 Barcelona, Spain*
[5)] *School of Molecular and Life Sciences, Curtin University, Kent St., Bentley, Perth, Western Australia 6102*
[6)] *Institute of Biophysics of the Czech Academy of Sciences, Královopolská 135, 612 65 Brno, Czech Republic*
[7)] *Department of Chemistry, Centre for Theoretical and Computational Chemistry, UiT, The Arctic University of Norway, N-9037 Tromsø, Norway*
[8)] *Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, United States*
[9)] *Department of Chemistry, University of Helsinki, P.O. Box 55 (A. I. Virtasen aukio 1), FI-00014 University of Helsinki, Finland*
[10)] *Center for Computational Quantum Chemistry, University of Georgia, Athens, Georgia 30602, United States*
[11)] *Interdisciplinary Center for Scientific Computing, Heidelberg University, D-69120 Heidelberg, Germany*
[12)] *ARC Centre of Excellence in Exciton Science, School of Science, RMIT University, Melbourne, VIC 3000, Australia*
[13)] *Department of Chemistry and Biochemistry, Auburn University, Auburn, Alabama 36849, United States*
[14)] *Stanford PULSE Institute, SLAC National Accelerator Laboratory, Menlo Park, California 94025, United States*
[15)] *Department of Chemistry and Biochemistry, The Ohio State University, Columbus, Ohio 43210, United States*
[16)] *Department of Chemistry and Biochemistry, Florida State University, Tallahassee, Florida 32306-4390, United States*
[17)] *Department of Chemistry, Hacettepe University, Ankara 06800, Turkey*
[18)] *Department of Chemistry, Bethel University, St. Paul, Minnesota 55112, United States*
[19)] *Department of Chemistry, Emory University, Atlanta, Georgia 30322, United States*

Psi4 is a free and open-source *ab initio* electronic structure program providing Hartree–Fock, density functional theory, many-body perturbation theory, configuration interaction, density cumulant theory, symmetry-adapted perturbation theory, and coupled-cluster theory. Most of the methods are quite efficient thanks to density fitting and multi-core parallelism. The program is a hybrid of C++ and Python, and calculations may be run with very simple text files or using the Python API, facilitating post-processing and complex workflows; method developers also have access to most of Psi4's core functionality via Python. Job specification may be passed using The Molecular Sciences Software Institute (MolSSI) QCSchema data format, facilitating interoperability. A rewrite of our top-level computation driver, and concomitant adoption of the MolSSI QCArchive Infrastructure project, make the latest version of Psi4 well suited to distributed computation of large numbers of independent tasks. The project has fostered the development of independent software components that may be reused in other quantum chemistry programs.

I. INTRODUCTION

a)Electronic mail: sherrill@gatech.edu

The Psi series of programs for quantum chemistry (QC) has undergone several major rewrites throughout its history. This is also true of the present ver-

sion, Psi4,[1] which bears little resemblance to its predecessor, Psi3. Whereas Psi3 was a *research code* aimed at providing a handful of high-accuracy methods for small molecules, Psi4 aims to be a user-friendly, general-purpose code suitable for fast, automated computations on molecules with up to hundreds of atoms. In particular, Psi4 has seen the introduction of efficient multi-core, density-fitted (DF) algorithms for Hartree–Fock (HF), density functional theory (DFT), symmetry-adapted perturbation theory (SAPT),[2,3] second- and third-order many-body perturbation theory (MP2, MP3), and coupled-cluster (CC) theory through perturbative triples [CCSD(T)].[4] While Psi3 was a stand-alone program that carried the assumption that QC computations were the final desired results and so offered few capabilities to interface with other program packages, Psi4 is designed to be part of a software ecosystem in which quantum results may only be intermediates in a more complex workflow. In Psi4, independent components accomplishing well-defined tasks are easily connected, and accessibility of key results through a Python interface has been emphasized.

Although the Psi project was first known as the BERKELEY package in the late 1970s, it was later renamed to reflect its geographical recentering alongside Henry F. Schaefer III to the University of Georgia. The code was ported to hardware-independent programming languages (Fortran and C) and UNIX in 1987 for Psi2; rewritten in an object-oriented language (C++), converted to free-format user input and flexible formatting of scratch files, and released under an open-source GPL-2.0 license in 1999 for Psi3;[5] reorganized around a programmer-friendly library for easy access to molecular integrals and related quantities, then unified into a single executable combining C++ for efficient QC kernels with Python for input parsing and for driver code in 2009 for Psi4;[6] and most recently, converted into a true Python module calling core C++ libraries, reorganized into an ecosystem with narrow data connections to external projects, opened to public development and open-source best practices, and relicensed as LGPL-3.0 to facilitate use with a greater variety of computational molecular sciences (CMS) software in 2017 for Psi4 v1.1.[1]

These rewrites have addressed challenges particular to quantum chemistry programs, including: (i) users want a fully featured program that can perform computations with the latest techniques; but (ii) QC methods are generally complex and difficult to implement; even more challenging, (iii) QC methods have a steep computational cost, and therefore must be implemented as efficiently as possible; yet this is a moving target as (iv) hardware is widely varied (e.g. from laptops to supercomputers) and frequently changing. We also note an emerging challenge: (v) thermochemical,[7] machine learning,[8] force-field fitting,[9] etc. applications can demand large numbers (10^5–10^8) of QC computations that may form part of complex workflows.

Psi4 has been designed with these challenges in mind.

For (i)–(iii), we have created a core set of libraries that are easy to program with and that provide some of the key functionalities required for modern QC techniques. These include the LibMints library that provides simple interfaces to compute one- and two-electron integrals, the DFHelper library to facilitate the computation and transformation of 3-index integrals for DF methods, and a library to build Coulomb and exchange (J and K) matrices in both the conventional and generalized forms that are needed in HF, DFT, SAPT, and other methods (see Refs. 1 and 6 and Sec. V B for more details). These libraries are also intended to address challenge (iv) above, as they have been written in a modular fashion so that alternative algorithms may be swapped in and out. For example, the LibMints library actually wraps lower-level integrals codes, and alternative integrals engines may be used as described in more detail in Sec. V G. Similarly, the object-oriented JK library is written to allow algorithms adapted for graphics processing units (GPU) or distributed-parallel computing. Challenge (v) is tackled by allowing computations via a direct application programming interface (API) and by encouraging machine-readable input and output.

The Psi4NumPy project[10] further simplifies challenge (ii), the implementation of new QC methods in Psi4. By making the core Psi4 libraries accessible through Python, it is now considerably easier to create pilot or reference implementations of new methods, since Python as a high-level language is easier to write, understand, and maintain than C++ code. Indeed, because the libraries themselves are written in efficient C++ code, a Python implementation of a new method is often sufficient as the final implementation as well, except in the cases that require manipulations of 3- or 4-index quantities that are not already handled by the efficient core Psi4 libraries. For reasons of readability, maintainability, and flexibility, the entire codebase is being migrated towards more top-level functions in Python.

Although the library design makes it easier for developers to add new methods into Psi4, we believe an even more powerful approach is to create a software ecosystem that facilitates the use of external software components. Our build system, driver, and distribution system have been rewritten specifically with this goal in mind, as discussed in Ref. 1 and Sec. VIII. The Python interface to Psi4 and the recently introduced ability to communicate via QCSchema further enhance this interoperability. Our recent moves to the more permissive LGPL-3.0 license and to fully open development on a public GitHub site (`https://github.com/psi4/psi4`) are also meant to foster this ecosystem.

Our recent infrastructure work since Ref. 1 is mainly focused on challenge (v), so that QC calculations can be routinely undertaken in bulk for use in various data analysis pipelines. As discussed in Sec. IV, Psi4 has reworked its driver layout to simplify nested post-processing calls and greatly promote parallelism and archiving. Python within Psi4's driver sets keywords ac-

cording to the molecular system and method requested, allowing straightforward input files. Additionally, Psi4 as a Python module (since v1.1, one can `import psi4`) means that codes may easily call Psi4 from Python to perform computations and receive the desired quantities directly via Python, either directly through the application programming interface (PsiAPI) or through JavaScript Object Notation (JSON) structured data.

Below, we present an overview of the capabilities of Psi4 (Sec. II). We then discuss the performance improvements in Psi4's core QC libraries (Sec. V), the expanding ecosystem of software components that can use or be used by Psi4 (Secs. VI and VII), and how the software driver has been rewritten to collect key quantities into a standard data format and to allow for parallel computation of independent tasks (Sec. IV).

II. CAPABILITIES

Psi4 provides a wide variety of electronic structure methods, either directly or through interfaces to external community libraries and plugins. Most of the code is threaded using OpenMP to run efficiently on multiple cores within a node.

Hartree–Fock and Kohn–Sham DFT. Conventional, integral-direct, Cholesky, and DF algorithms are implemented for self-consistent field (SCF) theory. Thanks to the interface with the LibXC library (see Sec. V A), nearly all popular functionals are available. The DF algorithms are particularly efficient, and computations on hundreds of atoms are routine. Energies and gradients are available for restricted and unrestricted Hartree–Fock and Kohn–Sham (RHF, RKS, UHF, UKS), and restricted open-shell Hartree–Fock (ROHF). RHF and UHF Hessians are available for both conventional and DF algorithms.

Perturbation Theory. Psi4 features Møller–Plesset perturbation theory up to fourth order. Both conventional and DF implementations are available for MP2, MP3, and MP2.5,[11] including gradients.[1,12,13] For very small molecules, the full configuration interaction (CI) code can be used[14,15] to generate arbitrary-order MPn and Z-averaged perturbation theory (ZAPTn)[16] results. Electron affinities and ionization potentials can now be computed through second-order electron-propagator theory (EP2)[17] and the extended Koopmans' theorem (EKT).[18–20]

Coupled-Cluster Theory. Psi4 supports conventional CC energies up to singles and doubles (CCSD) plus perturbative triples [i.e., CCSD(T)][4] for any single determinant reference and analytic gradients for RHF and UHF references.[5] For DF, energies and analytic gradients up to CCSD(T) are available for RHF references.[21–23] Cholesky decomposition CCSD and CCSD(T) energies[21] and conventional CC2[24] and CC3[25] energies are also available. To lower the computational cost of CC computations, Psi4 supports[26] approximations based on frozen natural orbitals (FNO)[27–30] that may be used to truncate the virtual space. Excited-state properties in Psi4 are supported with equation-of-motion CCSD[31,32] and the CC2 and CC3 approximations.[33] Linear-response properties, such as optical rotation,[34] are also available. Psi4 also supports additional CC methods through interfaces to the CCT3 (See Sec. VI C 6) and MRCC programs.[35]

Orbital-Optimized Correlation Methods. CC and Møller–Plesset perturbation methods are generally derived and implemented using the (pseudo)canonical Hartree–Fock orbitals. Choosing to instead use orbitals that minimize the energy of the targeted post-HF wavefunction has numerous advantages, including simpler analytic gradient expressions and improved accuracy in some cases. Psi4 supports a range of orbital-optimized methods, including MP2,[36] MP3,[37] MP2.5,[38] and linearized coupled-cluster doubles (LCCD).[39] DF energies and analytic gradients are available for all of these methods.[40–43]

Symmetry-Adapted Perturbation Theory. Psi4 features wavefunction-based SAPT through third-order to compute intermolecular interaction energies (IE), and leverages efficient, modern DF algorithms.[44–48] Psi4 also offers the ability to compute the zeroth-order SAPT (SAPT0) IE between open-shell molecules with either UHF or ROHF reference wavefunctions.[49–51] In addition to conventional SAPT truncations, Psi4 also features the atomic[52] and functional-group[53] partitions of SAPT0 (ASAPT0 and F-SAPT0, respectively), which partition SAPT0 IE and components into contributions from pairwise atomic or functional group contacts. Furthermore, Psi4 also offers the intramolecular formulation of SAPT0 (ISAPT0),[54] which can quantify the interaction between fragments of the same molecule as opposed to only separate molecules. The extensive use of core library functions for DF Coulomb and exchange matrix builds and integral transformations (see Section V B) has greatly accelerated the entire SAPT module in Psi4, with all SAPT0-level methods routinely deployable to systems of nearly 300 atoms (∼3500 basis functions); see also Secs. V C–V F for new SAPT functionality.

Configuration Interaction. Psi4 provides configuration interaction singles and doubles (CISD), quadratic CISD (QCISD),[55] and QCISD with perturbative triples [QCISD(T)][55] for RHF references. It also provides an implementation[56] of full configuration interaction (FCI) and the restricted active space configuration interaction (RASCI) approach.[57]

Multi-reference Methods. Psi4 provides conventional and DF implementations of complete-active-space SCF (CASSCF)[58,59] and restricted-active-space SCF (RASSCF).[60] Through the CheMPS2 code, density-matrix renormalization group (DMRG)[61,62] based CASSCF[63] and CASSCF plus second-order perturbation theory (CASPT2)[64] are available. The state-specific multireference CC method of Mukherjee and coworkers (Mk-MRCC) is implemented in Psi4 with singles, doubles, and perturbative triples.[65] A complementary

4

second-order perturbation theory based on the same formalism (Mk-MRPT2) also exists.[66] Psi4 can perform multireference CC calculations through an interface to the MRCC program of Kállay and coworkers,[35,67] where high-order excitations (up to sextuples) as well as perturbative methods are supported. Additional methods for strong correlation are available through the Forte[68–70] and v2rdm_casscf[71] (See Sec. VI C 5) plugins.

Density Cumulant Theory. Psi4 offers the reference implementation of Density Cumulant Theory (DCT), which describes electron correlation using the cumulant of the two-electron reduced density matrix (RDM) instead of a many-electron wave-function.[72] Psi4 includes an implementation[73] of the original DCT formulation,[72] a version with an improved description of the one-particle density matrix (DC-12),[74] their orbital-optimized variants (ODC-06 and ODC-12),[75] and more sophisticated versions that include N-representability conditions and three-particle correlation effects [ODC-13 and ODC-13(λ_3)].[76] In particular, ODC-12 maintains CCSD scaling but is much more tolerant of open-shell character and mild static correlation.[77,78] Analytic gradients are available for DC-06, ODC-06, ODC-12, and ODC-13 methods.[75,76,79]

Relativistic Corrections. Psi4 can perform electronic structure computations with scalar relativistic corrections either by calling the external DKH library for up to fourth-order Douglas-Kroll-Hess (DKH)[80,81] or by utilizing the the exact-two-component (X2C)[82–92] approach to supplement the one-electron Hamiltonian of a non-relativistic theory for relativistic effects. At present, only the point nuclear model is supported.

Automated Composite and Many-Body Computations. Psi4 provides a simple and powerful user interface to automate multi-component computations, including focal-point[93–95] approximations, complete-basis-set (CBS) extrapolation, basis set superposition corrections,[96–98] and many-body expansion (MBE) treatments of molecular clusters. These capabilities can all be combined to obtain energies, gradients, or Hessians, as discussed below in Sec. IV. For example, one can perform an optimization of a molecular cluster using focal-point gradients combining MP2/CBS estimates with CCSD(T) corrections computed in a smaller basis set, with counterpoise corrections. The MBE code allows for different levels of theory for different terms in the expansion (monomers, dimers, trimers, etc.) and also supports electrostatic embedding with point charges.

III. PSIAPI

Introduced in v1.1,[1] the Psi4 API (PsiAPI) enables deployment within custom Python workflows for a variety of applications, including quantum computing and machine learning, by making Psi4 a Python module (i.e., `import psi4`). Using Psi4 in this manner is no more difficult than writing a standard Psi4 input file, as shown

in the middle and left panels of Fig. 1, respectively. The true power of PsiAPI lies in the user's access to Psi4's core C++ libraries and data structures directly within the Python layer. PsiAPI thereby can be used to, e.g., combine highly optimized computational kernels for constructing Coulomb and exchange matrices from HF theory with syntactically intuitive and verbose Python array manipulation and linear algebra libraries like NumPy.[99] An example of PsiAPI for rapid prototyping is given in Sec. VI 1.

A. Psi4NumPy

Among the most well-developed examples of the advantages afforded by the direct Python-based PsiAPI is the Psi4NumPy project,[10] whose goal is to provide three services to the CMS community at large: (i) to furnish reference implementations of computational chemistry methods for the purpose of *validation and reproducibility*, (ii) to lower the barrier between theory and implementation by offering a *framework for rapid prototyping* where new methods could be easily developed, and (iii) to provide *educational materials* which introduce new practitioners to the myriad of practical considerations relevant to the implementation of quantum chemical methods. Psi4NumPy accomplishes these goals through its publicly available and open-source GitHub repository,[100] containing both reference implementations and interactive tutorials for many of the most common quantum chemical methods, such as HF, Møller–Plesset perturbation theory, CC, CI, and SAPT. Furthermore, since its publication in 2018, 17 separate projects to date have leveraged the Psi4NumPy framework to facilitate their development of novel quantum chemical methods.[101–117] Finally, Psi4NumPy is a thoroughly community-driven project; interested readers are highly encouraged to visit the repository[100] for the latest version of Psi4NumPy and to participate in "pull request" code review, issue tracking, or by contributing code to the project itself.

B. Jupyter Notebooks

Inspired by notebook interfaces to proprietary computer algebra systems (e.g., Mathematica and Maple), a Jupyter notebook is an open-source web application that allows users to create and share documents containing executable code, equations, visualizations, and text.[118] Jupyter notebooks are designed to support all stages of scientific computing, from the exploration of data to the creation of a detailed record for publishing. Leveraging Psi4 within this interface, therefore, provides *interactive* access to Psi4's data structures and functionality. Visualization and analysis of properties such as geometry and orbitals can be facilitated with tools available within The Molecular Sciences Software Institute's[119] (MolSSI) QCArchive[120,121] project. Additionally, the
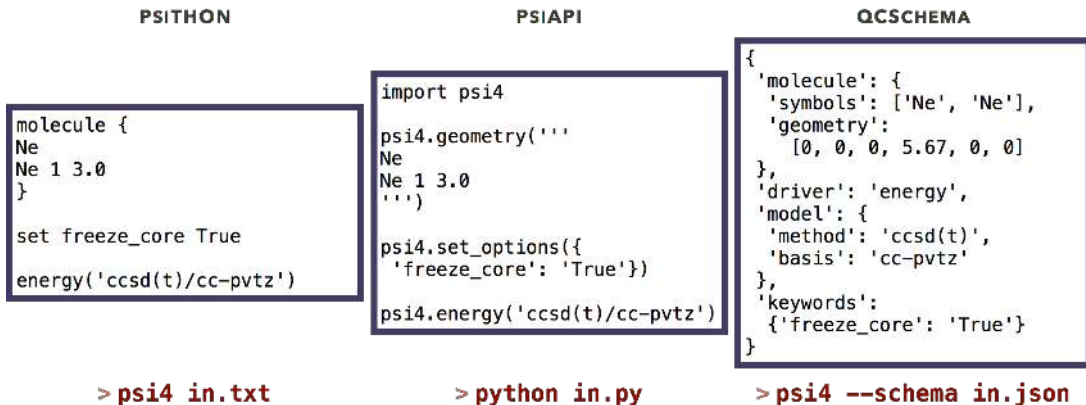
FIG. 1. Input modes for Psi4. A coupled-cluster calculation is run equivalently through its preprocessed text input language (PSIthon; left), through the Python API (PsiAPI; middle), and through structured JSON input (QCSchema; right).

unique combination of executable code cells, equations, and text makes Jupyter notebooks the perfect environment for the development and deployment of interactive educational materials, as illustrated by the Psi4NumPy and Psi4Education[122] projects, or for living supplementary materials for publications.[123,124]

IV. TASK-BASED DISTRIBUTED DRIVER

The *recursive driver* introduced in 2016 for Psi4 v1.0 to reorganize the outermost user-facing functions into a declarative interface has been refactored for Psi4 v1.4 into the *distributed driver* which emphasizes high-throughput readiness and discretized communication through schema. In the earlier approach, the user employed one of a few driver functions [`energy()`, `gradient()`, `optimize()`, `hessian()`, `frequency()`], and everything else was handled either by the driver behind the scenes (e.g., selecting analytic or finite-difference derivatives) or through keywords (e.g., `"mp2/cc-pv[t,q]z"`, `bsse_type="cp"`, `dertype="energy"`). When a user requested a composite computation that requires many individual computations (for example, a gradient calculation of a basis-set extrapolated method on a dimer with counterpoise correction), internal logic directed this into a handler function (one each for many-body expansion, finite difference derivatives, and composite methods like basis-set extrapolations and focal-point approximations) which broke the calculation into parts; then each part re-entered the original function, where it could be directed to the next applicable handler (hence, a "recursive driver"). At last, the handlers called the function on an analytic task on a single chemical system, at which point actual QC code would be launched. However, the code to implement this functionality was complex and not easily extendable to the nested parallelism (among many-body, finite-difference, and composite) to which these computations are naturally suited. Because of these limitations, the

internal structure of the driver has been reorganized so that all necessary QC input representations are formed before any calculations are run.

The motivation for the driver refactorization has been the shift toward task-based computing and particularly integration with the MolSSI QCArchive[120,121] project to run, store, and analyze QC computations at scale. The QCArchive software stack, collectively QCArchive Infrastructure, consists of several building blocks: QCSchema[125] for JSON representations of QC objects, job input, and job output; QCElemental[126] for Python models (constructors and helper functions) for QCSchema as well as fundamental physical constants and periodic table data; QCEngine[127] for compute configuration (e.g., memory, nodes) and QCSchema adaptors for QC programs; and QCFractal[128] for batch compute setup, compute management, storage, and query.

Psi4 v1.1 introduced a `psi4 --json input` mode that took in a data structure of molecular coordinates, driver, method, and keywords strings and returned a JSON structure with the requested driver quantity (energy, gradient, or Hessian), a success boolean, QCVariables (a map of tightly defined strings such as `CCSD CORRELATION ENERGY` or `HF DIPOLE` to float or array quantities), and string output. Since then, QC community input under MolSSI guidance has reshaped that early JSON into the current QCSchema `AtomicInput` model capable of representing most non-composite computations. ("Atomic" here refers not to atom vs. molecule but to single energy/derivative on a single molecule vs. multistage computations.) Psi4 v1.4 is fully capable of being directed by and emitting the MolSSI QCSchema v1 (see figure 1, right) via `psi4 --schema input` or `psi4.run_qcschema(input)`, where `input` is a Python dictionary, JSON text, or binary MessagePacked structure of NumPy arrays and other fields. Since Psi4 speaks QCSchema natively, its adaptor in QCEngine is light, consisting mostly of adaptations for older versions of Psi4 and of schema hotfixes. Several other QC packages without QCSchema input/ouput (I/O) have more exten-

sive QCENGINE adaptors that construct input files from `AtomicInput` and parse output files into `AtomicResult` (discussed below). The distributed driver is designed to communicate through QCSCHEMA and QCENGINE so that the driver is independent of the community adoption of QCSCHEMA.

The `AtomicInput` data structure includes molecule, driver function name, method and basis set (together, "model"), and keyword dictionary, while the output data structure `AtomicResult` additionally includes the primary return scalar or array, any applicable of a fixed set of QCSCHEMA properties, as well as PSI4 specialties like QCVariables. Importantly, the customary output file is included in the returned schema from a PSI4 computation. The driver has been revamped to use the `AtomicInput` and `AtomicResult` structures as the communication unit. In order for the abovementioned handler procedures (now "Computer" objects) of the PSI4 driver to communicate, specialized schemas that are supersets of `AtomicResult` have been developed. New fields have been introduced, including `bsse_type` and `max_nbody` for `ManyBodyComputer`; `stencil_size` and `displacement_space` for `FiniteDifferenceComputer`; `scheme` and `stage` for `CompositeComputer`; and `degeneracy` and `theta_vib` for the vibrational procedure. These contents are being optimized for practical use in PSI4 and have been or will be submitted to MolSSI QCSCHEMA and QCELEMENTAL for community input and review. A recently official schema already implemented in PSI4 is for wavefunction data and encodes orbital coefficients, occupations, and other information in standard CCA format.[129] This new schema is supported by native PSI4 infrastructure to permit serialization and deserialization of PSI4's internal `Wavefunction` class that contains more fields than the schema stores. Although not yet used for communication, PSI4 can also emit `BasisSet` schema. The layered procedures of the distributed driver involve sums of potentially up to thousands of schema-encoded results and are thus susceptible to numerical noise that a pure-binary data exchange would avoid. Nominally, JSON does not serialize NUMPY arrays or binary `float`s. However, the QCELEMENTAL/QCSCHEMA models support extended serialization through MESSAGEPACK[130] so that NUMPY arrays[99] can be transparently and losslessly moved through the distributed driver.

The task-oriented strategy for the distributed driver is illustrated in figure 2. The user interface with the customary driver functions, figure 2(a), remains unchanged. If a single analytic computation is requested, it proceeds directly into the core QC code of PSI4 (leftmost arrow), but if any of the handlers are requested, the driver diverts into successively running the "planning" function of each prescribed procedure (figure 2(b) with details in (z)) until a pool of analytic single-method, single-molecule jobs in QCSCHEMA `AtomicInput` format is accumulated. Although these could be run internally through the API counterpart of `psi4 --schema`

(figure 2(c.i)), PSI4 executes through QCENGINE so that other programs can be executed in place of PSI4 if desired (figure 2(c.ii)). An additional strategic benefit of running through QCENGINE is that the job pool can be run through QCFRACTAL (figure 2(c.iv)), allowing simultaneous execution of all jobs on a cluster or taking advantage of milder parallellism on a laptop, just by turning on the interface (∼5 additional Python lines). The database storage and QCSCHEMA indexing inherent to QCFRACTAL means that individual jobs are accessible after completion; if execution is interrupted and restarted, completed tasks are recognized, resulting in effectively free coarse-grained checkpointing. Alternatively, for the mild boost of single-node parallelism without the need to launch a QCFRACTAL database, one can run in "snowflake" mode (figure 2(c.iii)) which employs all of QCFRACTAL's job orchestration, indexing, and querying technology, except the internal database vanishes in the end. The use of these modes in input is shown in figure 3. When all jobs in the pool are complete (all QCSCHEMA `AtomicResult` are present), the "assemble" functions of each procedure are run in reverse order of invocation (figure 2(d) with details in figure 2(z)). That is, model chemistry energies are combined into composite energies by the `CompositeComputer`, then composite energies at different displacements are combined into a gradient by the `FiniteDifferenceComputer`, then gradients for different molecular subsystems and basis sets are combined into a counterpoise-corrected gradient by the `ManyBodyComputer`, and finally, the desired energy, gradient, or Hessian is returned, figure 2(e). The schema returned by driver execution has the same apparent (outermost) structure as a simple `AtomicResult` with a molecule, return result, properties, and provenance, so it is ready to use by other software expecting a gradient (like a geometry optimizer). However, each procedure layer returns its own metadata and the contributing QC jobs in a specialized schema, which is presently informal, so that the final returned JSON document is self-contained. Ensuring maintainability by merging code routes was given high priority in the distributed driver redesign: parallel and serial executions take the same routes, intra-project (API) and inter-project communications use the same QCSCHEMA medium, and (in a future revision) a generic QC driver calling PSI4 can proceed through QCSCHEMA.

V. NEW FEATURES AND PERFORMANCE IMPROVEMENTS

A. DFT

The DFT module now uses LIBXC[131] to evaluate the exchange-correlation terms. PSI4 thus has access to 400+ functionals, of which ∼100 are routinely tested against other implementations. Modern functionals, such as ωB97M-V[132] and the SCAN family,[133] are now avail-
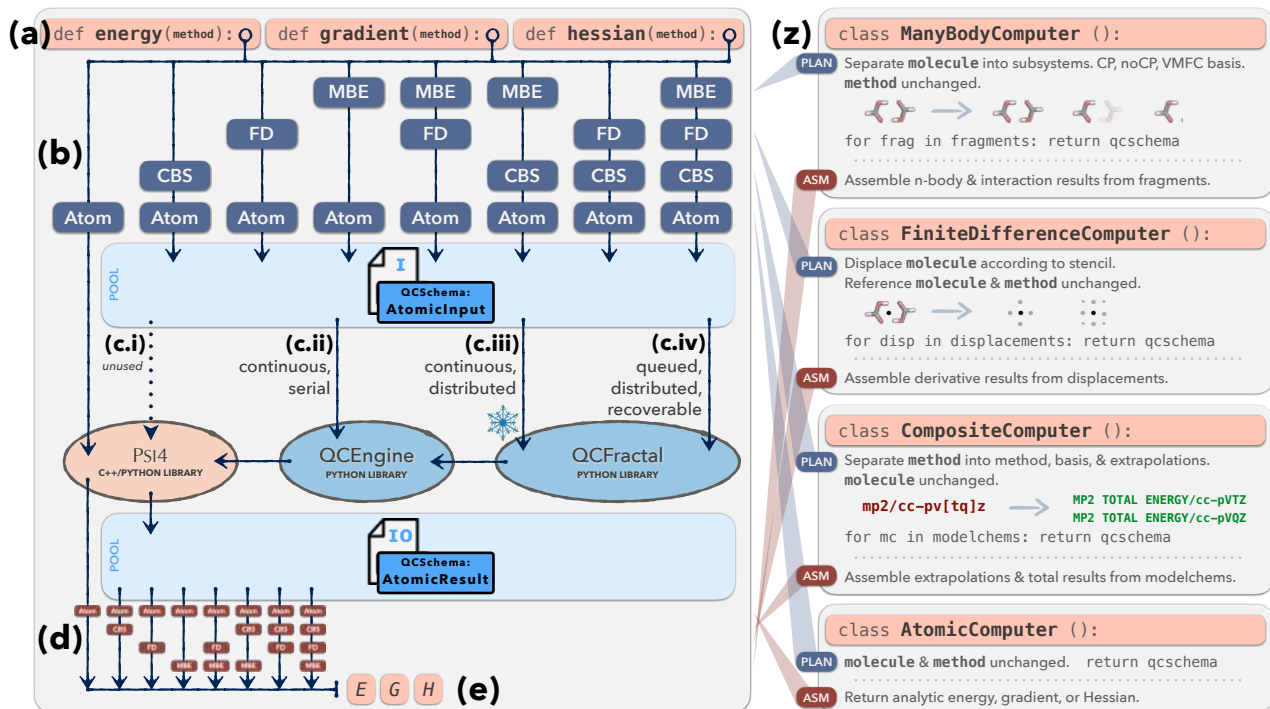
FIG. 2. Structure of the distributed driver: consult the final paragraph of Sec. IV for details. Briefly, a user request (a) for a multi-molecule, multi-model-chemistry, or non-analytic derivative passes into planning functions (b) that generate a pool of QCSchema for single-molecule, single-model-chemistry, analytic derivative inputs. These can run in several modes (c), depending on desired parallelism and recoverability. Completed QCSchema pass through assembly functions (d) that reconstitute into the requested energy, gradient, or Hessian (e).

able. Support for hybrid LDA functionals, pending their release in a stable version of LibXC, is also implemented. The new functional interface is Python-dictionary-based and uses LibXC-provided parameters where possible. Additional capabilities for dispersion-inclusive, tuned range-separated, and double-hybrid functionals are defined atop LibXC fundamentals. The interface also allows users to easily specify custom functionals, with tests and examples provided in the documentation.

The DFT module in Psi4 v1.4 is significantly faster than the one in Psi4 v1.1, both in single-threaded and multi-threaded use cases. Recent versions are compared in figure 4, showing the speed improvements for the adenine·thymine (A·T) stacked dimer from the S22 database.[135] With ωB97X-D/def2-SVPD (figure 4, upper), this test case corresponds to 234 and 240 basis functions for each monomer and 474 for the dimer, while the problem size is approximately doubled in B3LYP-D3(BJ)/def2-TZVPD (figure 4, lower).

Much of the speed improvement is due to improved handling of the DFT grids. Collocation matrices between basis functions and the DFT grid are now formed by an optimized library (Gau2Grid; Sec. VI B 3) and are automatically cached if sufficient memory is available, thus removing the need for their re-computation in every iteration. The whole module, including the generation of quadrature grids and collocation matrices, is now effi-

ciently parallelized. The overall speedup between v1.1 and v1.4 is $1.9\times$ on a single core. Notable speedups are obtained for range-separated functionals (e.g., the ωB97X-D functional, see figure 4, upper), as the MemD-FJK algorithm is now implemented for this class of methods (see Sec. V B).

As of Psi4 v1.4, grid screening based on exchange-correlation weights is applied with a conservative default cutoff of 10^{-15}. Grid pruning schemes are also implemented, the default `robust` scheme removing $\sim$30% of the grid points. Grid pruning on its own is responsible for a $1.3\times$ single-core speedup in the case of A·T dimer with B3LYP-D3(BJ)/def2-TZVPD. However, a loss of accuracy can be expected in the pruning of smaller grids (<0.1 kcal mol^{-1} for IE in the A24 database[136]).

B. MemDFJK Algorithm

The SCF Coulomb (J) and exchange (K) builds are the cornerstone of all SCF-level operations in Psi4, such as SCF iterations, MP2 gradients, SAPT induction terms, SCF response, time-dependent DFT (TDDFT) and more. Over the past decade, the raw floating point operations per second (FLOPS) ability of modern central processing units (CPU) has grown much faster than the speed of memory I/O, which can lead to memory I/O

```
import psi4
from qcfractal import FractalSnowflake
from qcfractal import FractalServer

client = None                    # Fig. 2(c.ii)
server = FractalSnowflake()      # Fig. 2(c.iii)
server = FractalServer()         # Fig. 2(c.iv)
client = server.client()

dimer = psi4.geometry("""
  He
  --
  He 1 4.0
""")

plan = psi4.gradient("HF/cc-pV[DT]Z",
                     bsse_type="vmfc",
                     molecule=dimer,
                     return_plan=True)
plan.compute(client)

server.await_results()
# re-run file after jobs complete for final processing

qcsk = plan.get_results(client)
print(qcsk.return_result)  # vmfc gradient


############################################################

plan = psi4.gradient("HF/cc-pV[DT]Z",
                     bsse_type=["cp", "nocp"],
                     molecule=dimer,
                     return_plan=True)
plan.compute(client)  # free! calcs in database

qcsk = plan.get_results(client)
print(qcsk.return_result)  # cp gradient
```

FIG. 3. Input file illustrating a CBS and many-body gradient run through the distributed driver in continuous mode (white-background lines; figure 2(c.ii)), distributed mode with `FractalSnowflake` (figure 2(c.iii); additional blue-background lines), and distributed mode with the full storage and queuing power of QCFRACTAL (figure 2(c.iv); additional red-background lines). The lower example is "free" when using QCFRACTAL since the components required for BSSE corrections have already been computed during the upper VMFC. While this example exposes the returned QCSCHEMA `AtomicResult`, the traditional syntax of `grad = psi4.gradient("HF/cc-pV[DT]Z", bsse_type="vmfc")` runs in mode (c.ii) and is identical to the upper example.
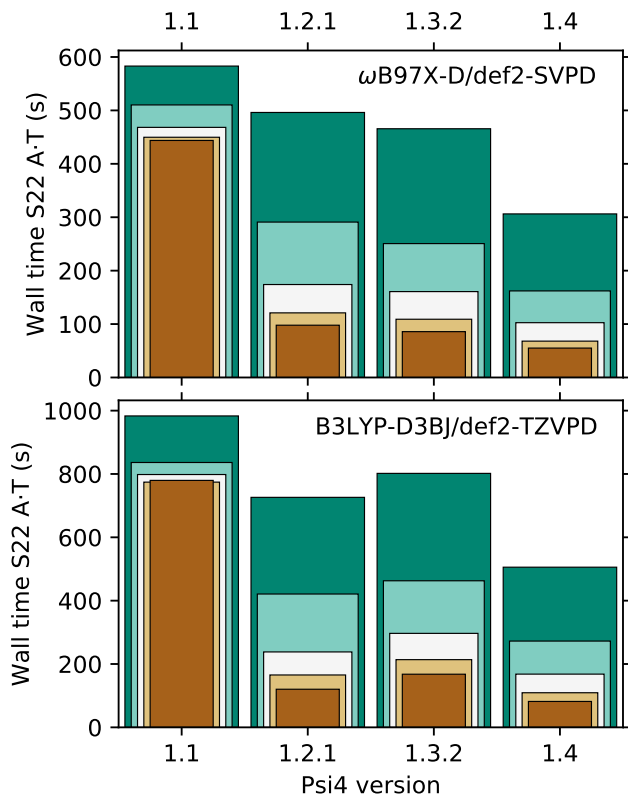

FIG. 4. Wall-time comparison for interaction energy of adenine·thymine stacked dimer from the S22 database with various versions of PSI4 using 1 (green) to 16 (brown) threads.[134] PSI4 v1.4 data obtained with `robust` grid pruning algorithm.

rather than raw FLOPS limiting operations. A large data copy quickly became the bottleneck of the PSI4 v1.1 JK algorithm, especially when running on many concurrent cores.

Examining the canonical K equations with DF shows the following (using the Einstein summation convention):

$$D_{\lambda\sigma} = C_{i\sigma}C_{i\lambda}, \tag{1}$$

$$\zeta_{P\nu i} = (P|\nu\lambda)C_{i\lambda}, \tag{2}$$

$$K[D_{\lambda\sigma}]_{\mu\nu} = \zeta_{P\mu i}\zeta_{P\nu i}, \tag{3}$$

where i is an occupied index, P is the index of the aux-

iliary basis function, and μ, ν, λ, and σ are atomic orbital (AO) indices. Holding the $(P|\nu\lambda)$ quantity in a tensor $T_{P\nu\lambda}$ offers the benefit of a straightforward optimized matrix-matrix operation in Eq. 2. However, this neglects the symmetricity and sparsity of the three-index integrals $(P|\nu\lambda)$. Accounting for both of these properties leads to the previously stored form of $T_{P\nu\lambda^\nu}$ where the λ index was represented sparsely for each $P\nu$ pair by removing all duplicate or zero values; the sparsity of the index λ depends on the value of ν, hence the notation λ^ν. This form provides a highly compact representation of the $(P|\nu\lambda)$ tensor; however, the matrix-matrix operation to form $\zeta_{P\nu i}$ in Eq. 2 requires unpacking to a dense form, causing the previously mentioned data bottleneck.

To overcome this issue, the new J and K builds in PSI4 hold the $(P|\nu\lambda)$ quantity in a $T_{\nu P\lambda^\nu}$ representation, where there is a unique mapping for the $P\lambda$ indices for each ν index. While full sparsity can also be represented in this form, the symmetry of the AOs is lost, leading to this quantity being twice as large in memory or disk. This form requires the $C_{i\lambda^\nu}$ matrix to be packed for every ν index for optimal matrix-matrix operations in Eq. 2. While both the $T_{P\nu\lambda^\nu}$ and $T_{\nu P\lambda^\nu}$ forms require packing or unpacking of tensors, the former requires QN^2 oper-

ations while the latter requires N^2o operations where Q is the size of the auxiliary index, N the number of basis functions, and o the size of the occupied index. In practice $o \ll Q$, often resulting in $15\times$ less data movement, and generally all but removing the bottleneck.

This small data organization change combined with vectorization and parallelization improvements has led to performance increases, especially for a high number of cores and when the system is very sparse, with the drawback of doubling the memory footprint. For a system of two stacked benzenes in the cc-pVDZ basis set (228 basis functions), the new JK algorithm is 2.6, 3.6, 3.7, and $4.3\times$ faster than the old algorithm for 1, 8, 16, and 32 threads, respectively. For a more extensive system of twenty stacked benzenes with cc-pVDZ (2280 basis functions), the respective speedups are 1.5, 1.7, 2.1, 2.2$\times$. Psi4 automatically detects which algorithm to use based on the amount of available memory.

C. Additive Dispersion Models

Psi4 specializes in providing convenient access to methods with additive dispersion corrections. While Grimme's three-component corrections to mean-field methods, HF-3c[137] and PBE-3c[138] (external via DFTD3[139] and GCP[140]), and the simpler pairwise additive schemes -D2[141] (internal) and -D3[142,143] (external via DFTD3) have long been available, a similar correction to perturbation theory, MP2-D[144] (external via MP2D[145]) and a non-local correction to DFT through the VV10 functional, DFT-NL[146] (internal) are now also available simply as `gradient("mp2-d")` or `energy("b3lyp-nl")`. See Table I for details of external software.

Psi4 v1.4 uses the -D3 correction in a new method, SAPT0-D. While SAPT0 has long been applicable to systems with upwards of 300 heavy atoms by leveraging optimized DF routines for both JK builds and MP2-like $E_{\mathrm{disp}}^{(20)}$ and $E_{\mathrm{exch-disp}}^{(20)}$ terms, it is limited by the $\mathcal{O}(N^5)$ scaling of the second-order dispersion (N proportional to system size). By refitting the -D3 damping parameters against a training set of nearly 2300 CCSD(T)/CBS IE and using the result in place of the analytic SAPT0 dispersion component, SAPT0-D at $\mathcal{O}(N^4)$ scaling achieves a $2.5\times$ speedup for systems with about 300 atoms (increasing for larger systems), while removing the worst outliers and achieving the same error statistics as full SAPT0 when evaluated on 8100 IE of bimolecular complexes.[147]

The SAPT0-D approach is also applicable to the functional group partition of SAPT.[53] The resulting F-SAPT0-D has been applied to understand the differential binding of the β_1-adrenoreceptor (β_1AR) (figure 5) in its active (G-protein coupled) versus inactive (uncoupled) forms to the partial agonist salbutamol. While experimentally determined $\Delta\Delta G_{\mathrm{bind}}$ was previously justified with respect to changes in binding site geometry upon β_1AR activation,[148] F-SAPT0-D quantifies the contribution of each functional group contact, revealing that differential binding is due in large part to cooperativity of distant amino acid residues and peptide bonds, rather than only local contacts.

D. SAPT(DFT)

Psi4 now provides SAPT(DFT),[149] also called DFT-SAPT,[150] which approximately accounts for the intramolecular electron correlation effects that are missed in SAPT0 by including correlation-like effects found in DFT. The Hartree–Fock orbitals are replaced with Kohn–Sham orbitals,[151] and induction terms are solved using the coupled-perturbed Kohn–Sham equations. The long-range behavior that is important for dispersion interactions is known to be problematic for generalized gradient approximation (GGA) functionals, and in DFT-SAPT this is corrected by gradient-regulated asymptotic correction (GRAC)[152] in obtaining the Kohn–Sham orbitals. Dispersion energies are obtained by solving for the TDDFT propagator of each monomer and integrating the product of the propagators over the frequency domain.[153,154] In Psi4 1.4 we have improved the TDDFT dispersion capabilities to allow hybrid kernels in the TDDFT equations,[155] which can significantly improve accuracy when hybrid functionals are used to determine the orbitals.[150,156]

E. SAPT0 Without the Single-Exchange Approximation

The SAPT module in Psi4 now has an option to compute the second-order SAPT0 exchange corrections $E_{\mathrm{exch-ind,resp}}^{(20)}$ and $E_{\mathrm{exch-disp}}^{(20)}$ without the use of the common S^2 approximation, that is, using the complete antisymmetrizer in the expressions instead of its approximation by intermolecular exchanges of a single electron pair. The working equations for the non-approximate second-order corrections were derived and implemented for the first time in Refs. 157 and 158 in the molecular-orbital (MO) form prevalent in the classic SAPT developments. We have recast the nonapproximate formulas for $E_{\mathrm{exch-ind,resp}}^{(20)}$ and $E_{\mathrm{exch-disp}}^{(20)}$ of Refs. 157,158 into the AO form and implemented them efficiently in Psi4 with DF. As these AO-based expressions have not been published before, we present them together with an outline of their derivation in the Supplementary Material. Thanks to this new development, the entire SAPT0 level of theory (but not higher levels such as second-order, SAPT2) is now available in Psi4 without the single-exchange approximation. Preliminary numerical tests show[157–159] that the replacement of $E_{\mathrm{exch-disp}}^{(20)}(S^2)$ by its nonapproximated counterpart introduces inconsequential changes to the SAPT0 interaction potentials at short intermolecular separations. In contrast, the full $E_{\mathrm{exch-ind,resp}}^{(20)}$ values often deviate significantly from $E_{\mathrm{exch-ind,resp}}^{(20)}(S^2)$ at short
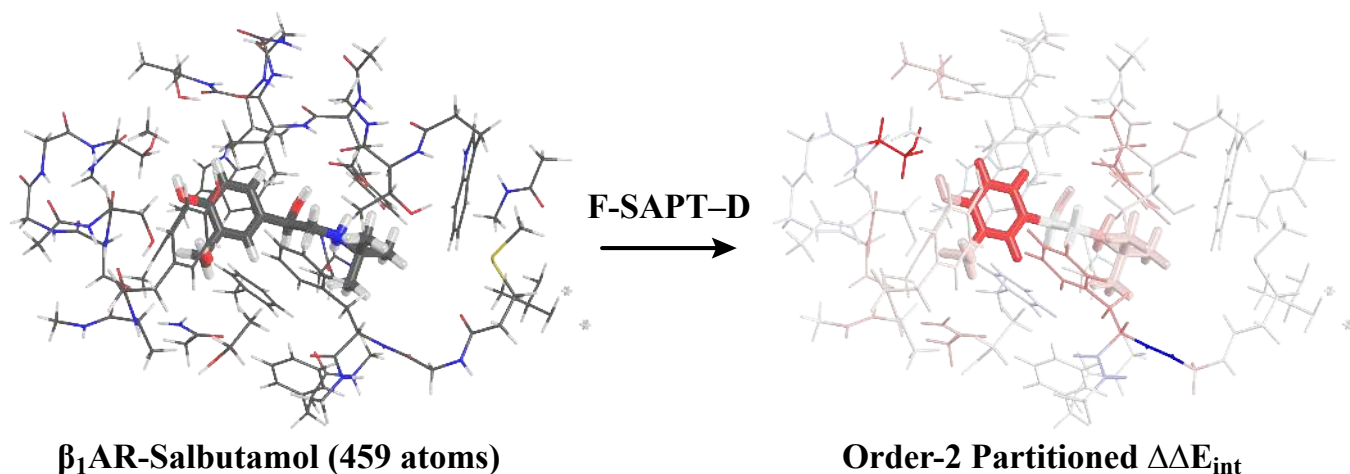
FIG. 5. F-SAPT0-D3M(BJ)/jun-cc-pVDZ analysis of 459 atoms (5,163 orbital and 22,961 auxiliary basis functions) from β_1AR–salbutamol co-crystal (PDB: 6H7M). (left) Geometry of ligand (wide sticks) and residues within 7Å (thin sticks). (right) Order-2 F-SAPT difference analysis of active vs. inactive complex, with functional groups colored by contribution to $\Delta\Delta E_{\text{int}}$ (red: more attractive in activated state; blue: more attractive in inactive state; color saturation at ± 10 kcal mol^{-1}).

range, especially for interactions involving ions.[160] At the usual SAPT0 level (as defined e.g. in Ref. 161), this difference between $E_{\text{exch−ind,resp}}^{(20)}$ and $E_{\text{exch−ind,resp}}^{(20)}(S^2)$ cancels out when the $\delta E_{\text{HF}}^{(2)}$ term that approximates the higher-order induction and exchange induction effects from a supermolecular HF calculation is taken into account. However, the removal of the S^2 approximation from second-order SAPT0 will significantly affect SAPT results computed without the $\delta E_{\text{HF}}^{(2)}$ correction.

F. SF-SAPT

An open-shell SAPT feature that is currently unique to Psi4 is the ability to compute the leading exchange term, $E_{\text{exch}}^{(10)}(S^2)$, for an arbitrary spin state of the interacting complex, not just its highest spin state. This *spin-flip SAPT (SF-SAPT)* method was introduced in Ref. 162 and so far applies to the interaction between two open-shell systems described by their ROHF determinants. Such an interaction leads to a bundle of asymptotically degenerate states of the interacting complex, characterized by different values of the spin quantum number S. These states share the same values of all electrostatic, induction, and dispersion energies, and the splitting between them arises entirely out of electron exchange. In such a case, the SF-SAPT approach implemented in Psi4 can provide an inexpensive (cost is similar to standard $E_{\text{exch}}^{(10)}(S^2)$) and qualitatively correct first-order estimate of the splittings between different spin states of the complex. In addition, all terms can be computed using standard SCF JK quantities and have been implemented within Psi4 in a Psi4NumPy formalism, as the best performance can be achieved without any additional compiled code.

G. Libint2 and Simint

The Libint package[163] has been the default engine for two-electron integrals since the development of Psi3 two decades ago. Allowing arbitrary levels of angular momentum and numerous integral kernels, Libint has proven a reliable tool for generating the integrals that are central to QC. However, modern CPUs increasingly derive their power from a combination of multi-core and single instruction, multiple data (SIMD) technologies, rather than the regular strides in clock speed that were realized around the time of Psi3's development. While Psi4 has exploited multi-core technologies for some time via OpenMP, its SIMD capabilities were previously limited to the linear algebra libraries used to power SCF and post-HF methods. In Psi4 v1.4, the Libint package has been superseded by Libint2,[164] which partially exploits SIMD capabilities by vectorizing the work needed for a given shell quartet, making it better suited for modern compute architectures. Libint2 permits additional integral kernels, including the Yukawa- and Slater-type geminal factors, which expand the range of DFT and explicitly correlated methods that may be implemented. Libint2 is also preferable from a software sustainability perspective as it is actively maintained and developed, unlike the original Libint.

Although Libint2 is now the default integrals engine, Psi4 is written to allow the use of alternative integrals packages, and an interface to Simint[165,166] is also provided. Simint was designed from the beginning with SIMD parallelism in mind. By reordering shell pairs to be grouped by common angular momentum classes, Simint achieves a compelling level of vectorization on the latest AVX512 chipsets. The Psi4 integrals interface has been generalized to allow the shell pairs to be given in arbitrary order and to account for the possibility of batching

among them, thus allowing SIMINT to take full advantage of its approach to vectorization.

H. SCF Guesses

The reliability of the atomic solver used for the superposition of atomic densities[167,168] (SAD) initial guess has been greatly improved in PSI4, and the SAD guess has been made the default also for open-shell and restricted open-shell calculations, resulting in significantly faster convergence, especially for systems containing heavy atoms such as transition metal complexes. Although powerful in many cases, the SAD guess does not yield molecular orbitals, and it may thereby be harder to build a guess with the wanted symmetry. The traditional alternatives to SAD that do yield molecular orbitals, the core orbital guess or the generalized Wolfsberg–Helmholz[169] modification thereof, fail to account for electronic screening effects whose importance increases rapidly with increasing nuclear charge, resulting in horrible performance.[170] However, guesses that both account for electronic screening and yield guess orbitals have recently been described in Ref. 170 and are now implemented in PSI4: an extended Hückel guess employing the atomic orbitals and orbital energies from the SAD solver, the SAD natural orbitals (SADNO) guess, and the superposition of atomic potentials (SAP) guess that constructs a guess Fock matrix from a sum of atomic effective potentials computed at the complete basis set limit.[171,172] With the improvements to SAD and the introduction of the three novel guesses, PSI4 can be applied even to challenging systems. Calculations are now possible even in overcomplete basis sets, as redundant basis functions are removed automatically by default in PSI4 via the pivoted Cholesky decomposition procedure.[173]

I. TDDFT

We have recently added time-dependent DFT capabilities using either the full TDDFT equations (also known as the random-phase approximation, RPA) or the Tamm–Dancoff approximation (TDA).[174] The former yields a *generalized* eigenvalue problem, and our solver leverages the Hamiltonian structure of the equations to ensure robust convergence.[175] The latter corresponds to a Hermitian eigenvalue problem, and we employ a Davidson solver.[176] The excitation energies and vectors are obtained from the following generalized eigenvalue problem, also known as the *response eigenvalue problem*:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^* & \mathbf{A}^* \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{Y}_n \end{pmatrix} = \omega_n \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{Y}_n \end{pmatrix}. \quad (4)$$

The excitation eigenvectors, $(\mathbf{X}_n, \mathbf{Y}_n)^t$, provide information on the nature of the transitions and can be used to form spectroscopic observables, such as oscillator and rotatory strengths. The $\mathbf{A}$ and $\mathbf{B}$ matrices appearing on the left-hand side are the blocks of the molecular electronic Hessian[177] whose dimensionality is $(ov)^2$, with o and v the number of occupied and virtual MOs, respectively. Due to this large dimensionality, rather than form $\mathbf{A}$ and $\mathbf{B}$ explicitly, one instead uses subspace iteration methods to extract the first few roots. In such methods, the solutions are expanded in a subspace of trial vectors $\mathbf{b}_i$, and the most compute- and memory-intensive operations are the formation and storage of the matrix-vector products $(\mathbf{A} + \mathbf{B})\mathbf{b}_i$ and $(\mathbf{A} - \mathbf{B})\mathbf{b}_i$. These matrix-vector products are equivalent to building generalized Fock matrices; the efficient JK-build infrastructure of PSI4 (Sec. V B) can thus be immediately put to use also for the calculation of TDDFT excitation energies. In fact, construction of these product vectors is the only part written in C++. All other components, including the subspace iteration techniques, are written in Python for easy readability and maintainability. Following our design philosophy, we have written the required subspace solvers for the response eigenvalue problems in a generic way, so that they may be reused for future features.

1. *Example of Rapid Prototyping*

To illustrate the use of PSI4 and PSI4NUMPY to rapidly implement new features, figure 6 shows an easy oscillator strength implementation at the Python layer. Excitations are obtained by calling the `tdscf_excitations()` function, and dipole moment integrals are calculated trivially in four lines of code by accessing the occupied and virtual parts of the SCF coefficient matrix and the dipole moment integrals from LIB-MINTS. The oscillator strengths are then computed from the MO basis electric dipole moment integrals $\langle\phi_a|\hat{\boldsymbol{\mu}}|\phi_i\rangle$ and the right excitation vectors $\mathbf{X}_n + \mathbf{Y}_n$:

$$f = \frac{2}{3}\omega_n \sum_{u=x,y,z} \sum_i^{\text{occ}} \sum_a^{\text{vir}} |(\mathbf{X}_n + \mathbf{Y}_n)_{ia}\langle\phi_a|\hat{\mu}_u|\phi_i\rangle|^2. \quad (5)$$

Fig. 7 shows an example UV-Vis spectrum using these oscillator strengths, as fitted by applying a Gaussian-shaped broadening to the computed excitation energies. We are also working on the implementation of gauge-including atomic orbitals (London orbitals) to enable magnetic response evaluations needed to calculate properties like optical rotation, electronic circular dichroism, etc.

VI. SOFTWARE ECOSYSTEM

Like all QC packages, PSI4 strives to continuously expand its capabilities to advance research both in methods development and applications. New methods are introduced frequently in electronic structure theory, and

FIG. 6. Example Python implementation of TDDFT oscillator strengths

```
import numpy as np
import psi4

# Import TDDFT solvers module
from psi4.driver.procrouting.response.scf_response import tdscf_excitations

psi4.set_output_file("tddft.out")
# set molecule "mol" here

psi4.set_options({"save_jk": True})
e, wfn = psi4.energy("B3LYP/aug-cc-pvdz", return_wfn=True, molecule=mol)

# Dipole moment integrals
mints = psi4.core.MintsHelper(wfn.basisset())
C_L = wfn.Ca_subset("SO", "OCC")
C_R = wfn.Ca_subset("SO", "VIR")
dipole = [psi4.core.triplet(C_L, x, C_R, True, False, False) for x in mints.so_dipole()]

# Compute 10 roots per irrep using full TDDFT
rpa = tdscf_excitations(wfn, states_per_irrep=[10], r_tol=1e-3)
# Now compute oscillator strengths
spectrum_rpa = []
for omega, (XpY, _), _ in rpa:
    edtm = np.array([XpY.vector_dot(u) for u in dipole])
    f = 2/3 * omega * np.sum(edtm**2)
    spectrum_rpa.append((omega, f))
```
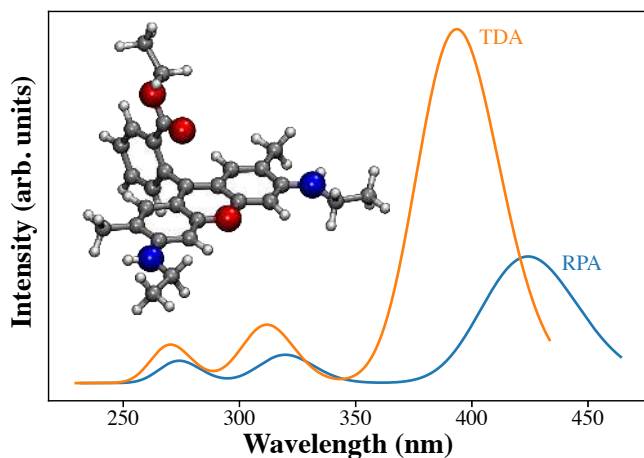
$$(6)$$

FIG. 7. UV-Vis spectrum of rhodamine 6G at the PBE0/aug-pcseg-2 level of theory. The spectra computed using full TDDFT (RPA) and the Tamm–Dancoff approximation (TDA) are reported in blue and orange, respectively.

it would be a challenge to implement all the latest advances. The Psi4 team prefers to encourage the development of reusable libraries, so that new methods need only be implemented once (by the experts), and can then be adopted by any QC code with merely a short, custom interface. This ecosystem-building approach has the advantages of (i) not binding a community library's use to a single software package, (ii) encouraging smaller software projects that are more modular in function and ownership and more localized in (funding) credit, (iii) facilitating the propagation of new features and bug fixes by using a generic interface rather than embedding a code snapshot. Since v1.1, Psi4 has added new projects to its ecosystem, contributed back to existing projects, and disgorged some of its own code into projects that are more tightly defined. Discussed below are a selection of illustrative or newly interfaced projects. The full ecosystem of external, connected software is collected into Table I: used by Psi4; and Table II: uses Psi4.

A. Sustainability through community libraries

The introduction of Libint2 and LibXC not only provides new features (see Secs. V G and V A, respectively), but also results in substantial simplifications to the code base. The previous version of Libint only provided the recursion routines, relying on the calling program to provide the fundamental s-type integrals used as the starting point. There were also restrictions on the angular momentum ordering among the four centers, requiring bookkeeping to apply permutations to the resulting integrals in the case where reorderings were necessary to satisfy these requirements. Furthermore, Libint1 provided only the minimal number of integral derivatives required by translational invariance,[236,237] requiring the

TABLE I. Quantum chemistry software that Psi4 can use (upstream interaction).

Software[a]	Group	Added	License	Language	Comm.[b]	Cite[c]		Capability
Upstream Required C-link								
Libint1	Valeev	v1.0[d]	LGPL-3.0	C	C API	163	–	two-electron and properties integrals
Libint2	Valeev	v1.4	LGPL-3.0	C^{++}	C^{++} API	164	–	two-electron and properties integrals
LibXC	Marques	v1.2	MPL-2.0	C	C API	178	131	definitions, compositions of density functionals
gau2grid	Smith	v1.2	BSD-3-Cl	C/Py	C API	179	–	Gaussian collocation grids for DFT
Upstream Required Py-link								
QCElemental	MolSSI	v1.3	BSD-3-Cl	Py	Py API	126	121	physical constants and molecule parsing
QCEngine	MolSSI	v1.4	BSD-3-Cl	Py	Py API	127	121	QC schema runner with dispersion & opt engines
Upstream Optional C-link								
dkh	Reiher	v1.0	LGPL-3.0	Fortran	C API	180	80,81	relativistic corrections
libefp	Slipchenko	v1.0[e]	BSD-2-Cl	C	C API	181	182	fragment potentials
gdma	Stone	v1.0	GPL-2.0	Fortran	C API	183	184	multipole analysis
CheMPS2	Wouters	v1.0	GPL-2.0	C^{++}	C^{++} API	185	186,187	DMRG and multiref. PT2 methods
PCMSolver	Frediani	v1.0	LGPL-3.0	C^{++}/Fortran	C^{++} API	188	189	polarizable continuum / implicit solvent modeling
erd	QTP	v1.0[d]	GPL-2.0	Fortran	C API	190	191	two-electron integrals
simint	Chow	v1.1	BSD-3-Cl	C	C API	192	165	vectorized two-electron integrals
ambit	Schaefer	v1.2	LGPL-3.0	C^{++}/Py	C^{++} API	193	–	tensor manipulations
Upstream Optional Py-link or exe								
dftd3	Grimme	v1.0	GPL-1.0	Fortran	QCSchema	139	142,143	empirical dispersion for HF & DFT
MRCC	Kallay	v1.0	pty	C^{++}/Fortran	text file	–	35	arbitrary order CC & CI
gCP	Grimme	v1.1	GPL-1.0	Fortran	Py intf./CLI	140	137,138	small-basis corrections
PylibEFP	Sherrill	v1.3	BSD-3-Cl	C^{++}/Py	Py API	194	–	Python API for libefp
MP2D	Beran	v1.4	MIT	C^{++}	QCSchema	145	144	empirical dispersion for MP2
cppe	Dreuw	v1.4	LGPL-3.0	C^{++}/Py	Py API	195	196	polarizable embedding / explicit solvent modeling
adcc	Dreuw	v1.4	GPL-3.0+pty	C^{++}/Py	Py API	197	113	algebraic-diagrammatic construction methods

[a] Binary distributions available from Anaconda Cloud for all projects except for MRCC. For the channel in `conda install <project> -c <channel>`, use `psi4` except for adcc from `adcc` and gau2grid, QCElemental, and QCEngine from `conda-forge`, the community packaging channel.
[b] Means by which Psi4 communicates with project.
[c] First reference is software repository. Second is theory or software in literature.
[d] No longer used. Libint1 last supported before v1.4. erd last supported before v1.2.
[e] Since v1.3, libefp called through PylibEFP.

calling code to compute the missing terms by application of the relationships. The combination of applying permutations and translational invariance amounted to over 3000 lines of code in previous Psi4 versions, primarily due to the complexity introduced by second derivative integrals. In Libint2, the fundamental integrals are provided and the translational invariance is applied automatically for derivatives, and the shells can be fed in in any order of the angular momenta. With these tasks outsourced to Libint2, the latest Psi4 codebase is significantly cleaner and more maintainable.

With the transition to the LibXC[131] library for DFT calculations, in accordance with the modular development model, Psi4 gains continuous fixes and new features, which is especially important as none of the primary Psi4 development groups specialize in DFT. Thanks to LibXC, Psi4 now supports over 400 functionals of various rungs. Final DFT compositions suitable for `energy()` are now defined by LibXC and are directly subsumed into Psi4's functional list, making for more maintainable code. In cooperation with LibXC upstream, the Psi4 authors have contributed an alternate CMake build system and a Python API, PylibXC, to LibXC, and also provided help in porting to Windows.

B. Launching community libraries

1. QCElemental

When the needs of ongoing research projects outgrew LibMints's C++ parsing of molecule specification strings, a redesign was implemented in Python and transferred to QCElemental to serve as the backend to QCSchema `Molecule` validation. The resulting code is easily extensible, mirrors the schema (though with additional fields to accomodate Psi4's Z-Matrix and deferred geometry finalization features), and accepts and returns dictionary, schema, array, or string-based representations. Additionally, it performs strong physics-based validation and defaulting for masses, mass numbers, total and fragment charges and multiplicities, and basis function ghosting, saving considerable validation code in Psi4 as a QCElemental client.

QCElemental additionally provides a light Python interface over NIST CODATA and periodic table data and other "look-up" quantities like van der Waals and covalent radii. By switching to QCElemental API calls in Psi4's Python code and using its header-writing utilities for C++ code, readability has improved, and datasets are easier to update.

TABLE II. Chemistry software that can use Psi4 (downstream interaction).

Software[a]	Group	V.[b]	License	Language	Comm.[c]	Cite[d]		Psi4 **Provides**
Downstream Optional C-link, plugins								
v2rdm_casscf	DePrince	v1.0	GPL-2.0	C++/Fortran	C++ API	71	198	backend for variational 2-RDM-driven CASSCF
Forte	Evangelista	v1.0	LGPL-3.0	C++/Py	C++ API	70	68,69	backend for multiref. many-body mtds & sel. CI
CCT3	Piecuch	v1.1	LGPL-3.0	Fortran	C++ API	199	200,201	backend for actv-sp CCSDt, CC(t;3), CR-CC(2,3)
gpu_dfcc	DePrince	v1.2	GPL-2.0	C++/Cuda	C++ API	202	203	backend for GPU-accelerated DF-CCSD & (T)
Downstream Optional Py-link or exe								
WebMO	Polik	v1.0	pty	Java/Perl	PSIthon	–	204	QC engine for GUI/web server
Molden	Schaftenaar	v1.0	pty	Fortran	Molden file	205	206	orbitals for orbital/density visualization
JANPA	Hovorun	v1.0	BSD-4-Cl	Java	Molden file	207	208	orbitals for natural population analysis (NPA)
Psi4NumPy	Smith	v1.1	BSD-3-Cl	Py	PsiAPI	100	10	QC essentials for rapid prototyping and QC edu.
Psi4Education	McDonald	v1.1	BSD-3-Cl	Py	PsiAPI	209	122	QC engine for instructional labs
PsiOMM	Sherrill	v1.1	BSD-3-Cl	Py	PsiAPI	210	–	self for interface btwn Psi4 and OpenMM
htmd/parameterize	Acellera	v1.1	pty	Py	PSIthon	211	212,213	QC engine for force-field parametrization for MD
GPUGrid	De Fabritiis	v1.1	pty	Py	PSIthon	214	215	QC torsion scans for MD-at-home
sns-mp2	D. E. Shaw	v1.2	BSD-2-Cl	Py	PsiAPI	216	217	backend for spin-network-scaled MP2 method
resp	Sherrill	v1.2	BSD-3-Cl	Py	PsiAPI	218	115	ESP for restrained ESP (RESP) fitting
QCEngine	MolSSI	v1.2	BSD-3-Cl	Py	QCSchema	127	121	QC engine for QC schema runner
QISkit-Aqua	IBM	v1.2	Apache-2.0	Py	PSIthon	219	–	engine for quantum computing algorithms
MS Quantum	Microsoft	v1.2	MIT	C#/Q#	PsiAPI	220	–	engine for quantum computing algorithms
Orion	OpenEye	v1.2	pty	Go/Py	PsiAPI	–	–	QC engine for drug-design workflow
CrystaLattE	Sherrill	v1.2	LGPL-3.0	Py	PSIthon	221	222	QC and MBE engine for crystal lattice energies
OpenFermion	Google	v1.3	Apache-2.0	Py	PSIthon	223	224	engine for quantum computing algorithms
OpenFermion-Psi4	Google	v1.3	LGPL-3.0	Py	PSIthon	225	224	self for interface btwn Psi4 and OpenFermion
QCDB	Sherrill	v1.3	BSD-3-Cl	Py	QCSchema	226	–	engine for QC common driver
OptKing	King	v1.3	BSD-3-Cl	Py	QCSchema	227	–	gradients for geometry optimizer
psixas	Gryn'ova	v1.3	GPL-3.0	Py	PsiAPI	228	–	backend for X-ray absorption spectra
ASE	ASE	v1.4	LGPL-2.1	Py	PsiAPI	229	230	QC engine for CMS code runner
i-PI	Ceriotti	v1.4	GPL-3.0	Fortran/Py	PsiAPI	231	232	QC gradients for MD runner
MDI	MolSSI	v1.4	BSD-3-Cl	C	PsiAPI	233	–	QC engine for standardized CMS API
geomeTRIC	Wang	v1.4[e]	BSD-3-Cl	Py	QCSchema	234	235	QC gradients for geometry optimizer
QCFractal	MolSSI	v1.4	BSD-3-Cl	Py	QCSchema	128	121	QC engine for database and compute manager

[a] Binary distributions available from Anaconda Cloud for some projects. For the channel in `conda install <project> -c <channel>`, use psi4 for v2rdm_casscf, gpu_dfcc, sns-mp2, resp, OpenFermion, and OpenFermion-Psi4; acellera for htmd/parameterize; and `conda-forge`, the community packaging channel, for QCEngine, ASE, MDI, geomeTRIC, and QCFractal.

[b] Earliest version of Psi4 with which software works.

[c] Apart from compiled plugins that interact directly with Psi4's C++ layer, downstream projects use established file formats like Molden or one of the three input modes of figure 1.

[d] First reference is software repository. Second is theory or software in literature.

[e] GeomeTRIC has called Psi4 through PSIthon since v1.0. QCEngine has driven geomeTRIC to drive Psi4 through QCSchema since v1.3. Psi4 can itself call geomeTRIC through QCSchema since v1.4.

2. QCEngine

Psi4 has long supplemented its internal empirical dispersion capabilities (Sec. V C) with external projects, namely DFTD3 and MP2D executables. These were run via a Python interface that additionally stores fitting and damping parameters at the per functional level, so that the programs are used solely for compute and not for internal parameters. Since operation is independent of Psi4, the Python interfaces have been adapted to QC-Schema and moved to the QCEngine repository where they can be of broader use.

3. Gau2Grid

Improvements to the Psi4 DFT code highlighted a bottleneck at the computation of the collocation matrix between basis functions and the DFT grid. It was found that the simple loops existing in Psi4 did not vectorize well and exhibited poor cache performance. Much in the same way that modern two-electron libraries work,

Gau2Grid[179] begins with a template engine to assist in writing unrolled C loops for arbitrary angular momentum and up to third-order derivatives. This template engine also allows multiple performance strategies to be employed and adjusted during code generation, depending on the angular momentum, derivative level of the requested matrix, and the hardware targeted. Gau2Grid also has a Python interface to allow usage in Python programs that need fast collocation matrices.

4. PylibEFP

In the course of shifting control of SCF iterations from C++ to Python, it became clear that the effective fragment potential[238,239] (EFP) capabilities through Kaliman and Slipchenko's LIBEFP library[182] would be convenient in Python. Since LIBEFP provides a C interface, a separate project of essentially two files, PylibEFP,[194] wraps it into an importable Python module. PylibEFP includes a full test suite, convenient EFP input parsing, and an interface amenable to schema communication (a QCEngine adaptor is in progress). Psi4 em-

ploys PYLIBEFP for EFP/EFP energies and gradients and EFP/SCF energies.

C. Selected new features from community libraries

1. adcc

ADC-connect (ADCC),[113] a hybrid Python/C++ toolkit for excited-state calculations based on the algebraic-diagrammatic construction scheme for the polarization propagator (ADC),[240–242] equips PSI4 with ADC methods (in-memory only) up to third order in perturbation theory. Expensive tensor operations use efficient C++ code, while the entire workflow is controlled by Python. PSI4 and ADCC can connect in two ways. First, PSI4 can be the main driver; here, method keywords are given through the PSI4 input file and ADCC is called in the background. Second, the PSI4 `Wavefunction` object from a SCF can be passed to ADCC directly in user code; here, there is more flexibility for complex workflows or for usage in a JUPYTER notebook.

2. SNS-MP2

McGibbon and coworkers[217] applied a neural network trained on HF and MP2 IE and SAPT0 terms to predict system-specific scaling factors for MP2 same- and opposite-spin correlation energies to define the spin-network-scaled, SNS-MP2, method. This has been made available in a PSI4 pure-Python plugin[216] so that users can call `energy("sns-mp2")`, which manages several QC calculations and the model prediction in the background, then returns an IE likely significantly more accurate than conventional MP2.[217] By using PSI4's export of wavefunction-level arrays to Python, the developers were able to speed up calculations through custom density matrix manipulations of basis projection, fragment stacking, and fragment ghosting.

3. CPPE

PSI4 now supports the polarizable embedding (PE) model[243,244] through the CPPE library.[196] In the PE model, interactions with the environment are represented by a multi-center multipole expansion for electrostatics, and polarization is modeled through dipole polarizabilities usually located at the expansion points. The interface to the CPPE library is entirely written in Python and supports a fully self-consistent description of polarization for all SCF methods inside PSI4. In the future, PE will also be integrated in a fully self-consistent manner for molecular property calculations and TDDFT. Integration of CPPE motivated efficiency improvements to the electric field integrals and multipole potential integrals, which also benefit the related EFP method.

4. geomeTRIC

Wang and Song[234,235] developed a robust geometry optimization procedure to explicitly handle multiple non-covalently bound fragments using a translation-rotation-internal coordinate (TRIC) system. Their standalone geometry optimizer, GEOMETRIC, supports multiple QC packages including PSI4 through a command-line interface. QCENGINE offers a GEOMETRIC procedure, allowing PSI4 and others to use the new optimizer with a Python interface. The latest PSI4 release adds native GEOMETRIC support, allowing users to specify the geometry optimizer within an input, e.g., `optimize(..., engine="geometric")`.

5. v2rdm_casscf

PSI4 can perform large-scale approximate CASSCF computations through the `v2rdm_casscf` plugin,[71] which describes the electronic structure of the active space using the variational two-electron RDM approach.[198,245,246] Version 0.9 of `v2rdm_casscf` can perform approximate CASSCF calculations involving active spaces as large as 50 electrons in 50 orbitals[198] and is compatible with both conventional four-center electron repulsion integrals (ERI) and DF/Cholesky decomposition approximations. Active-space specification in `v2rdm_casscf` is consistent with other active-space methods in PSI4, and users can write RDMs to disk in standard formats (*e.g.*, FCIDUMP) for post-processing or for post-CASSCF methods. Geometry optimizations using analytic energy gradients can also be performed (with four-center ERIs).[247] While most use cases of `v2rdm_casscf` involve calls to PSI4's `energy()` or `gradient()` functions, key components of the plugin such as RDMs are also accessible directly through Python.

6. CCT3

The CCT3 plugin[199] to PSI4 is capable of executing a number of closed- and open-shell CC calculations with up to triply excited (T_3) clusters. Among them is the active-space CC approach abbreviated as CCSDt,[248–251] which approximates full CCSDT by selecting the dominant T_3 amplitudes via active orbitals, and the CC(t;3) method, which corrects the CCSDt energies for the remaining, predominantly dynamical, triple excitations that have not been captured by CCSDt.[200,201] The CC(t;3) approach belongs to a larger family of methods that rely on the generalized form of biorthogonal moment expansions defining the CC(P;Q) formalism.[200,201]

The CCSDt method alone is already advantageous, since it can reproduce electronic energies of near-CCSDT quality at a small fraction of the computational cost while accurately describing select multireference situations, such as single bond breaking. CC(t;3) improves

on the CCSDt energetics even further, being practically as accurate as full CCSDT for both relative and total electronic energies at essentially the same cost as CCSDt. CCSDt and CC(t;3) converge systematically towards CCSDT as the active space is increased.

The CCT3 plugin can also be used to run CCSD and CR-CC(2,3) calculations. This can be done by making the active orbital set (defined by the user in the input) empty, since in this case CCSDt = CCSD and CC(t;3) = CR-CC(2,3). We recall that CR-CC(2,3) is a completely renormalized triples correction to CCSD, which improves the results obtained with the conventional CCSD(T) approach without resorting to any multireference concepts and being at most twice as expensive as CCSD(T).[252–254]

VII. DOWNSTREAM ECOSYSTEM

A. Computational Molecular Science Drivers

In addition to the closely associated ecosystem of the previous section, Psi4 is robust and simple enough that projects can develop interfaces that use it as a "black box". Of these, the one exposing the most Psi4 capabilities is the QCArchive Infrastructure project QCEngine, which can drive almost any single-command computation (e.g., gradient or complete basis set extrapolation, in contrast to a structure optimization followed by a frequency calculation) through the QCSchema specification. By launching Psi4 through QCFractal, the QCArchive database has stored 13M computations over the past year and is growing rapidly. A recent addition is the interface to the Atomic Simulation Environment[229,230] (ASE) through which energies and gradients are accessible as a `Calculator`. All Psi4 capabilities are available in ASE by using the in-built `psi4` module in PsiAPI. Another MolSSI project, the MolSSI Driver Interface[233] (MDI), devised as a light communication layer to facilitate complex QM/MM and machine learning workflows, has a Psi4 interface covering energies and gradients of HF and DFT methods. Finally, the i-PI universal force engine driver[231,232] has a Psi4 interface covering gradients of most methods.

B. Quantum Computing

Psi4 is also used in several quantum computing packages to provide orbitals and molecular integrals. For example, Psi4 is interfaced to the open-source quantum computing electronic structure package Open-Fermion[223,224] via the OpenFermion-Psi4 plugin.[225] The QISkit Aqua suite of algorithms for quantum computing developed by IBM[219] is also interfaced to Psi4 via input file. The Microsoft Quantum Development Kit[220] is another open-source project that takes advantage of Psi4's Python interface to generate molecular integrals

and then transform them in the Broombridge format, a YAML-based quantum chemistry schema.

C. Aiding Force-Field Development for Pharmaceutical Infrastructure

Many classical simulation methods have been developed with the aid of Psi4. As an illustrative example, torsion scans have been performed[9] using Open-Eye's Orion platform to provide a first principles evaluation of conformational preferences in crystals, and related methodology is used by the Open Force Field consortium[255] to parameterize force fields within the QCArchive framework. Psi4 has also found use in the development of nascent polarizable, anisotropic force fields by providing the distributed multipoles and MP2 electrostatic potentials (ESP) needed to parameterize the AMOEBA force field.[256] Moreover, the efficient SAPT code has been used in many recent developments in advanced force fields,[257] including the emerging successors to AMOEBA.[258,259] In collaboration with Bristol-Myers Squibb, we performed nearly 10,000 SAPT0 computations with Psi4 to train a pilot machine-learning model of hydrogen-bonding interactions,[8] and a much larger number is being computed for a follow-up study.

The restrained electostatic potential (RESP) model[260] is a popular scheme for computing atomic charges for use in force field computations. A Python implementation was initially contributed to the Psi4NumPy project, and later an independent open-source package was developed,[115,218] both of which employ Psi4 for the quantum electrostatic potential. The package supports the standard two-stage fitting procedure and multi-conformational fitting and also allows easy specification of complex charge constraints.

VIII. DEVELOPMENT AND DISTRIBUTION

A choose-your-own-adventure guide to obtaining Psi4 is available at `http://psicode.org/downloads`. Here, users and developers can select their operating system (Linux, Windows, Mac), Python version, then choose between downloading standalone installers for production-quality binaries, using the conda[261] package manager, and building the software from source. While standalone installers are only provided for stable releases, the source and conda installations also support the development branch. A new and substantial access improvement has been the porting of Psi4 to native Windows by one of the authors (R. G.) for the Acellera company (previously it was only available via Windows Subsystem for Linux, WSL) for GPUGrid, a distributed computing infrastructure for biomedical research.[214] This involved separate ports of the required dependency projects and introduction of Windows continuous integration to conserve compatibility during the course of largely Linux-

based development. The resulting uniform access to Psi4 in a classroom setting has been especially valuable for the Psi4Education project.

The cultivation of an ecosystem around Psi4 led to changes in the build system (Sec. 3 of Ref. 1), notably the maintain-in-pieces build-as-a-whole scheme where upstream and downstream dependencies remain in their own development repositories and are connected to Psi4 through a single-file footprint in the CMake build system. Through a "superbuild" setup, Psi4 and ecosystem projects can be flexibly built together upon a single command and use either pre-built packages or build dependencies from source. For distribution, we rely upon Anaconda Python (and its associated package manager, conda), which specializes in cross-platform building and management of Python/C++/Fortran software for the scientific community. Conda packages for Linux and Mac of Psi4 and its dependencies (such that `conda install psi4 -c psi4` yields a working installation) were in place by v1.1, when 11 packages were built for the `psi4` channel.

Since the v1.1 era, Psi4 developers have focused on modernization and compatibility. With the release of conda-build[262] v3 in late 2017 supporting enhanced build recipe language and built-in sysroots, Psi4 has upgraded to use the same compilers as the foundational Anaconda defaults and community conda-forge channels. A substantial improvement is that with the widespread availability of the Intel Math Kernel Library (MKL) through conda, Psi4 now uses the same libraries (`mkl_rt`) as those in packages like NumPy, rather than statically linking LAPACK, thereby eliminating a subtle source of import errors and numerical discrepancies. After these improvements, Psi4 today may be installed without fuss or incompatibility with other complex packages like Jupyter, OpenMM, and RDKit. While maintaining compatibility with defaults and conda-forge channels, Psi4 packages additionally build with Intel compilers and use flags that simultaneously generate optimized code for several architectures so that the same binary can run on old instruction sets like SSE2 but also run in an optimal fashion on AVX2 and AVX512. In keeping with our ecosystem philosophy, Psi4 will help a project with conda distribution on their own channel or ours or the community channel, or leave them alone, whichever level of involvement the developers prefer. We presently manage 23 packages. Since distributing through conda, Psi4 has accumulated 68k package manager and 93k installer downloads.

With a reliable distribution system for production-quality binaries to users, Psi4 can allow fairly modern code standards for developers, including C++14 syntax, Python 3.6+, and OpenMP 3+. By streamlining the build, Psi4 can be compiled and tested within time limits on Linux and Windows with multiple compilers. By performing this continuous integration testing on cloud services, developers receive quality control feedback on their proposed code changes. These include: through testing, rough assurance that changes do not break existing functionality; through coverage analysis, confidence that changes are being tested and a notice of testing gaps; through static analysis, alerts that changes have incorrect syntax, type mismatches, and more. The last reflects the advantages of using standard CMake build tools: the static analysis tool correctly guesses how to build the Psi4 source purely by examining build-language files in the repository.

IX. WEAKNESSES

Psi4's current focus on high-throughput quantum chemistry on conventional hardware has limited development of distributed parallel multi-node computing capabilities except for independent tasks managed by QCFractal as described in Sec. IV. GPU support is also limited beyond the gpu_dfcc module;[202,203] however, due to the plugin structure of Psi4, interfacing a GPU-based Coulomb (J) and exchange (K) code would enhance the majority of Psi4's capabilities, and Psi4 is in discussions to integrate such a plugin. Several other features have been requested by users such as advanced algorithms for transition state searching, implicit solvent gradients, and additional implicit solvent methods. Beyond the above capability weaknesses, a primary downside of open-source code is that there is no dedicated user support. While help can be found through a user forum at `http://forum.psicode.org`, a Slack workspace, and GitHub Issues, this support always comes from volunteers, and while questions are answered in the majority of cases, this is not guaranteed. On the other hand, the open-source software model empowers do-it-yourself fixes and extensions for power users and developers.

X. CONCLUSIONS

Psi4 is a freely available, open-source quantum chemistry (QC) project with a broad feature set and support for multi-core parallelism. The density-fitted MP2 and frozen natural orbital CCSD(T) codes are particularly efficient, even in comparison with commercial QC programs. Psi4 provides a number of uncommon features, including orbital-optimized electron correlation methods, density cumulant theory, and numerous intermolecular interaction methods in the symmetry-adapted perturbation theory family. With several input modes — text file, powerful Python application programming interface, and structured data — we can serve QC to traditional users, power users, developers, and database backends. The rewrite of our driver to work with task lists and integration with the MolSSI QCArchive Infrastructure project make Psi4 uniquely positioned for high-throughput QC.

Our development efforts and capabilities have been tremendously boosted by the "inversion" of Psi4 into a

Python module at v1.1. We are able to rely more heavily on Python for driver logic, simplifying export of structured data and transition to the new distributed driver. The hybrid C++/Python programming strategy has also greatly aided development in the multiconfigurational SCF (MCSCF) and SAPT modules. We are able to transparently convert between NuMPy and Psi4 linear algebra structures and fully access performance-critical C++ classes, facilitating rapid prototyping of novel SAPT and orbital-optimized MPn methods. We are able to load into Python scripts and connect easily with other CMS software like OpenMM and ASE.

Finally, we have fostered a QC software ecosystem meant to benefit the electronic structure software community beyond Psi4. Our adoption of the MolSSI QC-Schema should facilitate interoperability efforts, and our switch to a more permissive LGPL-3.0 license should aid developers and users who wish to deploy Psi4 as part of a larger toolchain or in cloud computing environments. We sincerely hope that the uptick in reusable software elements will continue in the future, so that new methods may be adopted quickly by many QC packages simply by interfacing a common implementation that is continuously updated, rather than developing redundant implementations in every code.

Data Availability

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

[1] R. M. Parrish, L. A. Burns, D. G. A. Smith, A. C. Simmonett, A. E. DePrince III, E. G. Hohenstein, U. Bozkaya, A. Y. Sokolov, R. Di Remigio, R. M. Richard, J. F. Gonthier, A. M. James, H. R. McAlexander, A. Kumar, M. Saitow, X. Wang, B. P. Pritchard, P. Verma, H. F. Schaefer III, K. Patkowski, R. A. King, E. F. Valeev, F. A. Evangelista, J. M. Turney, T. D. Crawford, and C. D. Sherrill, "Psi4 1.1: An open-source electronic structure program emphasizing automation, advanced libraries, and interoperability," J. Chem. Theory Comput., **13**, 3185 (2017).

[2] B. Jeziorski, R. Moszynski, and K. Szalewicz, "Perturbation theory approach to intermolecular potential energy surfaces of van der Waals complexes," Chem. Rev., **94**, 1887 (1994).

[3] K. Szalewicz, "Symmetry-adapted perturbation theory of intermolecular forces," WIREs Comput. Mol. Sci., **2**, 254 (2012).

[4] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, "A 5th-order perturbation comparison of electron correlation theories," Chem. Phys. Lett., **157**, 479 (1989).

[5] T. D. Crawford, C. D. Sherrill, E. F. Valeev, J. T. Fermann, R. A. King, M. L. Leininger, S. T. Brown, C. L. Janssen, E. T. Seidl, J. P. Kenny, and W. D. Allen, "PSI3: An open-source *ab initio* electronic structure package," J. Comput. Chem., **28**, 1610 (2007).

[6] J. M. Turney, A. C. Simmonett, R. M. Parrish, E. G. Hohenstein, F. A. Evangelista, J. T. Fermann, B. J. Mintz, L. A. Burns, J. J. Wilke, M. L. Abrams, N. J. Russ, M. L. Leininger, C. L. Janssen, E. T. Seidl, W. D. Allen, H. F. Schaefer III, R. A. King, E. F. Valeev, C. D. Sherrill, and T. D. Crawford, "Psi4: An open-source *ab initio* electronic structure program," WIREs Comput. Mol. Sci., **2**, 556 (2012).

[7] J. H. Thorpe, C. A. Lopez, T. L. Nguyen, J. H. Baraban, D. H. Bross, B. Ruscic, and J. F. Stanton, "High-accuracy extrapolated *ab initio* thermochemistry. IV. a modified recipe for computational efficiency," J. Chem. Phys., **150**, 224102 (2019).

[8] D. P. Metcalf, A. Koutsoukas, S. A. Spronk, B. L. Claus, D. A. Loughney, S. R. Johnson, D. L. Cheney, and C. D. Sherrill, "Approaches for machine learning intermolecular interaction energies and application to energy components from symmetry adapted perturbation theory," J. Chem. Phys., **152**, 074103 (2020).

[9] B. K. Rai, V. Sresht, Q. Yang, R. Unwalla, M. Tu, A. M. Mathiowetz, and G. A. Bakken, "Comprehensive assessment of torsional strain in crystal structures of small molecules and protein-ligand complexes using *ab initio* calculations," J. Chem. Inf. Model., **59**, 4195 (2019), ISSN 1549-9596.

[10] D. G. A. Smith, L. A. Burns, D. A. Sirianni, D. R. Nascimento, A. Kumar, A. M. James, J. B. Schriber, T. Zhang, B. Zhang, A. S. Abbott, E. J. Berquist, M. H. Lechner, L. A. Cunha, A. G. Heide, J. M. Waldrop, T. Y. Takeshita, A. Alenaizan, D. Neuhauser, R. A. King, A. C. Simmonett, J. M. Turney, H. F. Schaefer III, F. A. Evangelista, A. E. DePrince III, T. D. Crawford, K. Patkowski, and C. D. Sherrill, "Psi4NumPy: An interactive quantum chemistry programming environment for reference implementations and rapid development," J. Chem. Theory Comput., **14**, 3504 (2018).

[11] M. Pitoňák, P. Neogrády, J. Černý, S. Grimme, and P. Hobza, "Scaled MP3 non-covalent interaction energies agree closely with accurate CCSD(T) benchmark data," ChemPhysChem, **10**, 282 (2009).

[12] U. Bozkaya and C. D. Sherrill, "Orbital-optimized MP2.5 and its analytic gradients: Approaching CCSD(T) quality for non-covalent interactions," J. Chem. Phys., **141**, 204105 (2014).

[13] U. Bozkaya, "Derivation of general analytic gradient expressions for density-fitted post-Hartree–Fock methods: An efficient implementation for the density-fitted second-order Moller–Plesset perturbation theory," J. Chem. Phys., **141**, 124108 (2014).

[14] M. L. Leininger, W. D. Allen, H. F. Schaefer III, and C. D. Sherrill, "Is Møller–Plesset perturbation theory a convergent *ab initio* method?" J. Chem. Phys., **112**, 9213 (2000).

[15] S. E. Wheeler, W. D. Allen, and H. F. Schaefer III, "On the convergence of Z-averaged perturbation theory," J. Chem. Phys., **128**, 074107 (2008).

[16] T. J. Lee and D. Jayatilaka, "An open-shell restricted Hartree-Fock perturbation-theory based on symmetrical spin orbitals," Chem. Phys. Lett., **201**, 1 (1993).

[17] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory* (McGraw-Hill, New York, 1989).

[18] U. Bozkaya, "The extended Koopmans' theorem for orbital-optimized methods: Accurate computation of ionization potentials," J. Chem. Phys., **139**, 154105 (2013).

[19] U. Bozkaya, "Accurate electron affinities from the extended Koopmans' theorem based on orbital-optimized methods," J. Chem. Theory Comput., **10**, 2041 (2014).

[20] U. Bozkaya and A. Ünal, "State-of-the-art computations of vertical ionization potentials with the extended Koopmans' theorem integrated with the CCSD(T) method," J. Phys. Chem. A, **122**, 4375 (2018).

[21] A. E. DePrince III and C. D. Sherrill, "Accuracy and efficiency of coupled-cluster theory using density fitting/Cholesky decomposition, frozen natural orbitals, and a t_1-transformed Hamiltonian," J. Chem. Theory Comput., **9**, 2687 (2013).

[22] U. Bozkaya and C. D. Sherrill, "Analytic energy gradients for the coupled-cluster singles and doubles method with the density-fitting approximation," J. Chem. Phys., **144**, 174103 (2016).

[23] U. Bozkaya and C. D. Sherrill, "Analytic energy gradients for the coupled-cluster singles and doubles with perturbative triples method with the density-fitting approximation," J. Chem. Phys., **147**, 044104 (2017).

[24] O. Christiansen, H. Koch, and P. Jørgensen, "The second-order approximate coupled cluster singles and doubles model CC2," Chem. Phys. Lett., **243**, 409 (1995), ISSN 0009-2614.

[25] H. Koch, O. Christiansen, P. Jørgensen, A. M. Sanchez de Merás, and T. Helgaker, "The CC3 model: An iterative coupled cluster approach including connected triples," J. Chem. Phys., **106**, 1808 (1997).

[26] A. E. DePrince III and C. D. Sherrill, "Accurate noncovalent interaction energies using truncated basis sets based on frozen natural orbitals," J. Chem. Theory Comput., **9**, 293 (2013).

[27] C. Sosa, J. Geersten, G. W. Trucks, R. J. Barlett, and J. A. Franz, "Selection of the reduced virtual space for correlated calculations - an application to the energy and dipole-moment of H_2O," Chem. Phys. Lett., **159**, 148 (1989).

[28] W. Klopper, J. Noga, H. Koch, and T. Helgaker, "Multiple basis sets in calculations of triples corrections in coupled-cluster theory," Theor. Chem. Acc., **97**, 164 (1997).

[29] A. G. Taube and R. J. Bartlett, "Frozen natural orbitals: Systematic basis set truncation for coupled-cluster theory," Collect. Czech. Chem. Commun., **70**, 837 (2005).

[30] A. Landau, K. Khistyaev, S. Dolgikh, and A. I. Krylov, "Frozen natural orbitals for ionized states within equation-of-motion coupled-cluster formalism," J. Chem. Phys., **132**, 014109 (2010).

[31] J. Geertsen, M. Rittby, and R. J. Bartlett, "The equation-of-motion coupled-cluster method: Excitation energies of Be and CO," Chem. Phys. Lett., **164**, 57 (1989), ISSN 0009-2614.

[32] J. F. Stanton and R. J. Bartlett, "The equation of motion coupled-cluster method. a systematic biorthogonal approach to molecular excitation energies, transition probabilities, and excited state properties," J. Chem. Phys., **98**, 7029 (1993).

[33] C. E. Smith, R. A. King, and T. D. Crawford, "Coupled cluster methods including triple excitations for excited states of radicals," J. Chem. Phys., **122**, 054110 (2005).

[34] T. D. Crawford and P. J. Stephens, "Comparison of time-dependent density-functional theory and coupled cluster theory for the calculation of the optical rotations of chiral molecules," J. Phys. Chem. A, **112**, 1339 (2008).

[35] M. Kállay, P. R. Nagy, D. Mester, Z. Rolik, G. Samu, J. Csontos, J. Csóka, P. B. Szabó, L. Gyevi-Nagy, B. Hégely, I. Ladjánszki, L. Szegedy, B. Ladóczki, K. Petrov, M. Farkas, P. D. Mezei, and

Á. Ganyecz, "The MRCC program system: Accurate quantum chemistry from water to proteins," J. Chem. Phys., **152**, 074107 (2020).

[36] U. Bozkaya, J. M. Turney, Y. Yamaguchi, H. F. Schaefer III, and C. D. Sherrill, "Quadratically convergent algorithm for orbital optimization in the orbital-optimized coupled-cluster doubles method and in orbital-optimized second-order Møller–Plesset perturbation theory," J. Chem. Phys., **135**, 104103 (2011).

[37] U. Bozkaya, "Orbital-optimized third-order Møller–Plesset perturbation theory and its spin-component and spin-opposite scaled variants: Application to symmetry breaking problems," J. Chem. Phys., **135**, 224103 (2011).

[38] U. Bozkaya and C. D. Sherrill, "Analytic energy gradients for the orbital-optimized second-order Møller–Plesset perturbation theory," J. Chem. Phys., **138**, 184103 (2013).

[39] U. Bozkaya and C. D. Sherrill, "Orbital-optimized coupled-electron pair theory and its analytic gradients: Accurate equilibrium geometries, harmonic vibrational frequencies, and hydrogen transfer reactions," J. Chem. Phys., **139**, 054104 (2013).

[40] U. Bozkaya, "Orbital-optimized second-order perturbation theory with density-fitting and Cholesky decomposition approximations: An efficient implementation," J. Chem. Theory Comput., **10**, 2371 (2014).

[41] U. Bozkaya, "Orbital-optimized MP3 and MP2.5 with density-fitting and Cholesky decomposition approximations," J. Chem. Theory Comput., **12**, 1179 (2016).

[42] U. Bozkaya, "Orbital-optimized linearized coupled-cluster doubles with density-fitting and Cholesky decomposition approximations: An efficient implementation," Phys. Chem. Chem. Phys., **18**, 11362 (2016).

[43] U. Bozkaya and C. D. Sherrill, "Analytic energy gradients for orbital-optimized MP3 and MP2.5 with the density-fitting approximation: An efficient implementation," J. Comput. Chem., **39**, 351 (2018).

[44] E. G. Hohenstein and C. D. Sherrill, "Density fitting and Cholesky decomposition approximations in symmetry-adapted perturbation theory: Implementation and application to probe the nature of π-π interactions in linear acenes," J. Chem. Phys., **132**, 184111 (2010).

[45] E. G. Hohenstein, R. M. Parrish, C. D. Sherrill, J. M. Turney, and H. F. Schaefer III, "Large-scale symmetry-adapted perturbation theory computations via density fitting and Laplace transformation techniques: Investigating the fundamental forces of DNA-intercalator interactions," J. Chem. Phys., **135**, 174107 (2011).

[46] E. G. Hohenstein and C. D. Sherrill, "Density fitting of intra-monomer correlation effects in symmetry-adapted perturbation theory," J. Chem. Phys., **133**, 014101 (2010).

[47] E. G. Hohenstein and C. D. Sherrill, "Efficient evaluation of triple excitations in symmetry-adapted perturbation theory via second-order Møller–Plesset perturbation theory natural orbitals," J. Chem. Phys., **133**, 104107 (2010).

[48] R. M. Parrish, E. G. Hohenstein, and C. D. Sherrill, "Tractability gains in symmetry-adapted perturbation theory including coupled double excitations: CCD+ST(CCD) dispersion with natural orbital truncations," J. Chem. Phys., **139**, 174102 (2013).

[49] J. F. Gonthier and C. D. Sherrill, "Density-fitted open-shell symmetry-adapted perturbation theory and application to π-stacking in benzene dimer cation and ionized DNA base pair steps," J. Chem. Phys., **145**, 134106 (2016).

[50] M. Hapka, P. S. Żuchowski, M. M. Szczęśniak, and G. Chałasiński, "Symmetry-adapted perturbation theory based on unrestricted Kohn–Sham orbitals for high-spin open-shell van der Waals complexes," J. Chem. Phys., **137**, 164104 (2012).

[51] P. S. Żuchowski, R. Podeszwa, R. Moszyński, B. Jeziorski, and K. Szalewicz, "Symmetry-adapted perturbation theory utilizing density functional description of monomers for high-spin open-shell complexes," J. Chem. Phys., **129**, 084101 (2008).

[52] R. M. Parrish and C. D. Sherrill, "Spatial assignment of symmetry adapted perturbation theory interaction energy components: The atomic SAPT partition," J. Chem. Phys., **141**, 044115 (2014).

[53] R. M. Parrish, T. M. Parker, and C. D. Sherrill, "Chemical assignment of symmetry-adapted perturbation theory interaction energy components: The functional-group SAPT partition," J. Chem. Theory Comput., **10**, 4417 (2014).

[54] R. M. Parrish, J. F. Gonthier, C. Corminboeuf, and C. D. Sherrill, "Communication: Practical intramolecular symmetry adapted perturbation theory via Hartree–Fock embedding," J. Chem. Phys., **143**, 051103 (2015).

[55] J. A. Pople, M. Head-Gordon, and K. Raghavachari, "Quadratic configuration interaction. a general technique for determining electron correlation energies," J. Chem. Phys., **87**, 5968 (1987).

[56] C. D. Sherrill and H. F. Schaefer III, "The configuration interaction method: Advances in highly correlated approaches," in *Advances in Quantum Chemistry*, Vol. 34, edited by P.-O. Löwdin (Academic Press, New York, 1999) pp. 143–269.

[57] J. Olsen, B. O. Roos, P. Jørgensen, and H. J. Aa. Jensen, "Determinant based configuration interaction algorithms for complete and restricted configuration interaction spaces," J. Chem. Phys., **89**, 2185 (1988).

[58] B. O. Roos, P. R. Taylor, and P. E. M. Siegbahn, "A complete active space SCF method (CASSCF) using a density matrix formulated super-CI approach," Chem. Phys., **48**, 157 (1980).

[59] K. Ruedenberg, M. W. Schmidt, M. M. Gilbert, and S. T. Elbert, "Are atoms intrinsic to molecular electronic wavefunctions? I. the FORS model," Chem. Phys., **71**, 41 (1982).

[60] P.-Å. Malmqvist, A. Rendell, and B. O. Roos, "The restricted active space self-consistent-field method, implemented with a split graph unitary group approach," J. Phys. Chem., **94**, 5477 (1990).

[61] S. R. White and R. L. Martin, "Ab initio quantum chemistry using the density matrix renormalization group," J. Chem. Phys., **110**, 4127 (1999).

[62] G. K. L. Chan and M. Head-Gordon, "Highly correlated calculations with a polynomial cost algorithm: A study of the density matrix renormalization group," J. Chem. Phys., **116**, 4462 (2002).

[63] S. Wouters, T. Bogaerts, P. V. der Voort, V. V. Speybroeck, and D. Van Neck, "Communication: DMRG-SCF study of the singlet, triplet, and quintet states of Oxo-Mn(Salen)," J. Chem. Phys., **140**, 241103 (2014).

[64] S. Wouters, V. V. Speybroeck, and D. Van Neck, "DMRG-CASPT2 study of the longitudinal static second hyperpolarizability of all-trans polyenes," J. Chem. Phys., **145**, 054120 (2016).

[65] F. A. Evangelista, E. Prochnow, J. Gauss, and H. F. Schaefer III, "Perturbative triples corrections in state-specific multireference coupled cluster theory," J. Chem. Phys., **132**, 074107 (2010).

[66] F. A. Evangelista, A. C. Simmonett, H. F. Schaefer III, D. Mukherjee, and W. D. Allen, "A companion perturbation theory for state-specific multireference coupled cluster methods," Phys. Chem. Chem. Phys., **11**, 4728 (2009).

[67] M. Kállay, Z. Rolik, J. Csontos, I. Ladjánski, L. Szegedy, B. Ladóczki, G. Samu, K. Petrov, M. Farkas, P. Nagy, D. Mester, and B. Hégely, "MRCC, a quantum chemical program suite. see: http://www.mrcc.hu.".

[68] J. B. Schriber, K. P. Hannon, C. Li, and F. A. Evangelista, "A combined selected configuration interaction and many-body treatment of static and dynamical correlation in oligoacenes," J. Chem. Theory Comput., **14**, 6295 (2018).

[69] C. Li and F. A. Evangelista, "Multireference theories of electron correlation based on the driven similarity renormalization group," Ann. Rev. Phys. Chem., **70**, 245 (2019).

[70] J. B. Schriber, K. Hannon, York, T. Zhang, and F. A. Evangelista, Forte: a suite of quantum chemistry methods for strongly correlated electrons. For the current version, see

https://github.com/evangelistalab/forte (accessed January 2020).

[71] J. Fosso-Tande and A. E. DePrince III, V2RDM_CASSCF: a variational 2-RDM-driven CASSCF plugin to Psi4. For the current version, see https://github.com/edeprince3/v2rdm_casscf (accessed January 2020).

[72] W. Kutzelnigg, "Density-cumulant functional theory," J. Chem. Phys., **125**, 171101 (2006).

[73] A. C. Simmonett, J. J. Wilke, H. F. Schaefer III, and W. Kutzelnigg, "Density cumulant functional theory: First implementation and benchmark results for the DCFT-06 model," J. Chem. Phys., **133**, 174122 (2010).

[74] A. Y. Sokolov, A. C. Simmonett, and H. F. Schaefer III, "Density cumulant functional theory: The DC-12 method, an improved description of the one-particle density matrix," J. Chem. Phys., **138**, 024107 (2013).

[75] A. Y. Sokolov and H. F. Schaefer III, "Orbital-optimized density cumulant functional theory," J. Chem. Phys., **139**, 204110 (2013).

[76] A. Y. Sokolov, H. F. Schaefer III, and W. Kutzelnigg, "Density cumulant functional theory from a unitary transformation: N-representability, three-particle correlation effects, and application to O_4^+," J. Chem. Phys., **141**, 074111 (2014).

[77] A. V. Copan, A. Y. Sokolov, and H. F. Schaefer III, "Benchmark Study of Density Cumulant Functional Theory: Thermochemistry and Kinetics," J. Chem. Theory Comput., **10**, 2389 (2014).

[78] J. W. Mullinax, A. Y. Sokolov, and H. F. Schaefer III, "Can Density Cumulant Functional Theory Describe Static Correlation Effects?" J. Chem. Theory Comput., **11**, 2487 (2015).

[79] A. Y. Sokolov, J. J. Wilke, A. C. Simmonett, and H. F. Schaefer III, "Analytic gradients for density cumulant functional theory: The DCFT-06 model," J. Chem. Phys., **137**, 054105 (2012).

[80] A. Wolf, M. Reiher, and B. A. Hess, "The generalized Douglas–Kroll transformation," J. Chem. Phys., **117**, 9215 (2002).

[81] M. Reiher and A. Wolf, "Exact decoupling of the Dirac Hamiltonian. ii. the generalized Douglas–Kroll–Hess transformation up to arbitrary order," J. Chem. Phys., **121**, 10945 (2004).

[82] K. G. Dyall, "Interfacing relativistic and nonrelativistic methods. I. Normalized elimination of the small component in the modified Dirac equation," J. Chem. Phys., **106**, 9618 (1997).

[83] K. G. Dyall, "Interfacing relativistic and nonrelativistic methods. IV. One- and two-electron scalar approximations," J. Chem. Phys., **115**, 9136 (2001).

[84] W. Kutzelnigg, "Relativistic one-electron Hamiltonians 'for electrons only' and the variational treatment of the Dirac equation," Chem. Phys., **225**, 203 (1997).

[85] W. Kutzelnigg and W. Liu, "Quasirelativistic theory equivalent to fully relativistic theory," J. Chem. Phys., **123**, 241102 (2005).

[86] W. Kutzelnigg and W. Liu, "Quasirelativistic theory I. Theory in terms of a quasi-relativistic operator," Mol. Phys., **104**, 2225 (2006).

[87] W. Liu and W. Kutzelnigg, "Quasirelativistic theory. II. Theory at matrix level," J. Chem. Phys., **126**, 114107 (2007).

[88] W. Liu and D. Peng, "Exact two-component Hamiltonians revisited," J. Chem. Phys., **131**, 031104 (2009).

[89] M. Iliaš and T. Saue, "An infinite-order two-component relativistic Hamiltonian by a simple one-step transformation," J. Chem. Phys., **126**, 064102 (2007).

[90] W. Zou, M. Filatov, and D. Cremer, "Development and application of the analytical energy gradient for the normalized elimination of the small component method," J. Chem. Phys., **134**, 244117 (2011).

[91] L. Cheng and J. Gauss, "Analytic energy gradients for the spin-free exact two-component theory using an exact block diagonalization for the one-electron Dirac Hamiltonian," J. Chem. Phys., **135**, 084114 (2011).

[92] P. Verma, W. D. Derricotte, and F. A. Evangelista, "Predicting Near Edge X-ray Absorption Spectra with the Spin-Free Exact-Two-Component Hamiltonian and Orthogonality Constrained Density Functional Theory." J. Chem. Theory Comput., **12**, 144 (2016).

[93] A. L. L. East and W. D. Allen, "The heat of formation of NCO," J. Chem. Phys., **99**, 4638 (1993).

[94] A. G. Császár, W. D. Allen, and H. F. Schaefer III, "In pursuit of the *ab initio* limit for conformational energy prototypes," J. Chem. Phys., **108**, 9751 (1998).

[95] P. Kraus and I. Frank, "Validating additive correction schemes against gradient-based extrapolations," Int. J. Quantum Chem., **119**, e25953 (2019).

[96] S. F. Boys and F. Bernardi, "The calculation of small molecular interactions by the differences of separate total energies. Some procedures with reduced errors," Mol. Phys., **19**, 553 (1970).

[97] B. H. Wells and S. Wilson, "Van der Waals interaction potentials: Many-body basis set superposition effects," Chemical physics letters, **101**, 429 (1983).

[98] P. Valiron and I. Mayer, "Hierarchy of counterpoise corrections for N-body clusters: Generalization of the Boys-Bernardi scheme," Chem. Phys. Lett., **275**, 46 (1997).

[99] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," Computing in Science Engineering, **13**, 22 (2011), ISSN 1521-9615.

[100] D. G. A. Smith, Psi4NumPy: combining Psi4 and NumPy for education and development. For the current version, see https://github.com/psi4/psi4numpy (accessed January 2020).

[101] O. J. Backhouse, M. Nusspickel, and G. H. Booth, "Wave function perspective and efficient truncation of renormalized second-order perturbation theory," J. Chem. Theory Comput., **16**, 1090 (2020).

[102] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, "An adaptive variational algorithm for exact molecular simulations on a quantum computer," Nat. Commun., 3007 (2019).

[103] M. Kodrycka, C. Holzer, W. Klopper, and K. Patkowski, "Explicitly correlated dispersion and exchange dispersion energies in symmetry-adapted perturbation theory," J. Chem. Theory Comput., **15**, 5965 (2019).

[104] D. Claudino and N. J. Mayhall, "Simple and efficient truncation of virtual spaces in embedded wave functions via concentric localization," J. Chem. Theory Comput., **15**, 6085 (2019).

[105] W. D. Derricotte, "Symmetry-adapted perturbation theory decomposition of the reaction force: Insights into substituent effects involved in hemiacetal formation mechanisms," J. Phys. Chem. A, **123**, 7881 (2019).

[106] T. Zhang, C. Li, and F. A. Evangelista, "Improving the efficiency of the multireference driven similarity renormalization group via sequential transformation, density fitting, and the noninteracting virtual orbital approximation," J. Chem. Theory Comput., **15**, 4399 (2019).

[107] J. M. Waldrop and K. Patkowski, "Spin splittings from first-order symmetry-adapted perturbation theory without single-exchange approximation," J. Chem. Phys., **150**, 074109 (2019).

[108] J. A. Rackers and J. W. Ponder, "Classical Pauli repulsion: An anisotropic, atomic multipole model," J. Chem. Phys., **150**, 084104 (2019).

[109] H. E. Sauceda, S. Chmiela, I. Poltavsky, K.-R. Müller, and A. Tkatchenko, "Molecular force fields with gradient-domain machine learning: Construction and application to dynamics of small molecules with coupled cluster forces," J. Chem. Phys., **150**, 114102 (2019).

[110] J. T. Margraf, C. Kunkel, and K. Reuter, "Towards density functional approximations from coupled cluster correlation energy densities," J. Chem. Phys., **150**, 244116 (2019).

[111] T. D. Crawford, A. Kumar, A. P. Bazanté, and R. Di Remigio, "Reduced-scaling coupled cluster response theory: Challenges and opportunities," WIREs Comput. Mol. Sci., **9**, e1406 (2019).

[112] C. Zanchi, G. Longhi, S. Abbate, G. Pellegrini, P. Biagioni, and M. Tommasini, "Evaluation of molecular polarizability and of intensity carrying modes contributions in circular dichroism spectroscopies," Applied Sciences, **9**, 4691 (2019), ISSN 2076-3417.

[113] M. F. Herbst, M. Scheurer, T. Fransson, D. R. Rehn, and A. Dreuw, "adcc: A versatile toolkit for rapid development of algebraic-diagrammatic construction methods," WIREs Comput. Mol. Sci., e1462 (2020).

[114] Z. Rinkevicius, X. Li, O. Vahtras, K. Ahmadzadeh, M. Brand, M. Ringholm, N. H. List, M. Scheurer, M. Scott, A. Dreuw, and P. Norman, "VeloxChem: A python-driven density-functional theory program for spectroscopy simulations in high-performance computing environments," WIREs Comput. Mol. Sci., e1457 (2019).

[115] A. Alenaizan, L. A. Burns, and C. D. Sherrill, "Python implementation of the restrained electrostatic potential charge model," Int. J. Quantum Chem., **120**, e26035 (2020).

[116] S. E. Houck and N. J. Mayhall, "A combined spin-flip and IP/EA approach for handling spin and spatial degeneracies: Application to double exchange systems," J. Chem. Theory Comput., **15**, 2278 (2019).

[117] J. Townsend and K. D. Vogiatzis, "Data-driven acceleration of the coupled-cluster singles and doubles iterative solver," J. Phys. Chem. Lett., **10**, 4129 (2019).

[118] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt (IOS Press, 2016) pp. 87 – 90.

[119] A. Krylov, T. L. Windus, T. Barnes, E. Marin-Rimoldi, J. A. Nash, B. Pritchard, D. G. A. Smith, D. Altarawy, P. Saxe, C. Clementi, T. D. Crawford, R. J. Harrison, S. Jha, V. S. Pande, and T. Head-Gordon, "Perspective: Computational chemistry software and its advancement as illustrated through three grand challenge cases for molecular science," J. Chem. Phys., **149**, 180901 (2018).

[120] D. G. A. Smith, L. A. Burns, D. Altarawy, L. Naden, and M. Welborn, QCArchive: A central source to compile, aggregate, query, and share quantum chemistry data. `https://qcarchive.molssi.org` (accessed January 2020). ().

[121] D. G. A. Smith, D. Altarawy, L. A. Burns, M. Welborn, L. N. Naden, L. Ward, and S. Ellis, "The MolSSI QCArchive project: An open-source platform to compute, organize, and share quantum chemistry data," WIREs Comput. Mol. Sci., in review (2020).

[122] R. C. Fortenberry, A. R. McDonald, T. D. Shepherd, M. Kennedy, and C. D. Sherrill, "PSI4education: Computational chemistry labs using free software," in *The Promise of Chemical Education: Addressing our Students' Needs*, Vol. 1193, edited by K. Daus and R. Rigsby (American Chemical Society, Washington, D.C., 2015) pp. 85–98.

[123] D. A. Sirianni, A. Alenaizan, D. L. Cheney, and C. D. Sherrill, "Assessment of density functional methods for geometry optimization of biomolecular van der Waals complexes," J. Chem. Theory Comput., **14**, 3004 (2018).

[124] L. A. Burns, J. C. Faver, Z. Zheng, M. S. Marshall, D. G. A. Smith, K. Vanommeslaeghe, A. D. MacKerell, K. M. Merz, and C. D. Sherrill, "The biofragment database (BFDb): An open-data platform for computational chemistry analysis of noncovalent interactions," J. Chem. Phys., **147**, 161727 (2017).

[125] D. G. A. Smith, B. de Jong, L. A. Burns, G. Hutchison, and M. D. Hanwell, QCSchema: a schema for quantum chemistry. For the current version, see `https://github.com/MolSSI/QCSchema` (accessed January 2020). ().

[126] D. G. A. Smith, L. A. Burns, L. Naden, and M. Welborn, QCElemental: periodic table, physical constants, and molecule parsing for quantum chemistry. For the current version, see `https://github.com/MolSSI/QCElemental` (accessed January 2020). ().

[127] D. G. A. Smith, S. Lee, L. A. Burns, and M. Welborn, QCEngine: quantum chemistry program executor and IO standardizer (QCSchema). For the current version, see `https://github.com/MolSSI/QCEngine` (accessed January 2020). ().

[128] D. G. A. Smith, M. Welborn, D. Altarawy, and L. Naden, QCFractal: a distributed compute and database platform for quantum chemistry. For the current version, see `https://github.com/MolSSI/QCFractal` (accessed January 2020). ().

[129] J. P. Kenny, C. L. Janssen, E. F. Valeev, and T. L. Windus, "Components for integral evaluation in quantum chemistry," J. Comput. Chem., **29**, 562 (2008).

[130] I. Naoki, MessagePack-Python: MessagePack serializer implementation for Python. For the current version, see `https://github.com/msgpack/msgpack-python` (accessed January 2020). For the originating project, see `https://msgpack.org/`.

[131] S. Lehtola, C. Steigemann, M. J. Oliveira, and M. A. Marques, "Recent developments in libxc — a comprehensive library of functionals for density functional theory," SoftwareX, **7**, 1 (2018), ISSN 2352-7110.

[132] N. Mardirossian and M. Head-Gordon, "ωB97M-V: A combinatorially optimized, range-separated hybrid, meta-GGA density functional with VV10 nonlocal correlation," J. Chem. Phys., **144**, 214110 (2016).

[133] J. Sun, A. Ruzsinszky, and J. P. Perdew, "Strongly constrained and appropriately normed semilocal density functional," Phys. Rev. Lett., **115**, 036402 (2015).

[134] Due to a memory estimation error (since corrected), part of the v1.3.2 B3LYP computation uses the less-efficient `Disk_DF` algorithm for SCF, even though the job fits in memory, hence the non-monotonic decrease in timings with respect to releases. Figures like this are now routinely constructed before releases to prevent similar regressions in the future.

[135] P. Jurečka, J. Šponer, J. Černý, and P. Hobza, "Benchmark database of accurate (MP2 and CCSD(T) complete basis set limit) interaction energies of small model complexes, DNA base pairs, and amino acid pairs," Phys. Chem. Chem. Phys., **8**, 1985 (2006).

[136] J. Řezáč and P. Hobza, "Describing Noncovalent Interactions beyond the Common Approximations: How Accurate Is the "Gold Standard," CCSD(T) at the Complete Basis Set Limit?" J. Chem. Theory Comput., **9**, 2151 (2013), ISSN 1549-9618, 1549-9626.

[137] R. Sure and S. Grimme, "Corrected small basis set Hartree–Fock method for large systems," J. Comput. Chem., **34**, 1672 (2013), ISSN 1096-987X.

[138] S. Grimme, J. G. Brandenburg, C. Bannwarth, and A. Hansen, "Consistent structures and interactions by density functional theory with small atomic orbital basis sets," J. Chem. Phys., **143**, 054107 (2015).

[139] S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, DFTD3: dispersion correction for DFT, Hartree–Fock, and semi-empirical quantum chemical methods. For the current version, see `https://github.com/loriab/dftd3` (accessed January 2020). For the originating project, see `https://www.chemie.uni-bonn.de/pctc/mulliken-center/software/dft-d3`.% BibitemShut Stop

[140] H. Kruse and S. Grimme, gCP: geometrical counterpoise correction for DFT and Hartree–Fock quantum chemical methods. For the current version, see `https://www.chemie.uni-bonn.de/pctc/mulliken-center/software/gcp/gcp` (accessed January 2020).

[141] S. Grimme, "Semiempirical GGA-type density functional constructed with a long-range dispersion correction," J. Comput. Chem., **27**, 1787 (2006).

[142] S. Grimme, J. Antony, S. Ehrlich, and H. Krieg, "A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu," J. Chem. Phys., **132**, 154104 (2010).

[143] D. G. A. Smith, L. A. Burns, K. Patkowski, and C. D. Sherrill, "Revised damping parameters for the D3 dispersion correction to density functional theory," J. Phys. Chem. Lett., **7**, 2197 (2016).

[144] J. Řezáč, C. Greenwell, and G. J. O. Beran, "Accurate noncovalent interactions via dispersion-corrected second-order Møller–

[144]Plesset perturbation theory," J. Chem. Theory Comput., **14**, 4711 (2018), ISSN 1549-9618.

[145]C. Greenwell, MP2D: a program for calculating the MP2D dispersion energy. For the current version, see `https://github.com/Chandemonium/MP2D` (accessed January 2020).

[146]W. Hujo and S. Grimme, "Performance of the van der Waals density functional VV10 and (hybrid)GGA variants for thermochemistry and noncovalent interactions," J. Chem. Theory Comput., **7**, 3866 (2011).

[147]D. A. Sirianni, D. G. A. Smith, L. A. Burns, D. F. Sitkoff, D. L. Cheney, and C. D. Sherrill, "Optimized Damping Parameters for Empirical Dispersion Corrections to Symmetry-Adapted Perturbation Theory," *in preparation*.

[148]T. Warne, P. C. Edwards, A. S. Doré, A. G. W. Leslie, and C. G. Tate, "Molecular basis for high-affinity agonist binding in GPCRs," Science, **364**, 775 (2019), ISSN 0036-8075.

[149]A. J. Misquitta, R. Podeszwa, B. Jeziorski, and K. Szalewicz, "Intermolecular potentials based on symmetry-adapted perturbation theory with dispersion energies from time-dependent density-functional calculations," J. Chem. Phys., **123**, 214103 (2005).

[150]A. Heßelmann, G. Jansen, and M. Schütz, "Density-functional theory-symmetry-adapted intermolecular perturbation theory with density fitting: A new efficient method to study intermolecular interaction energies," J. Chem. Phys., **122**, 014103 (2005).

[151]H. L. Williams and C. F. Chabalowski, "Using Kohn–Sham orbitals in symmetry-adapted perturbation theory to investigate intermolecular interactions," J. Phys. Chem. A, **105**, 646 (2001).

[152]M. Grüning, O. V. Gritsenko, S. J. A. van Gisbergen, and E. J. Baerends, "Shape corrections to exchange-correlation potentials by gradient-regulated seamless connection of model potentials for inner and outer region," J. Chem. Phys., **114**, 652 (2001).

[153]A. Hesselmann and G. Jansen, "Intermolecular dispersion energies from time-dependent density functional theory," Chem. Phys. Lett., **367**, 778 (2003).

[154]R. Podeszwa, R. Bukowski, and K. Szalewicz, "Density-fitting method in symmetry-adapted perturbation theory based on Kohn-Sham description of monomers," J. Chem. Theory Comput., **2**, 400 (2006).

[155]R. Bukowski, R. Podeszwa, and K. Szalewicz, "Efficient calculation of coupled Kohn–Sham dynamic susceptibility functions and dispersion energies with density fitting," Chem. Phys. Lett., **414**, 111 (2005).

[156]G. Jansen, "Symmetry-adapted perturbation theory based on density functional theory for noncovalent interactions," WIREs Comput. Mol. Sci., **4**, 127 (2014).

[157]R. Schäffer and G. Jansen, "Intermolecular exchange-induction energies without overlap expansion," Theor. Chem. Acc., **131**, 1235 (2012).

[158]R. Schäffer and G. Jansen, "Single-determinant-based symmetry-adapted perturbation theory without single-exchange approximation," Mol. Phys., **111**, 2570 (2013).

[159]K. Patkowski, "Recent developments in symmetry-adapted perturbation theory," WIREs Comput. Mol. Sci., e1452 (2020).

[160]K. U. Lao, R. Schäffer, G. Jansen, and J. M. Herbert, "Accurate description of intermolecular interactions involving ions using symmetry-adapted perturbation theory," J. Chem. Theory Comput., **11**, 2473 (2015).

[161]T. M. Parker, L. A. Burns, R. M. Parrish, A. G. Ryno, and C. D. Sherrill, "Levels of symmetry adapted perturbation theory (SAPT). I. efficiency and performance for interaction energies," J. Chem. Phys., **140**, 094106 (2014).

[162]K. Patkowski, P. S. Żuchowski, and D. G. A. Smith, "First-order symmetry-adapted perturbation theory for multiplet splittings," J. Chem. Phys., **148**, 164110 (2018).

[163]E. F. Valeev and J. T. Fermann, Libint: a library for the evaluation of molecular integrals of many-body operators over Gaussian functions, For the current version, see `https://github.com/evaleev/libint/tree/v1` (accessed January 2020).

[164]E. F. Valeev, Libint: a library for the evaluation of molecular integrals of many-body operators over Gaussian functions, For the current version, see `https://github.com/evaleev/libint` (accessed January 2020). For the originating project, see `http://libint.valeyev.net/`.

[165]B. P. Pritchard and E. Chow, "Horizontal vectorization of electron repulsion integrals," J. Comp. Chem., **37**, 2537 (2016), ISSN 1096-987X.

[166]H. Huang and E. Chow, "Accelerating quantum chemistry with vectorized and batched integrals," in *SC18: The International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE Press, 2018) pp. 1–14.

[167]J. Almlöf, K. Faegri Jr., and K. Korsell, "Principles for a direct SCF approach to LICAO–MO ab-initio calculations," J. Comput. Chem., **3**, 385 (1982).

[168]J. H. Van Lenthe, R. Zwaans, H. J. J. Van Dam, and M. F. Guest, "Starting SCF calculations by superposition of atomic densities," J. Comput. Chem., **27**, 926 (2006).

[169]M. Wolfsberg and L. Helmholz, "The spectra and electronic structure of the tetrahedral ions MnO_4^-, CrO_4^{--}, and ClO_4^-," J. Chem. Phys., **20**, 837 (1952).

[170]S. Lehtola, "Assessment of initial guesses for self-consistent field calculations. superposition of atomic potentials: Simple yet efficient," J. Chem. Theory Comput., **15**, 1593 (2019).

[171]S. Lehtola, "Fully numerical Hartree–Fock and density functional calculations. I. Atoms," Int. J. Quantum Chem., **119**, e25945 (2019).

[172]S. Lehtola, "Fully numerical calculations on atoms with fractional occupations and range-separated exchange functionals," Phys. Rev. A, **101**, 012516 (2020).

[173]S. Lehtola, "Curing basis set overcompleteness with pivoted Cholesky decompositions," J. Chem. Phys., **151**, 241102 (2019).

[174]A. Dreuw and M. Head-Gordon, "Single-reference *ab initio* methods for the calculation of excited states of large molecules," Chem. Rev., **105**, 4009 (2005), ISSN 0009-2665.

[175]R. E. Stratmann, G. E. Scuseria, and M. J. Frisch, "An efficient implementation of time-dependent density-functional theory for the calculation of excitation energies of large molecules," J. Chem. Phys., **109**, 8218 (1998), ISSN 0021-9606, 1089-7690.

[176]E. R. Davidson, "The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices," J. Comp. Phys., **17**, 87 (1975), ISSN 0021-9991.

[177]P. Norman, K. Ruud, and T. Saue, *Principles and Practices of Molecular Properties: Theory, Modeling, and Simulations* (John Wiley & Sons, 2018) ISBN 9780470725627.

[178]M. Marques, S. Lehtola, M. Oliveira, X. Andrade, and D. Strubbe, LibXC: a library of exchange-correlation functionals for density-functional theory. For the current version, see `https://gitlab.com/libxc/libxc` (accessed January 2020). ().

[179]D. G. A. Smith, L. A. Burns, and A. C. Simmonett, gau2grid: fast computation of a gaussian and its derivative on a grid. For the current version, see `https://github.com/dgasmith/gau2grid` (accessed January 2020). ().

[180]A. Wolf, M. Reiher, and B. A. Hess, dkh: Wolf, Reiher, and Hess's Douglas-Kroll-Hess relativistic correction. For the current version, see `https://github.com/psi4/dkh` (accessed January 2020). For originating project, see `http://www.reiher.ethz.ch/software/dkh-x2c.html`.

[181]I. Kaliman, libefp: parallel implementation of the effective fragment potential method. For the current version, see `https://github.com/ilyak/libefp` (accessed January 2020).

[182]I. A. Kaliman and L. V. Slipchenko, "LIBEFP: A new parallel implementation of the effective fragment potential method as a portable software library," J. Comput. Chem., **34**, 2284 (2013).

[183]A. J. Stone, GDMA: a program to perform distributed multipole analysis. For the current version, see `https://github.com/psi4/gdma` (accessed January 2020). For originating project, see `http://www-stone.ch.cam.ac.uk/programs.html`.

[184] A. J. Stone, "Distributed multipole analysis: Stability for large basis sets," J. Chem. Theory Comput., **1**, 1128 (2005), ISSN 1549-9618.

[185] S. Wouters, CHEMPS2: a spin-adapted implementation of DMRG for *ab initio* quantum chemistry. For the current version, see `https://github.com/SebWouters/CheMPS2` (accessed January 2020).

[186] S. Wouters, W. Poelmans, P. W. Ayers, and D. Van Neck, "CheMPS2: A free open-source spin-adapted implementation of the density matrix renormalization group for *ab initio* quantum chemistry," Comput. Phys. Commun., **185**, 1501 (2014).

[187] S. Wouters and D. Van Neck, "The density matrix renormalization group for *ab initio* quantum chemistry," Eur. Phys. J. D, **68**, 272 (2014).

[188] R. D. Remigio and L. Frediani, PCMSOLVER: an API for the polarizable continuum model. For the current version, see `https://github.com/PCMSolver/pcmsolver` (accessed January 2020).

[189] R. Di Remigio, K. Mozgawa, H. Cao, V. Weijo, and L. Frediani, "A Polarizable Continuum Model for Molecules at Spherical Diffuse Interfaces," J. Chem. Phys., **144**, 124103 (2016), ISSN 0021-9606, 1089-7690.

[190] N. Flocke and V. Lotrich, ERD: ACESIII electron repulsion integrals. For the current version, see `https://github.com/psi4/erd` (accessed January 2020). For originating project, see `http://www.qtp.ufl.edu/Aces/`.

[191] N. Flocke and V. Lotrich, "Efficient electronic integrals and their generalized derivatives for object oriented implementations of electronic structure calculations," J. Comput. Chem., **29**, 2722 (2008).

[192] B. P. Pritchard and E. Chow, SIMINT: a code generator for vectorized integrals. For the current version, see `https://github.com/simint-chem/simint-generator` (accessed January 2020).

[193] J. M. Turney, AMBIT: a C++ library for the implementation of tensor product calculations through a clean, concise user interface. For the current version, see `https://github.com/jturney/ambit` (accessed January 2020).

[194] L. A. Burns, PYLIBEFP: a python wrapper to libefp library for effective fragment potentials. For the current version, see `https://github.com/loriab/pylibefp` (accessed January 2020).

[195] M. Scheurer, CPPE: C++ and Python library for polarizable embedding. For the current version, see `https://github.com/maxscheurer/cppe` (accessed January 2020).

[196] M. Scheurer, P. Reinholdt, E. R. Kjellgren, J. M. H. Olsen, A. Dreuw, and J. Kongsted, "CPPE: An Open-Source C++ and Python Library for Polarizable Embedding," J. Chem. Theory Comput., **15**, 6154 (2019).

[197] M. F. Herbst and M. Scheurer, ADCC: Seamlessly connect your program to ADC. For the current version, see `https://github.com/adc-connect/adcc` (accessed January 2020).

[198] J. Fosso-Tande, T.-S. Nguyen, G. Gidofalvi, and A. E. DePrince III, "Large-scale variational two-electron reduced-density-matrix-driven complete active space self-consistent field methods," J. Chem. Theory Comput., **12**, 2260 (2016).

[199] J. E. Deustua, J. Shen, and P. Piecuch, CCT3: a PSI4 plugin which performs active-space coupled-cluster CCSDt calculations and which can determine noniterative corrections to CCSDt defining the CC(t;3) approach. For the current version, see `https://github.com/piecuch-group/psi4_cct3` (accessed January 2020).

[200] J. Shen and P. Piecuch, "Biorthogonal moment expansions in coupled-cluster theory: Review of key concepts and merging the renormalized and active-space coupled-cluster methods," Chem. Phys., **401**, 180 (2012).

[201] J. Shen and P. Piecuch, "Combining active-space coupled-cluster methods with moment energy corrections via the CC(P;Q) methodology, with benchmark calculations for biradical transition states," J. Chem. Phys., **136**, 144104 (2012).

[202] A. E. DePrince III, GPU_DFCC: GPU-accelerated coupled cluster with density fitting. For the current version, see `https://github.com/edeprince3/gpu_dfcc` (accessed January 2020).

[203] A. E. DePrince III, M. R. Kennedy, B. G. Sumpter, and C. D. Sherrill, "Density-fitted singles and doubles coupled cluster on graphics processing units," Mol. Phys., **112**, 844 (2014).

[204] Schmidt, J. R.; Polik, W. F. *WebMO 17*, WebMO, LLC: Holland, MI, 2016. `http://www.webmo.net`.

[205] G. Schaftenaar and J. H. Noordik, MOLDEN: a pre- and post-processing program for molecular and electronic structures. For the current version, see `ftp://ftp.cmbi.umcn.nl/pub/molgraph/molden` (accessed January 2020).

[206] G. Schaftenaar and J. H. Noordik, "Molden: a pre- and post-processing program for molecular and electronic structures," J. Comput. Aided Mol. Des., **14**, 123 (2000), ISSN 1573-4951.

[207] I. Y. Nikolaienko, JANPA: a freeware program package for performing natural population analysis. For the current version, see `https://sourceforge.net/projects/janpa/files/` (accessed January 2020).

[208] I. Y. Nikolaienko, L. A. Bulavin, and D. M. Hovorun, "JANPA: An open source cross-platform implementation of the natural population analysis on the Java platform," Comput. Theor. Chem., **1050**, 15 (2014).

[209] A. Ringer McDonald, D. B. Magers, F. Heidar-Zadeh, T. Shepherd, and V. H. Chavez, PSI4EDUCATION: teaching chemistry through computation. For the current version, see `https://github.com/Psi4Education/psi4education` (accessed January 2020).

[210] M. Zott, PSIOMM: an interface between PSI4 and OpenMM. For the current version, see `https://github.com/mzott/Psi4-OpenMM-Interface` (accessed January 2020).

[211] S. Doerr, J. M. Damas, and R. Galvelis, HTMD: Programming Environment for Molecular Discovery. For the current version, see `https://github.com/Acellera/htmd` and `https://github.com/Acellera/parameterize` (accessed January 2020).

[212] S. Doerr, M. J. Harvey, F. Noé, and G. D. Fabritiis, "HTMD: High-throughput molecular dynamics for molecular discovery," J. Chem. Theory Comput., **12**, 1845 (2016), ISSN 1549-9618.

[213] R. Galvelis, S. Doerr, J. M. Damas, M. J. Harvey, and G. De Fabritiis, "A scalable molecular force field parameterization method based on density functional theory and quantum-level machine learning," J. Chem. Inf. Model., **59**, 3485 (2019), ISSN 1549-9596.

[214] I. Buch, M. J. Harvey, T. Giorgino, D. P. Anderson, and G. De Fabritiis, GPUGRID: volunteer computing for biomedicine. For the current version, see `http://gpugrid.net/` (accessed January 2020).

[215] I. Buch, M. J. Harvey, T. Giorgino, D. P. Anderson, and G. De Fabritiis, "High-throughput all-atom molecular dynamics simulations using distributed computing," J. Chem. Inf. Model., **50**, 397 (2010), ISSN 1549-9596.

[216] R. T. McGibbon, SNS-MP2: spin-network-scaled MP2. For the current version, see `https://github.com/DEShawResearch/sns-mp2` (accessed January 2020).

[217] R. T. McGibbon, A. G. Taube, A. G. Donchev, K. Siva, F. Hernández, C. Hargus, K.-H. Law, J. L. Klepeis, and D. E. Shaw, "Improving the accuracy of Møller–Plesset perturbation theory with neural networks," J. Chem. Phys., **147**, 161725 (2017).

[218] A. Alenaizan, RESP: a restrained electrostatic potential (RESP) plugin to PSI4. For the current version, see `https://github.com/cdsgroup/resp` (accessed January 2020).

[219] M. Marques, S. Hu, R. Chen, and S. Wood, QISKIT-AQUA: quantum algorithms & applications in Python. For the current version, see `https://github.com/Qiskit/qiskit-aqua` (accessed January 2020). ().

[220] C. Granade and A. Paz, QUANTUM: Microsoft Quantum Development Kit Samples. For the current version, see `https://github.com/microsoft/Quantum` (accessed January 2020).

[221] C. H. Borca, CRYSTALATTE: automating the calculation of crystal lattice energies. For the current version, see `https://github.com/carlosborca/CrystaLattE` (accessed January 2020).

[222] C. H. Borca, B. W. Bakr, L. A. Burns, and C. D. Sherrill, "CrystaLattE: Automated computation of lattice energies of organic crystals exploiting the many-body expansion to achieve dual-level parallelism," J. Chem. Phys., **151**, 144103 (2019).

[223] R. Babbush, OPENFERMION: OpenFermion plugin to interface with the electronic structure package Psi4. For the current version, see https://github.com/quantumlib/OpenFermion (accessed January 2020).

[224] J. R. McClean, K. J. Sung, I. D. Kivlichan, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, P. Gokhale, T. Häner, T. Hardikar, V. Havlíček, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle, M. Neeley, T. O'Brien, B. O'Gorman, I. Ozfidan, M. D. Radin, J. Romero, N. Rubin, N. P. D. Sawaya, K. Setia, S. Sim, D. S. Steiger, M. Steudtner, Q. Sun, W. Sun, D. Wang, F. Zhang, and R. Babbush, "OpenFermion: The electronic structure package for quantum computers," (2017), arXiv:1710.07629 [quant-ph].

[225] K. J. Sung and R. Babbush, OPENFERMION-PSI4: the electronic structure package for quantum computers. For the current version, see https://github.com/quantumlib/OpenFermion-Psi4 (accessed January 2020).

[226] L. A. Burns, A. Lolinco, and Z. Glick, QCDB: quantum chemistry common driver and databases. For the current version, see https://github.com/qcdb/qcdb (accessed January 2020).

[227] A. Heide and R. A. King, OPTKING: a Python version of the Psi4 geometry optimizer. For the current version, see https://github.com/psi-rking/optking (accessed January 2020).

[228] C. Ehlert, PSIXAS: a Psi4 plugin for X-ray absorption spectra (XPS, NEXAFS, PP-NEXAFS). For the current version, see https://github.com/Masterluke87/psixas (accessed January 2020).

[229] A. H. Larsen and J. J. Mortensen, ASE: Atomic Simulation Environment: A Python library for working with atoms. For the current version, see https://gitlab.com/ase/ase (accessed January 2020).

[230] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—a python library for working with atoms," J. Phys.: Condens. Matter, **29**, 273002 (2017).

[231] M. Ceriotti, B. Hirshberg, and V. Kapil, I-PI: a universal force engine. For the current version, see https://github.com/i-pi/i-pi (accessed January 2020).

[232] V. Kapil, M. Rossi, O. Marsalek, R. Petraglia, Y. Litman, T. Spura, B. Cheng, A. Cuzzocrea, R. H. Meißner, D. M. Wilkins, B. A. Helfrecht, P. Juda, S. P. Bienvenue, W. Fang, J. Kessler, I. Poltavsky, S. Vandenbrande, J. Wieme, C. Corminboeuf, T. D. Kühne, D. E. Manolopoulos, T. E. Markland, J. O. Richardson, A. Tkatchenko, G. A. Tribello, V. V. Speybroeck, and M. Ceriotti, "i-PI 2.0: A universal force engine for advanced molecular simulations," Comp. Phys. Commun., **236**, 214 (2019), ISSN 0010-4655.

[233] T. A. Barnes, MDI: a library that enables code interoperability via the MolSSI Driver Interface. For the current version, see https://github.com/MolSSI/MDI_Library (accessed January 2020). Also, https://doi.org/10.5281/zenodo.3659285.

[234] L.-P. Wang, D. G. A. Smith, and Y. Qiu, GEOMETRIC: a geometry optimization code that includes the TRIC coordinate system. For the current version, see https://github.com/leeping/geomeTRIC (accessed January 2020).

[235] L.-P. Wang and C. Song, "Geometry optimization made simple with translation and rotation coordinates," J. Chem. Phys., **144**, 214108 (2016).

[236] A. Banerjee, J. O. Jensen, and J. Simons, "Translational and rotational symmetries in integral derivatives," J. Chem. Phys., **82**, 4566 (1985).

[237] J. O. Jensen, , A. Banerjee, and J. Simons, "Translational and rotational symmetries in integral derivatives of arbitrary order," Chem. Phys., **102**, 45 (1986).

[238] M. S. Gordon, M. A. Freitag, P. Bandyopadhyay, J. H. Jensen, V. Kairys, and W. J. Stevens, "The effective fragment potential method: A QM-based MM approach to modeling environmental effects in chemistry," J. Phys. Chem. A, **105**, 293 (2001).

[239] D. Ghosh, D. Kosenkov, V. Vanovschi, C. F. Williams, J. M. Herbert, M. S. Gordon, M. W. Schmidt, L. V. Slipchenko, and A. I. Krylov, "Noncovalent interactions in extended systems described by the effective fragment potential method: Theory and application to nucleobase oligomers," J. Phys. Chem. A, **114**, 12739 (2010).

[240] J. Schirmer, "Beyond the random-phase approximation: A new approximation scheme for the polarization propagator," Phys. Rev. A, **26**, 2395 (1982).

[241] A. B. Trofimov, I. L. Krivdina, J. Weller, and J. Schirmer, "Algebraic-diagrammatic construction propagator approach to molecular response properties," Chem. Phys., **329**, 1 (2006).

[242] A. Dreuw and M. Wormit, "The algebraic diagrammatic construction scheme for the polarization propagator for the calculation of excited states," WIREs Comput. Mol. Sci., **5**, 82 (2015).

[243] J. M. Olsen, K. Aidas, and J. Kongsted, "Excited States in Solution through Polarizable Embedding," J. Chem. Theory Comput., **6**, 3721 (2010).

[244] J. M. H. Olsen and J. Kongsted, *Chapter 3 - Molecular Properties through Polarizable Embedding*, edited by J. R. Sabin and E. Brändas, Advances in Quantum Chemistry, Vol. 61 (Academic Press, 2011) pp. 107 – 143.

[245] D. A. Mazziotti, "Variational minimization of atomic and molecular ground-state energies via the two-particle reduced density matrix," Phys. Rev. A, **65**, 062511 (2002).

[246] G. Gidofalvi and D. A. Mazziotti, "Active-space two-electron reduced-density-matrix method: Complete active-space calculations without diagonalization of the n-electron Hamiltonian," J. Chem. Phys., **129**, 134108 (2008).

[247] E. Maradzike, G. Gidofalvi, J. M. Turney, H. F. Schaefer III, and A. E. DePrince III, "Analytic energy gradients for variational two-electron reduced-density-matrix-driven complete active space self-consistent field theory," J. Chem. Theory Comput., **13**, 4113 (2017).

[248] P. Piecuch, "Active-space coupled-cluster methods," Mol. Phys., **108**, 2987 (2010).

[249] N. Oliphant and L. Adamowicz, "The implementation of the multireference coupled-cluster method based on the single-reference formalism," J. Chem. Phys., **96**, 3739 (1992).

[250] P. Piecuch, N. Oliphant, and L. Adamowicz, "A state-selective multi-reference coupled-cluster theory employing the single-reference formalism," J. Chem. Phys., **99**, 1875 (1993).

[251] P. Piecuch, S. A. Kucharski, and R. J. Bartlett, "Coupled-cluster methods with internal and semi-internal triply and quadruply excited clusters: CCSDt and CCSDtq approaches," J. Chem. Phys., **110**, 6103 (1999).

[252] P. Piecuch and M. Wloch, "Renormalized coupled-cluster methods exploiting left eigenstates of the similarity-transformed Hamiltonian," J. Chem. Phys., **123**, 224105 (2005).

[253] P. Piecuch, M. Wloch, J. R. Gour, and A. Kinal, "Single-reference, size-extensive, non-iterative coupled-cluster approaches to bond breaking and biradicals," Chem. Phys. Lett., **418**, 467 (2006).

[254] M. Wloch, J. R. Gour, and P. Piecuch, "Extension of the renormalized coupled-cluster methods exploiting left eigenstates of the similarity-transformed Hamiltonian to open-shell systems: A benchmark study," J. Phys. Chem. A, **111**, 11359 (2007).

[255] OpenForceField https://openforcefield.org.

[256] J. C. Wu, G. Chattree, and P. Ren, "Automation of AMOEBA polarizable force field parameterization for small molecules," Theor. Chem. Acc., **131**, 1138 (2012).

[257] J. G. McDaniel and J. Schmidt, "Next-generation force fields from symmetry-adapted perturbation theory," Annu. Rev. Phys. Chem., **67**, 467 (2016).

[258] J. A. Rackers, C. Liu, P. Ren, and J. W. Ponder, "A physically grounded damped dispersion model with particle mesh Ewald summation," J. Chem. Phys., **149**, 084115 (2019).

[259] C. Liu, J.-P. Piquemal, and P. Ren, "AMOEBA+ classical potential for modeling molecular interactions," J. Chem. Theory Comput., **15**, 4122 (2019).

[260] C. I. Bayly, P. Cieplak, W. D. Cornell, and P. A. Kollman, "A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: The RESP model," J. Phys. Chem., **97**, 10269 (1993).

[261] K. Franz, I. Schnell, A. Meurer, and M. Sarahan, CONDA: OS-agnostic, system-level binary package manager and ecosystem. For the current version, see `https://github.com/conda/conda` (accessed January 2020). For documentation, see `https://conda.io/en/latest/`.

[262] M. Sarahan, A. Meurer, R. Donnelly, and I. Schnell, CONDA-BUILD: commands and tools for building conda packages. For the current version, see `https://github.com/conda/conda-build` (accessed January 2020).