JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability

Robert M. Parrish,[†] Lori A. Burns,[†] Daniel G. A. Smith,[†] Andrew C. Simmonett,[‡] A. Eugene DePrince, III,[¶] Edward G. Hohenstein,[§] Uğur Bozkaya,[∥] Alexander Yu. Sokolov,[⊥] Roberto Di Remigio,[#] Ryan M. Richard,[†] Jérôme F. Gonthier,[†] Andrew M. James,[@] Harley R. McAlexander,[@] Ashutosh Kumar,[@] Masaaki Saitow,[△] Xiao Wang,[@] Benjamin P. Pritchard,[†] Prakash Verma,[▽] Henry F. Schaefer, III,[○] Konrad Patkowski,[◆] Rollin A. King,[&] Edward F. Valeev,[@] Francesco A. Evangelista,[▽] Justin M. Turney,[○] T. Daniel Crawford,[@] and C. David Sherrill*[,†]

[†]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

[‡]National Institutes of Health, National Heart, Lung and Blood Institute, Laboratory of Computational Biology, 5635 Fishers Lane, T-900 Suite, Rockville, Maryland 20852, United States

[¶]Department of Chemistry and Biochemistry, Florida State University, Tallahassee, Florida 32306-4390, United States

[§]Department of Chemistry and Biochemistry, The City College of New York, New York, New York 10031, United States

[∥]Department of Chemistry, Hacettepe University, Ankara 06800, Turkey

[⊥]Division of Chemistry and Chemical Engineering, California Institute of Technology, Pasadena, California 91125, United States

[#]Department of Chemistry, Centre for Theoretical and Computational Chemistry, UiT, The Arctic University of Norway, N-9037 Tromsø, Norway

[@]Department of Chemistry, Virginia Tech, Blacksburg, Virginia 24061, United States

[△]Department of Chemistry and Research Center for Smart Molecules, Rikkyo University, 3-34-1 Nishi-ikebukuro, Toshima-ku, Tokyo 171-8501, Japan

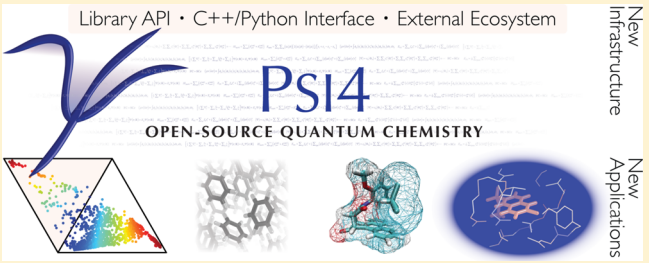
[▽]Department of Chemistry, Emory University, Atlanta, Georgia 30322, United States

[○]Center for Computational Quantum Chemistry, University of Georgia, Athens, Georgia 30602, United States

[◆]Department of Chemistry and Biochemistry, Auburn University, Auburn, Alabama 36849, United States

[&]Department of Chemistry, Bethel University, St. Paul, Minnesota 55112, United States

ABSTRACT: Psi4 is an ab initio electronic structure program providing methods such as Hartree−Fock, density functional theory, configuration interaction, and coupled-cluster theory. The 1.1 release represents a major update meant to automate complex tasks, such as geometry optimization using complete-basis-set extrapolation or focal-point methods. Conversion of the top-level code to a Python module means that Psi4 can now be used in complex workflows alongside other Python tools. Several new features have been added with the aid of libraries providing easy access to techniques such as density fitting, Cholesky decomposition, and Laplace denominators. The build system has been completely rewritten to simplify interoperability with independent, reusable software components for quantum chemistry. Finally, a wide range of new theoretical methods and analyses have been added to the code base, including functional-group and open-shell symmetry adapted perturbation theory, density-fitted coupled cluster with frozen natural orbitals, orbital-optimized perturbation and coupled-cluster methods (e.g., OO-MP2 and OO-LCCD), density-fitted multiconfigurational self-consistent field, density cumulant functional theory, algebraic-diagrammatic construction excited states, improvements to the geometry optimizer, and the "X2C" approach to relativistic corrections, among many other improvements.



1. INTRODUCTION

Quantum chemical computations have become an indispensable part of molecular science. They can provide accurate molecular geometries, reaction mechanisms and energetics, and simulated spectra. Quantum chemistry also yields parameters needed to develop force-field models and as such serves to anchor multi-scale modeling efforts. Advances in electronic structure theory

Received: February 17, 2017
Published: May 10, 2017

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

and in computer hardware have meant that quantum chemistry is now capable of giving very accurate results for systems with approximately three dozen atoms or less and reasonably accurate results for hundreds of atoms.

Unfortunately, electronic structure methods are inherently complicated, and the computer programs to implement them are quite lengthy. Fully featured quantum chemistry program packages can easily reach more than one million lines of source code. Additionally, it takes significant effort to optimize these codes so that they run as quickly as possible (an important consideration, given that computations are very time-consuming for larger molecules and/or more accurate models). Thus, most of the popular quantum chemistry programs have been developed over many years through the combined efforts of numerous programmers and often several research groups. This model of large, complex codes developed by multiple programmers over many years presents a tremendous challenge for keeping up with the rapid pace of recent innovations in computer hardware, including the increasing number of central processing unit (CPU) cores per node, the ubiquity of graphics processing units (GPUs), and emerging hardware like Intel's Xeon Phi. It also presents a challenge for the rapid deployment of new techniques from electronic structure theory because typically a desirable new feature is reimplemented for each quantum chemistry package.

We believe that this current development model is unsustainable and that the future of quantum chemistry software development lies in a more modular approach in which small, independent teams develop reusable software components that can be incorporated directly into multiple quantum chemistry packages, for example, as a library or standalone program that has no dependency on any particular quantum chemistry program.[1] For example, libraries like Libint,[2] libefp,[3,4] CheMPS2,[5−7] simint,[8,9] Libxc,[10,11] dkh,[12,13] and PCMSolver[14,15] are written so that they can be called from multiple quantum chemistry programs, and programs like dftd3[16] and MRCC[17,18] have been developed so that they are straightforward to interface to other packages. Several analysis and visualization tools (e.g., WebMO,[19] Molden,[20,21] and VMD[22,23]) have been written to analyze wave functions with data written in common formats. Over the past few years, we have substantially restructured and extended key portions of the Psi4 quantum chemistry package to facilitate interoperability with such reusable software components (or even with other large quantum chemistry programs).

Our build system, driver, and binary distribution system have been redesigned to make it easier for Psi4 to call independent software components. At the same time, we have also made Psi4 callable as a Python library so that it can be incorporated into complex workflows driven by custom Python scripts or interfaced with other scientific Python projects, of which there has been an explosion in the past few years. Wrapping much of the C++ functionality of Psi4 by Python has not only enabled interoperability, but it has also led to more rapid prototyping and development of complex functionality. Interoperability has also been enhanced by the addition of functions to write out data in common file formats used by quantum chemistry analysis tools, e.g., Molden format for molecular orbitals and formatted checkpoint files.

By moving our development system to a publicly viewable GitHub[24] repository and by incorporating social coding practices, we have made it easier for independent developers to interface their work to Psi4. Completely independently of the Psi4 developers, interfaces to Psi4 have been developed for a number of projects, including GeomeTRIC,[25,26] a geometry optimization code that includes the translation-rotation-internal coordinate (TRIC) system; JANPA, a Java-based implementation of the Natural Population Analysis (NPA)[27] method;[28,29] an implementation[30] of the restrained electrostatic potential (RESP) method for fitting atomic charges;[31] and htmd, a force-field parametrization tool.[32]

We have used object-oriented design to make certain key functionalities available through a generic application program interface (API) while allowing optimizations or adaptations to various hardware to be done "under the hood" without necessitating changes to the top-level code that implements a particular quantum chemistry method. For example, this allows us to transparently switch among different libraries to compute electron repulsion integrals depending on which is optimal for a particular type of hardware. Finally, we have also added libraries to enable techniques that are becoming more important in modern quantum chemistry, such as density fitting[33−41] and Cholesky decomposition.[42−46]

Here, we summarize our efforts to modularize and extend Psi4 along these lines. We also report on additional new capabilities of the package. This article, and its author list, reflects primarily the changes since the previous Psi4 publication,[47] written at the stage of an alpha release. The Psi4 1.1 release carries over much of the design, code, and capabilities of that earlier version of Psi4 and also retains some code and capabilities from Psi3.[48]

2. PYTHON INTERFACE

In the alpha release of Psi4 in 2012,[47] we introduced a Python front-end to what had previously been an exclusively C/C++ program. All computationally intensive modules remain written in C++ for efficiency reasons, but user input parsing and simple, high-level functions that help drive the program are written in Python.

2.1. Using Python to Parse User-Friendly Input and Automate Multi-Step Computations. One of the design goals of Psi4 is to make the program as easy to use as possible for both beginners and expert users. For beginners, a minimal input suffices for basic computations. Indeed, one does not even need to specify atomic coordinates; for common molecules, these can be obtained from the PubChem database (for nodes with Internet connectivity).[49] An input file to optimize the geometry of the benzene molecule using Hartree−Fock self-consistent-field (SCF) with the STO-3G basis[50] is as simple as

```
molecule benzene{
  pubchem:benzene
}
set basis sto-3g
optimize("scf")
```

For a graphical user interface (GUI), the popular WebMO program[19] is capable of drawing molecules, generating text input files for Psi4, managing a job queue, and visualizing orbitals and other results.

Although many solutions would have sufficed to parse input files like the example above, we also wished to allow much more complex input files by advanced users who might want to perform multistep computations, execute loops to build a potential energy surface, create tables and export data structures of results, or interface with other Python projects, all in the input file. In this, we were inspired by the Molpro program,[51] which enables looping and postprocessing operations.

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

Our approach is to use the Python language itself for input yet allow free-form specification of common quantum chemical input (e.g., molecules, options, custom basis sets) by intercepting and parsing them into proper Python commands. In this way, basic users submit minimal input files without any required knowledge of Python, whereas the inclined can draw upon the full scientific Python ecosystem. This combination of simple quantum chemistry directives and user-specified Python code forms a domain-specific language, "Psithon", that has shown itself an effective balance between power and ease-of-use since its introduction in alpha Psi4. Advanced users of quantum chemistry programs commonly write workflow-automating helper scripts to aggregate computations, drive supporting executables, or extract key quantities. With Psi4, such functionality can be expressed directly in an input file with conventional Python or, if a given functionality is popular, absorbed into the Psi4 driver.

2.1.1. Specifying Levels of Theory, Basis Set Extrapolations, Corrections for Basis Set Superposition Errors, and Many-Body Expansions of Clusters. In the 1.1 release, Psi4 has been significantly overhauled to make many common operations, which would normally be done by custom user scripts, as easy as specifying a single input string. For example, the common notation of computational chemistry that separates method and basis with a slash can be used, e.g., `optimize("SCF/STO-3G")`. This idea can be extended to complete-basis-set (CBS) extrapolations, which can be expressed as `energy("SCF/cc-pV[D,T,Q]Z")` to perform an exponential extrapolation of the SCF energy using the cc-pVDZ, cc-pVTZ, and cc-pVQZ basis sets.[52] A two-point extrapolation of the correlation energy can be specified in a similar way, e.g., `energy("MP2/aug-cc-pV[T,Q]Z")` for second-order Møller−Plesset perturbation theory (MP2) in augmented basis sets.[52,53] Depending on the method (SCF or post-SCF) and number of ζ-levels specified, extrapolation formulas are selected automatically among 2- and 3-point Helgaker schemes for SCF energies[54] and 2-point Helgaker for correlation energies.[55] Alternate or user-defined (by writing a Python function) extrapolation equations may be specified through additional keywords. For example, `opt("MP2/aug-cc-pV[T,Q]Z + D:CCSD(T)/cc-pV[D,T]Z", delta_scheme=myxtpl_2)` performs a focal point optimization with the coupled-cluster delta correction employing a custom formula for its 2-point extrapolation atop MP2 using the default 2-point Helgaker formula. Importantly, the 1.1 release also generalizes these extrapolations to work or geometry optimizations and vibrational frequencies, e.g., `frequencies("MP2/cc-pV[D,T]Z")` for an MP2 extrapolated vibrational frequency computation.

In a similar way, corrections for basis set superposition errors (BSSE) can also be specified very simply, `energy("MP2/aug-cc-pV[T,Q]Z", bsse_type="cp")`, for an MP2-extrapolated Boys−Bernardi counterpoise correction.[56] For a molecular cluster, the Valiron−Mayer functional counterpoise (VMFC) correction[57] is also available. Molecular clusters can also be analyzed in an N-body fashion in terms of dimers, trimers, etc. This is fully automated by Psi4 's Python driver and can be called by altering the `bsse_type` argument to the usual `energy()` function. Grep-able, tabulated, and programmatically accessible quantities for each requested fragment and basis set partitioning combination are made available, and multiple BSSE treatments can be computed concurrently.

Handling such complex operations with such minimal user input is the role of the Python driver portion of Psi4. It contains generic functions like `energy()`, `optimize()`, and `frequencies()`, which take a quantum chemistry method as an argument, then set appropriate options based on user specifications and best practices for the method. The driver then decides what C++ modules need to be called and in what order. Results are passed back from the C++ modules to the Python driver so that they can undergo various types of postprocessing (e.g., converting to other units, combining with results from other programs, exporting as data structures, etc.). For composite and otherwise complex methods, the main driver functions are often recursively called.

2.1.2. Passing Wave Function Information. As the input to Psi4 computations is a single Python script, information may be passed through objects in main memory. The 1.1 release greatly improves information passing over the alpha release by allowing wave function objects to be passed between modules. For example, one could use density functional theory (DFT) orbitals as guess orbitals for a complete-active-space self-consistent-field (CASSCF) procedure

```
scf_e, scf_wfn = energy("bp86",
   return_wfn=True)
energy("casscf", ref_wfn=scf_wfn)
```

Additionally, one can obtain any method's wave function and pass it to generic objects that can write out or manipulate data: `molden()`, which writes Molden files; `cubeprop()`, which writes densities and electrostatic potentials on a grid; and `oeprop()`, which computes arbitrary one-electron properties. For example, writing the optimized-orbital MP2 orbitals in Molden format can be accomplished by

```
omp2_e, omp2_wfn = energy("omp2",
   return_wfn=True)
molden(omp2_wfn, "omp2_orbitals.molden")
```

2.2. Rapid Prototyping. Much of the C++ infrastructure of Psi4 has been made accessible at the Python layer. This provides opportunities for rapid prototyping of quantum chemical procedures in Python while leveraging the existing, efficient C++ infrastructure. Indeed, for some procedures it is even possible to write the final, production-level version of the code by organizing C++ calls completely in Python. Interoperability between C++ and Python was previously accomplished using Boost Python,[58] but in the 1.1 release, we have removed Boost due to its large size and build difficulties; we now use the header-only pybind11 library.[59]

Psi4 modules such as the "JK" object that generates Coulomb and exchange matrices are readily available through Python

```
# Create a JK object in the primary basis set
jk = psi4.core.JK.build(primary_basis)
# Provide an occupied orbital matrix for the
# density matrix:

# D = C_left * C_right(T)
# by default C_right = C_left
jk.add_C_left(Cocc)
# Note that there are many possible sources
# for Cocc, e.g. sphericalized atomic
# orbitals, a converged SCF, etc., and these
# details are up to the user
# Perform the computation and get the J and K
# matrices
jk.compute()
J = jk.J()
K = jk.K()
```


DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

As JK builds are the bottleneck in SCF computations, a Python-based SCF algorithm can have nearly identical execution time to that of full C++ code. This is one of many cases where the most computationally intensive parts of calculations can be handled by a Psi4 library, resulting in top-level code that is easy to write, easy to understand, and computationally efficient.

The Psi4NumPy project[60] aims to enable the creation of clear, readable code for development or pedagogical purposes by combining the extensive Python accessibility of Psi4 with the popular Numerical Python (NumPy)[61] library for high-level array manipulation and access to the BLAS library. The latest version of Psi4 adopts code from that project[60] allowing Psi4 data and NumPy objects to be seamlessly interconverted. For example,

```
np_array = numpy.zeros((5, 5))

psi4_matrix = psi4.Matrix.from_array(np_array)

new_np_array = numpy.array(psi4_matrix)
```

2.3. Opportunities for Education in Electronic Structure Theory. Writing quantum chemistry code in this manner allows for the decoupling of learning quantum chemistry from having to simultaneously learn a low-level programming language. As low-level languages typically have much steeper learning curves than high-level languages like Python, this can be quite advantageous. We have leveraged the Python accessibility of much of the Psi4 infrastructure to develop guided projects for students to write their own Hartree−Fock code; several students in Georgia Tech's Computational Chemistry course have done this as a class project. One of us (D.G.A.S.) offered an introduction to electronic structure theory and programming in the summer of 2016 for summer students and new graduate students, utilizing Psi4 and Psi4NumPy.

2.4. Psi4 as a Python Module. Finally, the most recent improvement in Psi4 is its ability to be loaded into a Python script as a regular Python module. This ability has also opened up numerous opportunities to integrate Psi4 with other powerful Python modules and related tools. For example, we have been working on integrating Psi4 with the popular OpenMM package for molecular mechanics. As both of these programs have Python front-ends, passing data between the programs is straightforward. Psi4 can also be run interactively through Jupyter notebooks,[62] which are like Mathematica notebooks for Python; intermediate results can be graphed and analyzed with standard Python tools. We believe the Jupyter notebooks hold promise as "lab notebooks" (or Supporting Information) for computational research as well as for guided computational chemistry laboratory exercises.

3. INTERFACES TO EXTERNAL PROJECTS

The current model of quantum chemistry software development is one in which each quantum chemistry package is a large "silo" of code standing independently of all the others. Although it is convenient for the user to have a package that contains many different features, it is wasteful for different teams of developers to duplicate the effort of adding new features. It would be better for specialists in a particular method to develop a reusable component that could be adopted by multiple programs. Psi4 strives to minimize the barrier to interoperability with these add-ons by keeping their build and maintenance back-ends separate from Psi4, yet maintaining a monolithic front-end for users. Components not written by Psi4 core developers have been removed from the code repository; the source instead resides in the author's primary (upstream) repository where possible. In their place are single "External Project" files that, when activated by `ENABLE_<AddOn>` options, queue up the task of building that library from *external* source code using an *internal* configuration or common set of build parameters and shared dependencies. This "superbuild" framework is driven by the popular CMake tool.[63] The External Project scheme ensures that the footprint of each interfaced project within the Psi4 codebase consists of one build file and typically half-a-dozen lines of linking and include directives. For greatest flexibility, all quantum chemistry dependencies and add-ons that require linking or internal code activation can be built from source (downloaded from GitHub or research Web sites) during the Psi4 build. Alternatively, existing prebuilt libraries can be detected and seamlessly linked in to the final Psi4 library and plugin system. Wherever rational, shared and archive (static) libraries are equally supported. Whenever possible, the changes to an external project's build system required by Psi4 have been accepted upstream to the main project. These typically consist of a few short files that should also prove useful to non-Psi4 users of the add-ons.

Forming a clear delineation between Psi4 code and external projects simplifies build maintenance but also permits better management and reuse of libraries so that core developers and users alike can access a full-featured Psi4 compilation without hassle. Invaluable in this endeavor is Anaconda,[64] a distribution of Python (and its associated package manager, conda) that focuses on the scientific community and the cross-platform accessibility of Python and Python/C++/Fortran binaries. Conda tools facilitate the construction of relocatable (installable into any directory) and generic (executable on almost any OS distribution) binary packages, which are hosted at https://anaconda.org/psi4 for Mac and Linux (and, through The Windows Subsystem for Linux, recent versions of Microsoft Windows). To make binaries as foolproof as possible while also maintaining the efficiency so vital to quantum chemistry computations, a number of techniques are employed, including static linking of high-quality math libraries, static linking of system libraries wherever possible,[65] resolving symbols to a widely available version of glibc, and turning off processor-specific optimizations (users wishing to obtain the full performance of processor-specific optimizations, which vary from negligible to considerable, can activate these by building the project from scratch). Conda packages are built for Psi4 itself and each of its add-ons. Each can be downloaded individually, or an environment specification can install a complete set. For the Psi4 developer, pointing CMake to that install location allows all the add-on and dependency libraries to be detected and linked into Psi4 itself, drastically reducing the compile time and resulting in a full-featured Psi4 build. For someone who wishes to use the Psi4 libraries to develop quantum chemistry code as quickly as possible, C++ and Python Psi4 plugins can be developed from conda packages and conda compilers without a local development environment. For the user, the Psi4 conda package, some half-a-dozen add-on packages, and still more non-quantum-chemistry dependencies are bundled up along with the conda package manager into a standalone installer, available from http://psicode.org/downloads.html, that unpacks into a Miniconda + Psi4 + AddOns release distribution. Psi4 is also available as a package in many Linux distributions, such as Debian, Fedora, and Ubuntu.

For external projects, the Psi4 project will (1) leave control of their code under their purview, (2) maintain any interfacing code

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

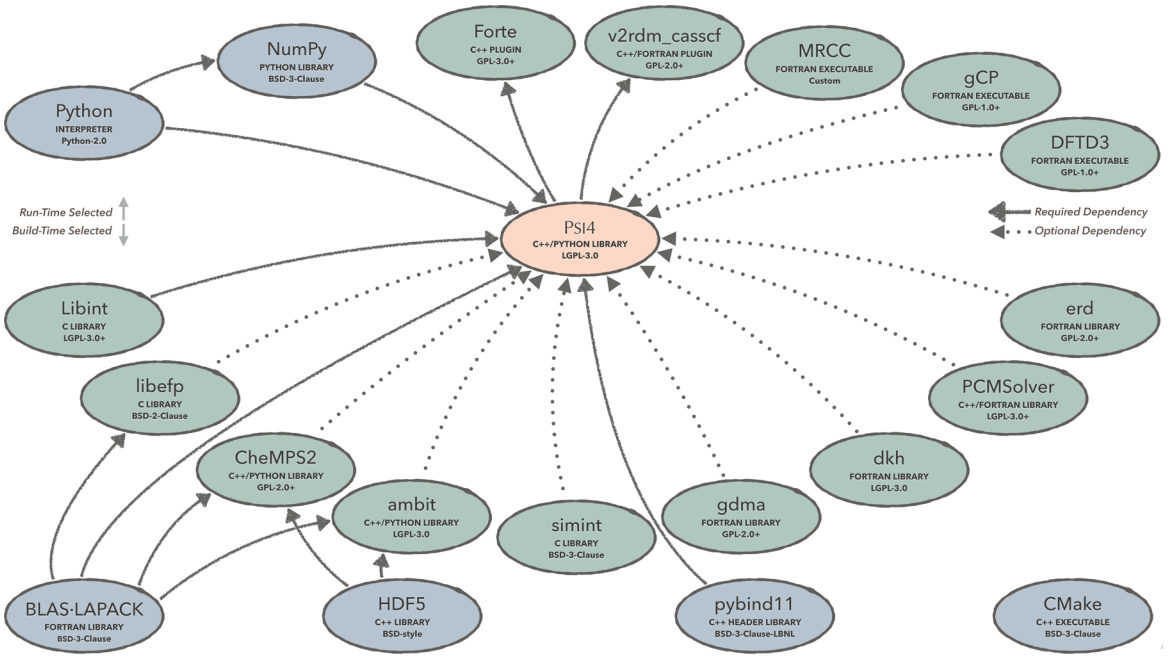


Figure 1. Psi4 creates an ecosystem for code, not a silo. Software packages required to build, required to run, or optionally provide additional capabilities to Psi4 are shown. All are available as conda packages from the `default` or `psi4` channels (except pybind11, which is built internally, MRCC, AMBIT, and Forte).

needed, (3) regularly run integration tests between Psi4 and their code, (4) build a mostly statically linked conda package so that any of their users can obtain a prebuilt binary distribution through `conda install addon --channel psi4`, (5) provide a development sandbox for their code through Psi4 plugins, and (6) provide conda download counts independent of Psi4. This maintain-in-pieces build-as-whole model is working for add-ons that are executables, libraries, and plugins from languages C, C++, and Fortran (and could be extended to proprietary or restricted license add-ons), as seen in Figure 1.

Cognizant that the expanding interconnectivity of quantum chemical software demonstrated by Figure 1 does not end with Psi4 (especially now that Psi4 is a library more than an executable) and that Psi4 has benefited from the flexible licensing terms of our add-ons, the license for Psi4 itself has been relaxed from the GNU General Public License version 2 or later (GPL-2.0+) to the GNU Lesser General Public License version 3 (LGPL-3.0), thereby permitting Psi4 to be incorporated downstream without downstream projects inheriting the GPL. The required dependencies have been minimized so that it is possible to build a (non-full-featured) Psi4 without any GPL involvement, and a conda package meeting those restrictions is distributed.

The following is a list of external projects integrated with Psi4:

- DFTD3: This independent executable program by S. Grimme provides damped dispersion corrections for use with density functional theory.[16] The latest version incorporates modified damping parameters [-D3M and -D3M(BJ)] to better model short-range contacts as described by Smith et al.[66]
- gCP: This "geometric counterpoise" executable by S. Grimme provides approximate corrections for basis set superposition error based on the molecular geometry, allowing the implementation of the semiempirical methods HF-3c and PBEh-3c.[67,68]
- PCMSolver: This library by Di Remigio, Frediani, and co-workers[14,15] implements the widely used polarizable continuum model (PCM) for solvation.[69−72] Within PCM, the mutual solute−solvent polarization is represented by an apparent surface charge σ spread over the cavity boundary and computed as the unique solution to the integral equation associated with the classical Poisson problem,[73] $\hat{\mathcal{T}}\sigma = -\hat{\mathcal{R}}\varphi$, where the definitions of the operators $\hat{\mathcal{T}}$ and $\hat{\mathcal{R}}$ depend on the molecular geometry and Green's function for the differential problem at hand. This mathematical framework can be applied to complex environments, such as homogeneous isotropic, ionic, and anisotropic liquids[73] and systems with interfaces.[74−77] In the model, a molecular cavity built from a set of interlocking atom-centered spheres is partitioned into a mesh of (curvilinear or planar) finite elements to each of which is attached a finite basis set of piecewise-regular polynomials that offer a natural discretization of the above equation into matrix form.
- LIBEFP: This library by Kaliman and Slipchenko[3,4] implements the effective fragment potential (EFP) method of Gordon and co-workers[78,79] to provide ab initio force fields.
- MRCC: The high-order coupled-cluster methods available in the multireference coupled-cluster program of Kállay and co-workers[17,18] are accessible from Psi4.
- DKH: This library by Wolf, Reiher, and Hess implements the Douglas−Kroll−Hess relativistic corrections between second and fourth order.[12,13]
- Integrals: Psi4 is unusual in that it supports multiple electron repulsion integral (ERI) libraries. We continue to support the LIBINT library (version 1)[2] originally written for Psi3. However, an interface layer called libmints allows us to replace LIBINT with other integrals libraries, and we have written an interface to the ERD library of Flocke and Lotrich.[80] Very recently, we have also interfaced to the new SIMINT library of Pritchard[8,9] optimized for architectures like Intel Xeon Phi.
- CheMPS2: This library[5−7] presents a spin-adapted implementation of the density matrix renormalization group (DMRG) approach,[81,82] which is an effective way to approximate complete active space configuration interaction (CAS CI) with much larger active spaces (up to approximately 50 electrons in 50 orbitals for

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

general problems). DMRG-based complete active space self-consistent field (DMRG-CASSCF)[83] and CASSCF plus second-order perturbation theory (DMRG-CASPT2)[84] are available through CheMPS2.

- GDMA: Psi4 has been interfaced with the GDMA program of Stone,[85] which provides distributed multipole analysis of wave functions to represent the electrostatic field as a sum of contributions from point charges, dipoles, etc., at atomic centers or other sites.
- V2RDM_CASSCF: This plugin[86] (requiring Psi4) implements a semidefinite programming algorithm for the variational optimization of the ground-state two-electron reduced-density matrix (2RDM). This variational 2RDM (v2RDM) approach is most useful as an alternative to CAS CI for describing the electronic structure of an active space. Unlike CAS CI, v2RDM methods can be applied to large active spaces comprised of as many as 50 electrons in 50 orbitals. The plugin also provides a v2RDM-driven CASSCF procedure that exploits density fitting technology for the treatment of large numbers of external orbitals.[87]
- AMBIT: This library by Turney and co-workers[88] provides tensors to represent quantum chemistry quantities such as electron repulsion integrals or coupled-cluster amplitudes.
- CFOUR: We have nearly completed a prototype interface that allows Psi4 to drive computations performed by the CFOUR package for coupled-cluster theory.[89]
- FORTE: This plugin (requiring Psi4) by Li, Hannon, Schriber, Zhang, and Evangelista[90] implements a number of multireference quantum chemistry methods, including approaches based on the driven similarity renormalization group (DSRG)[91] with perturbative[92,93] and nonperturbative truncation schemes.[94] Forte also implements two new selected CI approaches: adaptive[95] and projector CI.[96]

4. DEVELOPMENT PROCESS

Psi4 has adopted tools standard in the software industry such as GitHub for collaboration and version control, Travis CI[97] for continuous integration testing, and CodeCov[98] for test coverage metrics. This ensures the primary source code retains its reproducibility, and all code changes can be reviewed by the entire Psi4 community. New developers feel safe making changes to the base code, as all changes will be publicly visible and reviewed by the core team through the GitHub "pull request" mechanism, and the changes will also be tested automatically by the Travis CI system. The public visibility of the GitHub project helps communication between developers, both within the core team and also with independent developers who want their code to work with Psi4. Several improvements and bug fixes have been contributed back to the Psi4 development team from the broader community.

5. LIBRARIES

5.1. Density Fitting. One of the major barriers to efficient electronic structure computations is the large number of ERIs. For a set of real one-particle spatial basis functions $\{\phi_p(\vec{r}_1)\}$ (e.g., Gaussian atomic orbitals, molecular orbitals), the ERIs are defined as

$$(pq|rs) \equiv \iint_{\mathbb{R}^6} \mathrm{d}^3 r_1 \, \mathrm{d}^3 r_2 \, \phi_p(\vec{r}_1)\phi_q(\vec{r}_1)\frac{1}{r_{12}}\phi_r(\vec{r}_2)\phi_s(\vec{r}_2) \tag{1}$$

where $\vec{r}_1$ and $\vec{r}_2$ denote the Cartesian coordinates of electrons 1 and 2, respectively, and r_{12} is the distance between electrons 1 and 2. Each ERI encapsulates the electrostatic interaction between a particle in the "pair-space" basis function $\rho_{pq}(\vec{r}_1) \equiv \phi_p(\vec{r}_1)\phi_q(\vec{r}_1)$ in coordinate 1 with another particle in pair-space basis function $\rho_{rs}(\vec{r}_2) \equiv \phi_r(\vec{r}_2)\phi_s(\vec{r}_2)$ in coordinate 2. Despite the existence of rich literature on the efficient evaluation of the ERIs in Gaussian orbitals,[99−101] the order-4 ERI tensor is a major bottleneck in terms of generation, storage, and utilization.

A particularly elegant approach to reducing the impact of the 4-index nature of the ERIs in electronic structure theory begins from the observation that the quadratic-scaling pair-space basis $\{\rho_{pq}(\vec{r}_1)\}$ is highly numerically redundant, particularly in large molecular systems and/or high-quality basis sets. Fortunately, the usual quadratic-scaling pair-space basis can be accurately represented in a linear-scaling "auxiliary basis" $\{\chi_A(\vec{r}_1)\}$, e.g., $\rho_{pq}(\vec{r}_1) \approx \sum_A d^A_{pq}\chi_A(\vec{r}_1)$, where d^A_{pq} is an order-3 fitting coefficient tensor. This is the crux of the extremely successful Cholesky decomposition (CD)[42−46] and density fitting (DF)[33−41] approaches. In particular, we focus on the now-ubiquitous "Coulomb-metric" DF approach in which the ERI is approximated as

$$(pq|rs) \approx \sum_{AB} (pq|A)(A|B)^{-1}(B|rs) \equiv B^C_{pq}B^C_{rs} \tag{2}$$

In the last step, and below, we assume that repeated indices are summed over. The use of DF reduces the storage burden of the ERI tensor from order-4 to order-3. In addition, transforming the integrals from the atomic orbital to molecular orbital basis now becomes an $O(n^4)$ operation instead of an $O(n^5)$ operation. Moreover, the subsequent utilization of the ERI tensor in contractions with density matrices or wave function amplitudes is often considerably more efficient than with conventional integrals due to reduced storage cost and the occasional opportunity to achieve formal scaling reduction by the "unpinning" of the pq and rs indices in the DF representation (i.e., this sometimes allows more efficient ways to order the steps in the tensor multiplications). Coulomb-metric DF is exceedingly accurate with errors in relative properties of chemical interest (barrier heights, interaction energies, etc.) generally lower than 0.05 kcal mol^{-1} when using standardized auxiliary fitting basis sets.[41]

Coulomb-metric DF is utilized in many electronic structure methods in Psi4, as discussed below. The use of DF to compress the molecular-orbital basis ERIs is so ubiquitous that we have introduced a `DFERI` helper class to quickly construct the DF factorization for a given set of orbital pair-spaces (including customized pair-spaces). An example of using the `DFERI` object to build the DF factorization of the hole-particle integrals $(ia|jb) \approx B^Q_{ia}B^Q_{jb}$ (where i and j indicate hole orbitals, and a and b indicate particle orbitals) is presented as

```
// Build a DFERI object with default
// options/orbital spaces
shared_ptr<DFERI> df = DFERI::build(pri-
mary,aux,options);

// Request the decomposition (ia|jb) ~=
// B_ia^Q B_jb^Q for MP2
df->add_pair_space("MP2", "ACTIVE_OCC",
   "ACTIVE_VIR");
// Compute the DF factorization
df->compute();
// Grab the DF tensor (on disk) for
// subsequent use
shared_ptr<Tensor>BiaQ=df->ints["MP2"];
```


DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

This helper class shields the user from the specifics of forming the DF integrals: parallelization over multiple cores, using Cauchy−Schwarz spatial sieving to remove insignificant pq pairs in the AO basis DF integrals, applying the atomic orbital to molecular orbital integral transform, locating the pseudoinverse of the fitting metric, and applying the fitting metric. Additionally, if multiple pair-space tasks are requested, the generation of the integrals, first-half integral transformation, and fitting metric inverse are coalesced to avoid redundant effort. Thus, in just a few lines of client code, the user has access to DF factorizations that are tractable on problems with up to ∼6000 basis functions and can then focus on the application of these DF tensors to implement their desired method.

5.2. JK Object. A fundamental (and often quite computationally expensive) operation in a myriad of electronic structure methods is the construction of one-electron "Coulomb" (J_{pq}) or "exchange" (K_{pq}) matrices corresponding to a prespecified generalized one-particle density matrix (D_{rs})

$$J_{pq} \equiv \sum_{rs} (pq|rs)D_{rs} \tag{3}$$

$$K_{pq} \equiv \sum_{rs} (pr|qs)D_{rs} \tag{4}$$

SCF, coupled-perturbed Hartree−Fock, configuration interaction singles, and many terms in symmetry-adapted perturbation theory and multiconfigurational self-consistent-field (MCSCF) can be formulated in terms of these matrices. There are many approaches to efficiently and/or exactly forming these matrices, including integral-direct, PK supermatrix,[102] or density fitting. To hide the details of these approaches from the user, we have introduced an abstract `JK` class, implemented by any one of a number of `DirectJK`, `DFJK`, `DiskJK`, and so forth classes. After construction, the user interacts with the `JK` object in an abstract manner, requesting J and K matrices without needing to know the details of how they are constructed. This means that any method coded in terms of the `JK` object automatically can use DF, Direct, or any other integral technology from the outset. An example of computing J and K matrices at the Python layer was presented above in section 2.2.

The density matrices supplied to the `JK` object are actually specified in terms of their underlying factorization in terms of occupied orbital coefficients. This factorization naturally occurs in most places that J and K matrices appear (and can always be achieved by supplying the density and identity matrices as the factor tensors, if needed). However, endowing the `JK` object with knowledge of the factorization and symmetry of the density matrix often allows for significant performance gains, e.g., in DF Hartree−Fock.[41]

6. NEW CAPABILITIES

6.1. Symmetry-Adapted Perturbation Theory (SAPT). One of the specialties of Psi4 is a suite of symmetry-adapted perturbation theory (SAPT)[103−105] codes for the analysis of noncovalent interactions.[106] Previously, we had introduced efficient density-fitted and/or natural-orbital-accelerated implementations of SAPT0,[107,108] SAPT2+3,[109,110] and SAPT2+3 with a coupled-cluster doubles treatment of dispersion, SAPT2+3(CCD).[111] These allow the use of DF-SAPT0 on systems with up to ∼3500 basis functions and the use of DF-SAPT2+3(CCD) on systems with up to ∼1200 basis functions. Recently, we have extended the capabilities of the SAPT codes in Psi4 to broaden their applicability beyond closed-shell dimers and to deepen the insight obtained. These developments include the "functional group" SAPT (F-SAPT) decomposition,[112,113] which provides a partition of the SAPT0 energy at the chemical functional group level through partial summations of the SAPT perturbation series in terms of pairs of nuclei and localized electrons. F-SAPT0 has the same computational cost as a standard DF-SAPT0 analysis but provides deeper insight by quantifying the contribution of each pair of functional groups to each intermolecular interaction energy term. For example, with F-SAPT we can partition the interaction energy in a protein−ligand interaction into contributions between functional groups on the ligand and individual side-chains in the active site. An example of F-SAPT is depicted in Figure 2.

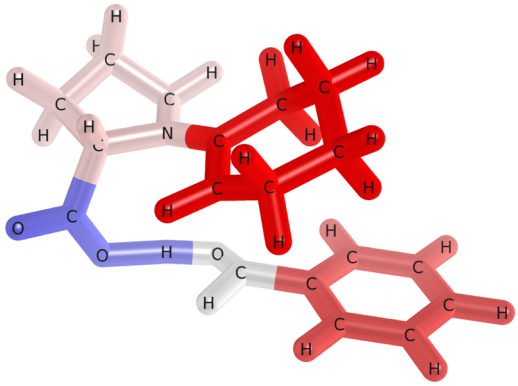


Figure 2. Example F-SAPT analysis of noncovalent stabilization of a transition state in the reaction of benzaldehyde with an enamine intermediate (see ref 114). Color coding shows the electrostatic interaction between the phenyl group of benzaldehyde with the three functional groups of the enamine intermediate. A red group is attracted to the other monomer, whereas a blue group is repelled by the other monomer; C−H/π contacts contribute to the favorable red/red interaction between the cyclohexenyl and phenyl groups.

Another new development is "intramolecular" SAPT (I-SAPT),[115] which uses Hartree−Fock-in-Hartree−Fock embedding[116] to produce a SAPT0-type interaction energy analysis for two moieties interacting within the presence of a third body, which might be covalently bound to one or both of the moieties in question. I-SAPT thus yields insight into intramolecular noncovalent interactions previously inaccessible to SAPT. The last new extension of SAPT methods in Psi4 is an efficient implementation of open-shell SAPT0.[117] The corresponding equations were only recently published,[118,119] and we present the first publicly available fully density-fitted implementation SAPT0 using unrestricted Hartree−Fock references, allowing computations on high-spin radical dimers of unprecedented sizes. The SAPT0 codes, and their F-SAPT, I-SAPT, and open-shell extensions, make substantial use of the JK object and other libraries described in section 5, which greatly accelerated the implementation of these methods. Moreover, the use of these common library primitives serve to "standardize" the tractability limits of all of the SAPT0-type methods, meaning that they all are deployable to the same ∼3500 basis function scale.

6.2. Density-Fitted and Cholesky Decomposition Coupled-Cluster Theory with Frozen Natural Orbitals. One of the major additions to Psi4 since the alpha release is coupled-cluster theory utilizing density-fitting or Cholesky decomposition approximations for the ERIs, as provided by the libraries described in section 5.1. The DF/CD approximations substantially reduce the input/output time associated with processing the ERIs. However, the rate-determining

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

step in coupled-cluster singles and doubles (CCSD)[120] remains the same, scaling as $O(o^2v^4)$, where o and v are the number of occupied and virtual orbitals, respectively. Indeed, this rate-determining step, the so-called particle−particle ladder term, becomes actually more computationally costly in the DF/CD formalism, as one has to create the all-virtual ERIs from the 3-index quantities in a step that scales as $O(N_{aux}v^4)$, where N_{aux} is the size of the auxiliary index (the number of auxiliary basis functions in the DF approach, or the number of Cholesky vectors in the CD approach). Although technically only a fifth-power term, it is a very expensive one, because N_{aux} and v are both large compared to o.

We addressed this issue by using a synergistic approximation: frozen natural orbitals (FNOs).[121−124] These orbitals are better suited to truncating the virtual orbital subspace; one can delete a significant fraction of the most weakly occupied natural orbitals with only a minor impact on the total energy. Indeed, in our tests of intermolecular interaction energies in the S22 test set,[125] we performed computations with coupled-cluster through perturbative triples [CCSD(T)] in which we deleted FNO virtual orbitals with occupation numbers less than 10^{-5}. In an aug-cc-pVDZ basis set,[52,53] the mean absolute error for this approximation was only 0.012 kcal/mol when MP2 was used to approximately correct for the deleted FNO virtual orbitals.[126] The errors due to density fitting are an order of magnitude smaller than this, approximately 0.001−0.002 kcal/mol when using the standard double-ζ density fitting auxiliary basis sets.[127] On the other hand, the combined DF/CD and FNO approximations can lead to substantial speedups in some cases, particularly for intermolecular interactions when counterpoise corrections are applied because most of the "ghost" functions are weakly occupied and can be effectively truncated by the FNO procedure. To compute the relatively small and subtle three-body contribution to the interaction energy of the benzene trimer, we used a very conservative 10^{-7} cutoff on the FNOs; nevertheless, the overall counterpoise-corrected DF-FNO-CCSD(T) computation ran four times faster than the corresponding conventional CCSD(T) computation.[127] This allowed us to perform a large number of such computations, which resulted in the first definitive resolution of the three-body contribution to the lattice energy of crystalline benzene.[128] Additional speedups of approximately 3× are possible using our version of the code optimized for GPUs.[129] The MOLCAS[130] and Q-Chem[131] programs have also implemented coupled-cluster codes using DF and/or CD approximations combined with FNOs or similar orbitals.[132,133]

6.3. Perturbation Theory and Coupled-Cluster Gradients. Analytic gradients have been implemented for MP2 and third-order Møller−Plesset perturbation theory (MP3) using restricted Hartree−Fock (RHF) or unrestricted Hartree−Fock (UHF) orbitals with conventional integrals or density fitting. From these codes, it was straightforward[134] to also implement "MP2.5", which is a simple average of MP2 and MP3.[135] This method has been found[134−136] to give rather accurate results for noncovalent interaction energies at a substantially reduced cost compared to that of CCSD(T).

We have also recently added analytic gradients for CCSD and CCSD(T) with RHF or UHF orbitals. Additionally, we published the analytic gradients of density-fitted CCSD.[137] The density fitting approximation, although speeding up the computation of certain terms, has very little effect on geometries or vibrational frequencies.[137,138]

6.4. Optimized-Orbital Methods. In wave function methods, one typically uses canonical Hartree−Fock orbitals. However, it is also possible to use orbitals that minimize the energy of the targeted post-Hartree−Fock wave function; MCSCF is a popular example. Minimizing the energy of other kinds of wave functions with respect to orbital rotations is not nearly as well explored, but it has been reported in the past for MP2[139−142] and coupled-cluster doubles (CCD).[120,143−145] The use of optimized orbitals simplifies computation of response properties and means that orbital response terms are already accounted for in analytic gradients. Additionally, optimized-orbital (OO) CCD was found,[144] like Brueckner CCD, to provide much better vibrational frequencies for systems exhibiting artifactual spatial symmetry-breaking like NO_3 and O_4^+.[146,147]

Psi4 provides extensive support for optimized-orbital wave function methods, including an improved Newton−Raphson algorithm for the orbital optimization.[148] Optimized-orbital versions of MP2,[148] MP3,[149] MP2.5,[150] linearized CCD,[151] and CCD[148] are implemented as well as standard or asymmetric (T) triples corrections[152,153] to the latter.[154] In addition, analytic energy gradients are available for OO-MP2,[150] OO-MP3,[155] OO-linearized CCD,[151] and OO-MP2.5.[134] Density-fitted versions of the energies and gradients are also available for MP2,[156] MP3,[157] MP2.5,[157] and linearized CCD.[158] For a set of weakly interacting dimers, orbital optimization was found to maintain the good performance of Hartree−Fock-based MP2.5 for closed-shell systems, but for open-shell systems, orbital optimization decreased errors by a factor of 5 (leading to better results than those of the more expensive CCSD method).[134]

6.5. Density-Fitted Multi-Configurational Self-Consistent-Field. In Psi4 1.1, the MCSCF module has been completely rewritten using an approach similar to the atomic-orbital formulation.[159] This method is able to reduce the most computationally expensive pieces of MCSCF to J and K builds. There are several pieces of the pure atomic-orbital formulation that do not scale well to large active spaces, and these pieces can be supplied either by density-fitting techniques[160] or transforming exact ERIs.

For flexibility, all methods of the MCSCF wave function object have been exported to Python so that advanced MCSCF methods can be easily built.

```
cas_e, cas_wfn = energy("CASSCF",
   return_wfn=True)
# Compute the sigma vector for the current
# MCSCF state
inp_vec = cas_wfn.reference_civector()
out_vec = cas_wfn.new_civector()
cas_wfn.sigma(inp_vec, out_vec)
# Build an arbitrary two-particle
# density matrix between two vectors
tpdm = cas_wfn.tpdm(inp_vec, out_vec)
```

Using these flexible objects allows for the rapid exploration of various post-MCSCF theories. For example, SAPT based on CASSCF wave functions is being developed in such a manner.

6.6. Density Cumulant Functional Theory. Psi4 is the only quantum chemistry program that features a publicly available implementation of density cumulant functional theory (DCT). Rather than describing electron correlation using a many-electron wave function, as in conventional ab initio theories, DCT uses the cumulant of the two-electron reduced density matrix.[161] By parametrizing the cumulant via a set of approximate constraints (also known as N-representability

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

conditions), DCT provides a way to directly obtain electronic density matrices and energies without explicitly constructing the electronic wave function, thus making the computation of molecular properties (e.g., equilibrium structures, dipole moments, vibrational spectra) very efficient. Among other attractive features of DCT are size-extensivity, orbital relaxation, and the ability to efficiently incorporate high-order electron correlation effects. The earlier version of Psi4 presented the first implementation[162] of the original DCT formulation that used a simple set of approximate *N*-representability conditions (the DC-06 method).[161] In the Psi4 1.1 release, several new DCT methods have been implemented, including an improved description of the one-particle density matrix (DC-12),[163] orbital-optimized DCT formulations (ODC-06 and ODC-12),[164] and more sophisticated *N*-representability conditions and three-particle correlation effects [ODC-13 and ODC-13(λ_3)].[165] These new methods have a similar computational cost to that of the original DC-06 method but exhibit much higher accuracies, especially for systems with unpaired electrons and significant multireference effects.[166,167] The new DCT code also features analytic gradients for the DC-06, ODC-06, and ODC-12 methods.[164,168] Very recently, we introduced density fitting and spin-adaptation, which significantly reduce the cost of the two-electron integral transformation for the orbital-optimized DCT methods (e.g., ODC-12).[169] Using this new DF-based implementation, it is now possible to perform DCT computations with all electrons correlated for systems with more than 1000 basis functions.

6.7. Algebraic-Diagrammatic Construction Excitation Energies. The latest version of Psi4 includes code to compute electronic excitation energies using the second-order algebraic-diagrammatic construction scheme [ADC(2)].[170] This can be thought of as an excited state generalization of MP2. The code accounts for point-group symmetry through the LIBDPD library, and it also includes a "partially renormalized" variant, PR-ADC(2), meant to confer resistance against quasi-degeneracy, as described in ref 171. It can perform routine computations on systems with ∼1000 basis functions.

6.8. Geometry Optimizer. By default, the geometry optimizer uses redundant internal coordinates, but the ability to alternatively use Cartesian or delocalized internal coordinates has been added since the alpha release. The default optimization step is now the iterative, restricted one[172] arising from the rational function optimization method.[173] Cartesian Hessians from any source and the Hessian guess from Lindh et al.[174] may now be used.

There is improved support for constrained optimizations, including the ability to freeze selected internal and Cartesian coordinates, and to optimize toward a structure with a desired coordinate value using extra forces. The user can also turn on a "dynamic-level" algorithm that incrementally tries more robust optimization methods for difficult systems when poor steps occur. This approach was critical in obtaining optimized geometries in a study of highly strained alkoxy-substituted 1,8-bis((propyloxyphenyl) ethynyl) naphthalenes.[175]

6.9. X2C Relativistic Corrections. One-component scalar theories offer an economical and convenient way to include approximate relativistic effects in electronic structure computations both at the self-consistent-field level (Hartree−Fock, DFT) and in electron correlation methods. Psi4 implements the spin-free one-electron version of the exact two-component approach,[176−185] abbreviated as X2C. An X2C computation starts with the diagonalization of the one-electron Dirac Hamiltonian using a kinetically balanced basis set. Next, a Foldy−Wouthuysen transformation[186] is performed, which yields an effective one-electron Hamiltonian matrix ($\mathbf{h}_{X2C}$).

The choice of basis for X2C computations requires some considerations. In general, contracted basis sets do not offer a balanced description of the large and small components of the wave function, so it is generally recommended to evaluate the X2C Hamiltonian via a decontraction/recontraction procedure. The default procedure used in Psi4 consists of solving the Dirac Hamiltonian in a decontracted atomic orbital basis followed by projection of the solutions onto the contracted basis. Alternatively, the user may perform a computation in a fully decontracted basis.

One of the major advantages of spin-free scalar relativistic approaches like X2C is that they can be easily interfaced with nonrelativistic electronic structure methods by replacing the one-electron Hamiltonian matrix with the quantity $\mathbf{h}_{X2C}$. All other quantities that enter these theories (two-electron integrals, exchange-correlation functionals, etc.) are evaluated using standard nonrelativistic equations. The cost of X2C computation has a scaling proportional to the cube of the number of basis functions (the number of primitive basis functions, in the case of the decontraction/recontraction procedure) and is generally negligible. An example application of the X2C capabilities of Psi4 is a recently developed X2C/orthogonality-constrained DFT[187] approach to predict near-edge X-ray absorption spectra.[188] The use of scalar relativistic Hamiltonians was found to be essential, because for second-row elements, the correction to nonrelativistic excitation energies is already as large as 8 eV.

7. COMMUNITY AND OUTREACH

Finally, the Psi4 team and our collaborators have fostered the growth of a user community. We have spent significant effort in writing and maintaining a clear user manual,[189] which presents not only the features and input file syntax but also discussions on simplifying and automating more complex computations. The manual is formatted using Sphinx,[190] which facilitates heavy use of cross-referencing for user convenience. To ensure that documentation on API and user options is kept up-to-date, we automatically generate this information directly from the source code. New users can consult a number of tutorial videos posted on YouTube[191] and post questions at our user forum.[192] News about Psi4, and downloads, are posted on our Web site.[193]

Not only do we wish to help experienced users learn about how to use Psi4, but we also want to leverage Psi4 to teach undergraduates and graduate students about computational chemistry. We have formed the Psi4Education group to develop and distribute a set of computational chemistry laboratory exercises using freely available software, including Psi4. Our webpage provides the laboratories developed to date,[194] and a book chapter describes the project as a whole.[195] We use the popular WebMO graphical front-end,[19] which is interfaced to Psi4. A Psi4Education workshop was held at the Biennial Conference on Chemical Education demonstrating these lab exercises in 2014. In 2016, a joint workshop on quantum chemistry and molecular mechanics simulations at the Florida American Chemical Society meeting introduced attendees to Psi4 integrated with plotting capabilities in Jupyter notebooks. We plan to continue developing additional lab exercises as part of this effort. In addition, as mentioned in section 2.3, advanced exercises in electronic structure theory for upper-level undergraduates and beginning graduate students, developed using our user-friendly Python front-end and the NumPy linear algebra library, are available through the Psi4NumPy project.[60]

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

8. CONCLUSIONS

The hybrid C++/Python programming model seems successful in our experience so far. C++ allows fast, efficient code for lower-level, computationally intensive tasks. For the driver and other higher-level portions of the code, Python seems to offer a lower barrier to entry for new programmers, and it can facilitate rapid prototyping thanks to the interface with NumPy. Moreover, the large number of active scientific computing projects in Python is likely to aid interoperability of Psi4 with other projects, which is one of our major goals. To that end, the 1.1 release uses a new build and distribution system to make it easy to connect Psi4 with independently developed, reusable software components. Finally, we have developed a new set of libraries providing efficient generation of generalized Coulomb and exchange matrices, density-fitted electron repulsion integrals, and related quantities. These libraries have greatly simplified and sped up our development of numerous new features over the past few years.

AUTHOR INFORMATION

Corresponding Author

*E-mail: sherrill@gatech.edu.

ORCID

Lori A. Burns: 0000-0003-2852-5864
Andrew C. Simmonett: 0000-0002-5921-9272
Edward G. Hohenstein: 0000-0002-2119-2959
Roberto Di Remigio: 0000-0002-5452-9239
Henry F. Schaefer III: 0000-0003-0252-2083
Rollin A. King: 0000-0002-1173-4187
Edward F. Valeev: 0000-0001-9923-6256
T. Daniel Crawford: 0000-0002-7961-7016
C. David Sherrill: 0000-0002-5570-7666

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We are grateful to the contributors of all earlier versions of the Psi program. Brandon Bakr provided Figure 2, and Michael Zott provided a figure element for the TOC. Several of the coauthors have been supported in their development of Psi4 and affiliated projects by the U.S. National Science Foundation through grants ACI-1147843, CHE-1300497, CHE-1351978, CHE-1361178, ACI-1449723, ACI-1450169, ACI-1465149, CHE-1566192, and ACI-1609842; by the U.S. Department of Energy through grants DE-SC0015512 and DE-SC0016004; and by the U.S. Department of Defense through an HPCMP Applications Software Initiative (HASI) grant. U.B. acknowledges support from the Scientific and Technological Research Council of Turkey (Grant No. TUBITAK-114Z786) and the European Cooperation in Science and Technology (Grant CM1405). R.D.R. acknowledges support from the Research Council of Norway through a Centre of Excellence Grant (Grant 179568/V30). J.F.G. acknowledges the postdoctoral fellowship P2ELP2_155351 from the Swiss NSF. The authors thank Prof. Stefan Grimme for consultations in validating HF-3c and PBEh-3c.

REFERENCES

(1) Wilson, E. K. *Chem. Eng. News* **2014**, *92*, 26.
(2) Valeev, E. F.; Fermann, J. T. *Libint*: A high-perforrmance library for computing Gaussian integrals in quantum mechanics; 2017. https://github.com/evaleev/libint (accessed May 8, 2017).
(3) Kaliman, I. A.; Slipchenko, L. V. *libefp*; http://libefp.github.io (accessed May 8, 2017).
(4) Kaliman, I. A.; Slipchenko, L. V. *J. Comput. Chem.* **2013**, *34*, 2284−2292.
(5) Wouters, S. *CheMPS2*: A Spin-Adapted Implementation of DMRG for ab initio Quantum Chemistry. http://github.com/SebWouters/CheMPS2 (accessed May 8, 2017).
(6) Wouters, S.; Poelmans, W.; Ayers, P. W.; van Neck, D. *Comput. Phys. Commun.* **2014**, *185*, 1501−1514.
(7) Wouters, S.; Van Neck, D. *Eur. Phys. J. D* **2014**, *68*, 272.
(8) Pritchard, B. P.; Chow, E. *Simint*. https://github.com/simint-chem.
(9) Pritchard, B. P.; Chow, E. *J. Comput. Chem.* **2016**, *37*, 2537−2546.
(10) Marques, M. A. L.; Oliveira, M.; Burnus, T.; Madsen, G.; Andrade, X.; Strubbve, D.; Lehtola, S. *Libxc*, a library of exchange-correlation functionals for density-functional theory, version 3.0.0; 2016. http://octopus-code.org/wiki/Libxc (accessed May 8, 2017).
(11) Marques, M. A. L.; Oliveira, M. J. T.; Burnus, T. *Comput. Phys. Commun.* **2012**, *183*, 2272−2281.
(12) Wolf, A.; Reiher, M.; Hess, B. A. *J. Chem. Phys.* **2002**, *117*, 9215−9226.
(13) Reiher, M.; Wolf, A. *J. Chem. Phys.* **2004**, *121*, 10945−10956.
(14) Di Remigio, R.; Frediani, L.; Mozgawa, K. *PCMSolver*, an Application Programming Interface for the Polarizable Continuum Model electrostatic problem, v1.1.7. http://pcmsolver.readthedocs.io/ (accessed May 8, 2017).
(15) Di Remigio, R.; Crawford, T. D.; Frediani, L. *PCMSolver*, An Application Programming Interface for the Polarizable Continuum Model. Proceedings of the SC15 Workshop on Producing High Performance and Sustainable Software for Molecular Simulation; 2016.
(16) *DFTD3*, a FORTRAN program implementing the DFT-D3 method, version 3.2 Rev. 0; Grimme Research Group: Mulliken Center for Theoretical Chemistry, Universität Bonn, 2016. http://www.thch.uni-bonn.de/tc/index.php?section/downloads&subsection/DFT-D3 (accessed May 8, 2017).
(17) Kállay, M.; Rolik, Z.; Csontos, J.; Ladjánski, I.; Szegedy, L.; Ladóczki, B.; Samu, G.; Petrov, K.; Farkas, M.; Nagy, P.; Mester, D.; Hégely, B. *MRCC*, A Quantum Chemical Program Suite. See: http://www.mrcc.hu.
(18) Rolik, Z.; Szegedy, L.; Ladjánszki, I.; Ladóczki, B.; Kállay, M. *J. Chem. Phys.* **2013**, *139*, 094105.
(19) Schmidt, J. R.; Polik, W. F. *WebMO 17*; WebMO, LLC: Holland, MI, 2016. http://www.webmo.net (accessed May 8, 2017).
(20) Schaftenaar, G.; *MOLDEN*, version 5.7. http://www.cmbi.ru.nl/molden/ (accessed May 8, 2017).
(21) Schaftenaar, G.; Noordik, J. H. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 123−134.
(22) VMD: Visual Molecular Dynamics; Theoretical and Computational Biophysics Group; University of Illinois at Urbana-Champaign, 2017. http://www.ks.uiuc.edu/Research/vmd/ (accessed May 8, 2017).
(23) Humphrey, W.; Dalke, A.; Schulten, K. *J. Mol. Graphics* **1996**, *14*, 33−38.
(24) GitHub. https://github.com/ (accessed May 8, 2017).
(25) Wang, L.-P.; Song, C. *GeomeTRIC*, a geometry optimization code that includes the TRIC coordinate system; 2017. http://github.com/leeping/geomeTRIC (accessed May 8, 2017).
(26) Wang, L.-P.; Song, C. *J. Chem. Phys.* **2016**, *144*, 214108.
(27) Reed, A. E.; Weinstock, R. B.; Weinhold, F. *J. Chem. Phys.* **1985**, *83*, 735−746.
(28) *JANPA*, A Freeware Program Package for Performing Natual Population Analysis, version 1.04; 2016. http://janpa.sourceforge.net/ (accessed May 8, 2017).
(29) Nikolaienko, I. Y.; Bulavin, L. A.; Hovorun, D. M. *Comput. Theor. Chem.* **2014**, *1050*, 15−22.
(30) McGibbon, R. T. *RESP2*, Restrained Electrostatic Potential Fitting for Psi4; 2016. http://github.com/rmcgibbo/resp2 (accessed May 8, 2017).
(31) Bayly, C. I.; Cieplak, P.; Cornell, W. D.; Kollman, P. A. *J. Phys. Chem.* **1993**, *97*, 10269−10280.
(32) Doerr, S.; Harvey, M. J.; Noé, F.; De Fabritiis, G. *J. Chem. Theory Comput.* **2016**, *12*, 1845−1852.

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

(33) Whitten, J. L. *J. Chem. Phys.* **1973**, *58*, 4496−4501.
(34) Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. *Int. J. Quantum Chem.* **1977**, *12*, 81.
(35) Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. *J. Chem. Phys.* **1979**, *71*, 3396−3402.
(36) Komornicki, A.; Fitzgerald, G. *J. Chem. Phys.* **1993**, *98*, 1398−1421.
(37) Feyereisen, M.; Fitzgerald, G.; Komornicki, A. *Chem. Phys. Lett.* **1993**, *208*, 359−363.
(38) Vahtras, O.; Almlöf, J.; Feyereisen, M. W. *Chem. Phys. Lett.* **1993**, *213*, 514−518.
(39) Rendell, A. P.; Lee, T. J. *J. Chem. Phys.* **1994**, *101*, 400−408.
(40) Kendall, R. A.; Fruchtl, H. A. *Theor. Chem. Acc.* **1997**, *97*, 158−163.
(41) Weigend, F. *Phys. Chem. Chem. Phys.* **2002**, *4*, 4285−4291.
(42) Beebe, N. H. F.; Linderberg, J. *Int. J. Quantum Chem.* **1977**, *12*, 683−705.
(43) Roeggen, I.; Wisloff-Nilssen, E. *Chem. Phys. Lett.* **1986**, *132*, 154−160.
(44) Koch, H.; Sánchez de Meras, A.; Pedersen, T. B. *J. Chem. Phys.* **2003**, *118*, 9481−9484.
(45) Aquilante, F.; Pedersen, T. B.; Lindh, R. *J. Chem. Phys.* **2007**, *126*, 194106.
(46) Aquilante, F.; Gagliardi, L.; Pedersen, T. B.; Lindh, R. *J. Chem. Phys.* **2009**, *130*, 154107.
(47) Turney, J. M.; Simmonett, A. C.; Parrish, R. M.; Hohenstein, E. G.; Evangelista, F. A.; Fermann, J. T.; Mintz, B. J.; Burns, L. A.; Wilke, J. J.; Abrams, M. L.; Russ, N. J.; Leininger, M. L.; Janssen, C. L.; Seidl, E. T.; Allen, W. D.; Schaefer, H. F.; King, R. A.; Valeev, E. F.; Sherrill, C. D.; Crawford, T. D. *WIREs Comput. Mol. Sci.* **2012**, *2*, 556−565.
(48) Crawford, T. D.; Sherrill, C. D.; Valeev, E. F.; Fermann, J. T.; King, R. A.; Leininger, M. L.; Brown, S. T.; Janssen, C. L.; Seidl, E. T.; Kenny, J. P.; Allen, W. D. *J. Comput. Chem.* **2007**, *28*, 1610−1616.
(49) Bolton, E. E.; Wang, Y.; Thiessen, P. A.; Bryant, S. H. In *Annual Reports in Computational Chemistry*; Wheeler, R. A., Spellmeyer, D. C., Eds.; Elsevier: Amsterdam, 2005; Vol. *4*; pp 217−241.
(50) Hehre, W. J.; Stewart, R. F.; Pople, J. A. *J. Chem. Phys.* **1969**, *51*, 2657−2664.
(51) Werner, H.-J.; Knowles, P. J.; Manby, F. R.; Schütz, M.; Celani, P.; Knizia, G.; Korona, T.; Lindh, R.; Mitrushenkov, A.; Rauhut, G.; Adler, T. B.; Amos, R. D.; Bernhardsson, A.; Berning, A.; Cooper, D. L.; Deegan, M. J. O.; Dobbyn, A. J.; Eckert, F.; Goll, E.; Hampel, C.; Hesselmann, A.; Hetzer, G.; Hrenar, T.; Jansen, G.; Köppl, C.; Liu, Y.; Lloyd, A. W.; Mata, R. A.; May, A. J.; Tarroni, R.; Thorsteinsson, T.; Wang, M.; Wolf, A. *MOLPRO*, version 2010.1; 2010. http://www.molpro.net (accessed May 8, 2017).
(52) Dunning, T. H. *J. Chem. Phys.* **1989**, *90*, 1007−1023.
(53) Kendall, R. A.; Dunning, T. H.; Harrison, R. J. *J. Chem. Phys.* **1992**, *96*, 6796−6806.
(54) Halkier, A.; Helgaker, T.; Jorgensen, P.; Klopper, W.; Olsen, J. *Chem. Phys. Lett.* **1999**, *302*, 437−446.
(55) Halkier, A.; Helgaker, T.; Jørgensen, P.; Klopper, W.; Koch, H.; Olsen, J.; Wilson, A. K. *Chem. Phys. Lett.* **1998**, *286*, 243−252.
(56) Boys, S. F.; Bernardi, F. *Mol. Phys.* **1970**, *19*, 553−566.
(57) Valiron, P.; Mayer, I. *Chem. Phys. Lett.* **1997**, *275*, 46−55.
(58) BOOST C++ Libraries. http://www.boost.org (accessed May 8, 2017).
(59) Jakob, W.; Rhinelander, J.; Moldovan, D. *pybind11*, Seamless operability between C++11 and Python; 2017. https://github.com/pybind/pybind11 (accessed May 8, 2017).
(60) Smith, D. G. A.; Sirianni, D. A.; Burns, L. A.; Patkowski, K.; Sherrill, C. D. *psi4/psi4numpy*: beta release; 2017. https://doi.org/10.5281/zenodo.293020.
(61) van der Walt, S.; Colbert, S. C.; Varoquaux, G. *Comput. Sci. Eng.* **2011**, *13*, 22−30.
(62) Perez, F.; Granger, B. E. *Comput. Sci. Eng.* **2007**, *9*, 21−29.
(63) *CMake*, a cross-platform family of tools designed to buid, test and package software. http://cmake.org (accessed May 8, 2017).
(64) *Anaconda*; Continuum IO, 2017. https://www.continuum.io/ (accessed May 8, 2017).
(65) Any system libraries not linked statically fall within the purview of https://www.python.org/dev/peps/pep-0513/#external-shared-libraries.
(66) Smith, D. G. A.; Burns, L. A.; Patkowski, K.; Sherrill, C. D. *J. Phys. Chem. Lett.* **2016**, *7*, 2197−2203.
(67) Sure, R.; Grimme, S. *J. Comput. Chem.* **2013**, *34*, 1672−1685.
(68) Grimme, S.; Brandenburg, J. G.; Bannwarth, C.; Hansen, A. *J. Chem. Phys.* **2015**, *143*, 054107.
(69) Miertuš, S.; Scrocco, E.; Tomasi, J. *Chem. Phys.* **1981**, *55*, 117−129.
(70) Tomasi, J.; Persico, M. *Chem. Rev.* **1994**, *94*, 2027−2094.
(71) Tomasi, J.; Mennucci, B.; Cammi, R. *Chem. Rev.* **2005**, *105*, 2999−3093.
(72) Mennucci, B.; Cammi, R. *Continuum Solvation Models in Chemical Physics: From Theory to Applications*; Wiley, 2008.
(73) Cancès, E.; Mennucci, B. *J. Math. Chem.* **1998**, *23*, 309−326.
(74) Corni, S.; Tomasi, J. *J. Chem. Phys.* **2002**, *117*, 7266.
(75) Frediani, L.; Cammi, R.; Corni, S.; Tomasi, J. *J. Chem. Phys.* **2004**, *120*, 3893−3907.
(76) Delgado, A.; Corni, S.; Goldoni, G. *J. Chem. Phys.* **2013**, *139*, 024105.
(77) Di Remigio, R.; Mozgawa, K.; Cao, H.; Weijo, V.; Frediani, L. *J. Chem. Phys.* **2016**, *144*, 124103.
(78) Gordon, M. S.; Freitag, M. A.; Bandyopadhyay, P.; Jensen, J. H.; Kairys, V.; Stevens, W. J. *J. Phys. Chem. A* **2001**, *105*, 293−307.
(79) Ghosh, D.; Kosenkov, D.; Vanovschi, V.; Williams, C. F.; Herbert, J. M.; Gordon, M. S.; Schmidt, M. W.; Slipchenko, L. V.; Krylov, A. I. *J. Phys. Chem. A* **2010**, *114*, 12739−12754.
(80) Flocke, N.; Lotrich, V. *J. Comput. Chem.* **2008**, *29*, 2722−2736.
(81) White, S. R.; Martin, R. L. *J. Chem. Phys.* **1999**, *110*, 4127−4130.
(82) Chan, G. K. L.; Head-Gordon, M. *J. Chem. Phys.* **2002**, *116*, 4462−4476.
(83) Wouters, S.; Bogaerts, T.; Van der Voort, P.; Van Speybroeck, V.; Van Neck, D. *J. Chem. Phys.* **2014**, *140*, 241103.
(84) Wouters, S.; Van Speybroeck, V. V.; Van Neck, D. *J. Chem. Phys.* **2016**, *145*, 054120.
(85) Stone, A. J. *GDMA*, A Program to Perform Distributed Multipole Analysis, version 2.2.11; 2015. http://www-stone.ch.cam.ac.uk/programs.html (accessed May 8, 2017).
(86) DePrince, A. E. *v2rdm-CASSCF*, A variational 2-RDM-driven CASSCF plugin to Psi4; 2016. http://github.com/edeprince3/v2rdm_casscf (accessed May 8, 2017).
(87) Fosso-Tande, J.; Nguyen, T.-S.; Gidofalvi, G.; DePrince, A. E., III *J. Chem. Theory Comput.* **2016**, *12*, 2260−2271.
(88) Turney, J. M. *AMBIT*, A C++ Library for the Implementation of Tensor Product Calculations through a Clean, Concise User Interaface; 2017. http://github.com/jturney/ambit (accessed May 8, 2017).
(89) Stanton, J. F.; Gauss, J.; Harding, M. E.; Szalay, P. G. *cfour*, Coupled-Cluster Techniques for Computational Chemistry. With contributions from Auer, A. A.; Bartlett, R. J.; Benedikt, U.; Berger, C.; Bernholdt, D. E.; Bomble, Y. J.; Cheng, L.; Christiansen, O.; Heckert, M.; Heun, O.; Huber, C.; Jagau, T.-C.; Jonsson, D.; Jusélius, J.; Klein, K.; Lauderdale, W. J.; Lipparini, F.; Matthews, D. A.; Metzroth, T.; Mück, L. A.; O'Neill, D. P.; Price, D. R.; Prochnow, E.; Puzzarini, C.; Ruud, K.; Schiffmann, F.; Schwalbach, W.; Simmons, C.; Stopkowicz, S.; Tajti, A.; Vázquez, J.; Wang, F.; Watts, J. D.; and the integral packages MOLECULE (Almlöf, J. and Taylor, P. R.), PROPS (Taylor, P. R.), ABACUS (Helgaker, T.; Aa. Jensen, H. J.; Jørgensen, P.; and Olsen, J.), and ECP routines by Mitin, A. V. and van Wüllen, C. http://www.cfour.de (accessed May 8, 2017).
(90) *Forte*, a suite of quantum chemistry methods for strongly correlated electrons; Evangelista Research Group; Emory University, Atlanta, GA, 2016. https://github.com/evangelistalab/forte (accessed May 8, 2017).
(91) Evangelista, F. A. *J. Chem. Phys.* **2014**, *141*, 054109.
(92) Li, C.; Evangelista, F. A. *J. Chem. Theory Comput.* **2015**, *11*, 2097−2108.

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

(93) Hannon, K. P.; Li, C.; Evangelista, F. A. *J. Chem. Phys.* **2016**, *144*, 204111.
(94) Li, C.; Evangelista, F. A. *J. Chem. Phys.* **2016**, *144*, 164114.
(95) Schriber, J. B.; Evangelista, F. A. *J. Chem. Phys.* **2016**, *144*, 161106.
(96) Zhang, T.; Evangelista, F. A. *J. Chem. Theory Comput.* **2016**, *12*, 4326−4337.
(97) TravisCI. https://travis-ci.org/ (accessed May 8, 2017).
(98) CodeCov. https://codecov.io/ (accessed May 8, 2017).
(99) McMurchie, L. E.; Davidson, E. R. *J. Comput. Phys.* **1978**, *26*, 218−231.
(100) Rys, J.; Dupuis, M.; King, H. F. *J. Comput. Chem.* **1983**, *4*, 154−157.
(101) Obara, S.; Saika, A. *J. Chem. Phys.* **1986**, *84*, 3963−3974.
(102) Raffenetti, R. C. *Chem. Phys. Lett.* **1973**, *20*, 335−338.
(103) Jeziorski, B.; Moszynski, R.; Szalewicz, K. *Chem. Rev.* **1994**, *94*, 1887−1930.
(104) Szalewicz, K. *WIREs Comput. Mol. Sci.* **2012**, *2*, 254−272.
(105) Hohenstein, E. G.; Sherrill, C. D. *WIREs Comput. Mol. Sci.* **2012**, *2*, 304−326.
(106) Sherrill, C. D. *Acc. Chem. Res.* **2013**, *46*, 1020−1028.
(107) Hohenstein, E. G.; Sherrill, C. D. *J. Chem. Phys.* **2010**, *132*, 184111.
(108) Hohenstein, E. G.; Parrish, R. M.; Sherrill, C. D.; Turney, J. M.; Schaefer, H. F. *J. Chem. Phys.* **2011**, *135*, 174107.
(109) Hohenstein, E. G.; Sherrill, C. D. *J. Chem. Phys.* **2010**, *133*, 014101.
(110) Hohenstein, E. G.; Sherrill, C. D. *J. Chem. Phys.* **2010**, *133*, 104107.
(111) Parrish, R. M.; Hohenstein, E. G.; Sherrill, C. D. *J. Chem. Phys.* **2013**, *139*, 174102.
(112) Parrish, R. M.; Sherrill, C. D. *J. Chem. Phys.* **2014**, *141*, 044115.
(113) Parrish, R. M.; Parker, T. M.; Sherrill, C. D. *J. Chem. Theory Comput.* **2014**, *10*, 4417−4431.
(114) Bakr, B. W.; Sherrill, C. D. *Phys. Chem. Chem. Phys.* **2016**, *18*, 10297−10308.
(115) Parrish, R. M.; Gonthier, J. F.; Corminboeuf, C.; Sherrill, C. D. *J. Chem. Phys.* **2015**, *143*, 051103.
(116) Manby, F. R.; Stella, M.; Goodpaster, J. D.; Miller, T. F. *J. Chem. Theory Comput.* **2012**, *8*, 2564−2568.
(117) Gonthier, J. F.; Sherrill, C. D. *J. Chem. Phys.* **2016**, *145*, 134106.
(118) Hapka, M.; Żuchowski, P. S.; Szcześniak, M. M.; Chałasiński, G. *J. Chem. Phys.* **2012**, *137*, 164104.
(119) Żuchowski, P. S.; Podeszwa, R.; Moszyński, R.; Jeziorski, B.; Szalewicz, K. *J. Chem. Phys.* **2008**, *129*, 084101.
(120) Purvis, G. D.; Bartlett, R. J. *J. Chem. Phys.* **1982**, *76*, 1910−1918.
(121) Sosa, C.; Geertsen, J.; Trucks, G. W.; Bartlett, R. J.; Franz, J. A. *Chem. Phys. Lett.* **1989**, *159*, 148−154.
(122) Klopper, W.; Noga, J.; Koch, H.; Helgaker, T. *Theor. Chem. Acc.* **1997**, *97*, 164−176.
(123) Taube, A. G.; Bartlett, R. J. *Collect. Czech. Chem. Commun.* **2005**, *70*, 837−850.
(124) Landau, A.; Khistyaev, K.; Dolgikh, S.; Krylov, A. I. *J. Chem. Phys.* **2010**, *132*, 014109.
(125) Jurečka, P.; Šponer, J.; Černý, J.; Hobza, P. *Phys. Chem. Chem. Phys.* **2006**, *8*, 1985−1993.
(126) DePrince, A. E.; Sherrill, C. D. *J. Chem. Theory Comput.* **2013**, *9*, 293−299.
(127) DePrince, A. E.; Sherrill, C. D. *J. Chem. Theory Comput.* **2013**, *9*, 2687−2696.
(128) Kennedy, M. R.; McDonald, A. R.; DePrince, A. E.; Marshall, M. S.; Podeszwa, R.; Sherrill, C. D. *J. Chem. Phys.* **2014**, *140*, 121104.
(129) DePrince, A. E.; Kennedy, M. R.; Sumpter, B. G.; Sherrill, C. D. *Mol. Phys.* **2014**, *112*, 844−852.
(130) Aquilante, F.; De Vico, L.; Ferre, N.; Ghigo, G.; Malmqvist, P.; Neogrady, P.; Pedersen, T. B.; Pitonak, M.; Reiher, M.; Roos, B. O.; Serrano-Andres, L.; Urban, M.; Veryazov, V.; Lindh, R. *J. Comput. Chem.* **2010**, *31*, 224−247.
(131) Shao, Y.; Gan, Z.; Epifanovsky, E.; Gilbert, A. T. B.; Wormit, M.; Kussmann, J.; Lange, A. W.; Behn, A.; Deng, J.; Feng, X.; Ghosh, D.; Goldey, M.; Horn, P. R.; Jacobson, L. D.; Kaliman, I.; Khaliullin, R. Z.; Kus, T.; Landau, A.; Liu, J.; Proynov, E. I.; Rhee, Y. M.; Richard, R. M.; Rohrdanz, M. A.; Steele, R. P.; Sundstrom, E. J.; Woodcock, H. L.; Zimmerman, P. M.; Zuev, D.; Albrecht, B.; Alguire, E.; Austin, B.; Beran, G. J. O.; Bernard, Y. A.; Berquist, E.; Brandhorst, K.; Bravaya, K. B.; Brown, S. T.; Casanova, D.; Chang, C.; Chen, Y.; Chien, S. H.; Closser, K. D.; Crittenden, D. L.; Diedenhofen, M.; DiStasio, R. A.; Do, H.; Dutoi, A. D.; Edgar, R. G.; Fatehi, S.; Fusti-Molnar, L.; Ghysels, A.; Golubeva-Zadorozhnaya, A.; Gomes, J.; Hanson-Heine, M. W. D.; Harbach, P. H. P.; Hauser, A. W.; Hohenstein, E. G.; Holden, Z. C.; Jagau, T.; Ji, H.; Kaduk, B.; Khistyaev, K.; Kim, J.; Kim, J.; King, R. A.; Klunzinger, P.; Kosenkov, D.; Kowalczyk, T.; Krauter, C. M.; Lao, K. U.; Laurent, A. D.; Lawler, K. V.; Levchenko, S. V.; Lin, C. Y.; Liu, F.; Livshits, E.; Lochan, R. C.; Luenser, A.; Manohar, P.; Manzer, S. F.; Mao, S.; Mardirossian, N.; Marenich, A. V.; Maurer, S. A.; Mayhall, N. J.; Neuscamman, E.; Oana, C. M.; Olivares-Amaya, R.; O'Neill, D. P.; Parkhill, J. A.; Perrine, T. M.; Peverati, R.; Prociuk, A.; Rehn, D. R.; Rosta, E.; Russ, N. J.; Sharada, S. M.; Sharma, S.; Small, D. W.; Sodt, A.; Stein, T.; Stueck, D.; Su, Y.; Thom, A. J. W.; Tsuchimochi, T.; Vanovschi, V.; Vogt, L.; Vydrov, O.; Wang, T.; Watson, M. A.; Wenzel, J.; White, A.; Williams, C. F.; Yang, J.; Yeganeh, S.; Yost, S. R.; You, Z.; Zhang, I. Y.; Zhang, X.; Zhao, Y.; Brooks, B. R.; Chan, G. K. L.; Chipman, D. M.; Cramer, C. J.; Goddard, W. A.; Gordon, M. S.; Hehre, W. J.; Klamt, A.; Schaefer, H. F.; Schmidt, M. W.; Sherrill, C. D.; Truhlar, D. G.; Warshel, A.; Xu, X.; Aspuru-Guzik, A.; Baer, R.; Bell, A. T.; Besley, N. A.; Chai, J.; Dreuw, A.; Dunietz, B. D.; Furlani, T. R.; Gwaltney, S. R.; Hsu, C.; Jung, Y.; Kong, J.; Lambrecht, D. S.; Liang, W.; Ochsenfeld, C.; Rassolov, V. A.; Slipchenko, L. V.; Subotnik, J. E.; Van Voorhis, T.; Herbert, J. M.; Krylov, A. I.; Gill, P. M. W.; Head-Gordon, M. *Mol. Phys.* **2015**, *113*, 184−215.
(132) Pitonak, M.; Aquilante, F.; Hobza, P.; Neogrady, P.; Noga, J.; Urban, M. *Collect. Czech. Chem. Commun.* **2011**, *76*, 713−742.
(133) Epifanovsky, E.; Zuev, D.; Feng, X.; Khistyaev, K.; Shao, Y.; Krylov, A. I. *J. Chem. Phys.* **2013**, *139*, 134105.
(134) Bozkaya, U.; Sherrill, C. D. *J. Chem. Phys.* **2014**, *141*, 204105.
(135) Pitoňák, M.; Neogrády, P.; Černý, J.; Grimme, S.; Hobza, P. *ChemPhysChem* **2009**, *10*, 282−289.
(136) Burns, L. A.; Marshall, M. S.; Sherrill, C. D. *J. Chem. Phys.* **2014**, *141*, 234111.
(137) Bozkaya, U.; Sherrill, C. D. *J. Chem. Phys.* **2016**, *144*, 174103.
(138) Bozkaya, U. *J. Chem. Phys.* **2014**, *141*, 124108.
(139) Adamowicz, L.; Bartlett, R. J. *J. Chem. Phys.* **1987**, *86*, 6314−6324.
(140) Lochan, R. C.; Head-Gordon, M. *J. Chem. Phys.* **2007**, *126*, 164101.
(141) Lochan, R. C.; Shao, Y.; Head-Gordon, M. *J. Chem. Theory Comput.* **2007**, *3*, 988−1003.
(142) Neese, F.; Schwabe, T.; Kossmann, S.; Schirmer, B.; Grimme, S. *J. Chem. Theory Comput.* **2009**, *5*, 3060−3073.
(143) Scuseria, G. E.; Schaefer, H. F. *Chem. Phys. Lett.* **1987**, *142*, 354−358.
(144) Sherrill, C. D.; Krylov, A. I.; Byrd, E. F. C.; Head-Gordon, M. *J. Chem. Phys.* **1998**, *109*, 4171−4181.
(145) Jankowski, K.; Rubiniec, K.; Wasilewski, J. *Chem. Phys. Lett.* **2001**, *343*, 365−374.
(146) Stanton, J. F.; Gauss, J.; Bartlett, R. J. *J. Chem. Phys.* **1992**, *97*, 5554−5559.
(147) Barnes, L. A.; Lindh, R. *Chem. Phys. Lett.* **1994**, *223*, 207−214.
(148) Bozkaya, U.; Turney, J. M.; Yamaguchi, Y.; Schaefer, H. F.; Sherrill, C. D. *J. Chem. Phys.* **2011**, *135*, 104103.
(149) Bozkaya, U. *J. Chem. Phys.* **2011**, *135*, 224103.
(150) Bozkaya, U.; Sherrill, C. D. *J. Chem. Phys.* **2013**, *138*, 184103.
(151) Bozkaya, U.; Sherrill, C. D. *J. Chem. Phys.* **2013**, *139*, 054104.
(152) Kucharski, S. A.; Bartlett, R. J. *J. Chem. Phys.* **1998**, *108*, 5243−5254.
(153) Crawford, T.; Stanton, J. *Int. J. Quantum Chem.* **1998**, *70*, 601−611.
(154) Bozkaya, U.; Schaefer, H. F. *J. Chem. Phys.* **2012**, *136*, 204114.
(155) Bozkaya, U. *J. Chem. Phys.* **2013**, *139*, 104116.

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197

(156) Bozkaya, U. *J. Chem. Theory Comput.* **2014**, *10*, 2371−2378.
(157) Bozkaya, U. *J. Chem. Theory Comput.* **2016**, *12*, 1179−1188.
(158) Bozkaya, U. *Phys. Chem. Chem. Phys.* **2016**, *18*, 11362−11373.
(159) Hohenstein, E. G.; Luehr, N.; Ufimtsev, I. S.; Martínez, T. J. *J. Chem. Phys.* **2015**, *142*, 224103−10.
(160) Aquilante, F.; Pedersen, T. B.; Lindh, R.; Roos, B. O.; Sánchez de Merás, A.; Koch, H. *J. Chem. Phys.* **2008**, *129*, 024113.
(161) Kutzelnigg, W. *J. Chem. Phys.* **2006**, *125*, 171101.
(162) Simmonett, A. C.; Wilke, J. J.; Schaefer, H. F.; Kutzelnigg, W. *J. Chem. Phys.* **2010**, *133*, 174122.
(163) Sokolov, A. Y.; Simmonett, A. C.; Schaefer, H. F. *J. Chem. Phys.* **2013**, *138*, 024107.
(164) Sokolov, A. Y.; Schaefer, H. F. *J. Chem. Phys.* **2013**, *139*, 204110.
(165) Sokolov, A. Y.; Schaefer, H. F.; Kutzelnigg, W. *J. Chem. Phys.* **2014**, *141*, 074111.
(166) Copan, A. V.; Sokolov, A. Y.; Schaefer, H. F. *J. Chem. Theory Comput.* **2014**, *10*, 2389−2398.
(167) Mullinax, J. W.; Sokolov, A. Y.; Schaefer, H. F. *J. Chem. Theory Comput.* **2015**, *11*, 2487−2495.
(168) Sokolov, A. Y.; Wilke, J. J.; Simmonett, A. C.; Schaefer, H. F. *J. Chem. Phys.* **2012**, *137*, 054105.
(169) Wang, X.; Sokolov, A. Y.; Turney, J. M.; Schaefer, H. F. *J. Chem. Theory Comput.* **2016**, *12*, 4833−4842.
(170) Trofimov, A. B.; Krivdina, I. L.; Weller, J.; Schirmer, J. *Chem. Phys.* **2006**, *329*, 1−10.
(171) Saitow, M.; Mochizuki, Y. *Chem. Phys. Lett.* **2012**, *525*, 144−149.
(172) Besalu, E.; Bofill, J. M. *Theor. Chem. Acc.* **1998**, *100*, 265−274.
(173) Banerjee, A.; Adams, N.; Simons, J.; Shepard, R. *J. Phys. Chem.* **1985**, *89*, 52−57.
(174) Lindh, R.; Bernhardsson, A.; Karlstrøm, G.; Malmqvist, P.-Å. *Chem. Phys. Lett.* **1995**, *241*, 423−428.
(175) Carson, B. E.; Parker, T. M.; Hohenstein, E. G.; Brizius, G. L.; Komorner, W.; King, R. A.; Collard, D. M.; Sherrill, C. D. *Chem. - Eur. J.* **2015**, *21*, 19168−19175.
(176) Dyall, K. G. *J. Chem. Phys.* **1997**, *106*, 9618−9626.
(177) Dyall, K. G. *J. Chem. Phys.* **2001**, *115*, 9136−9143.
(178) Kutzelnigg, W. *Chem. Phys.* **1997**, *225*, 203−222.
(179) Kutzelnigg, W.; Liu, W. *J. Chem. Phys.* **2005**, *123*, 241102.
(180) Kutzelnigg, W.; Liu, W. *Mol. Phys.* **2006**, *104*, 2225−2240.
(181) Liu, W.; Kutzelnigg, W. *J. Chem. Phys.* **2007**, *126*, 114107.
(182) Liu, W.; Peng, D. *J. Chem. Phys.* **2009**, *131*, 031104.
(183) Iliaš, M.; Saue, T. *J. Chem. Phys.* **2007**, *126*, 064102.
(184) Zou, W.; Filatov, M.; Cremer, D. *J. Chem. Phys.* **2011**, *134*, 244117.
(185) Cheng, L.; Gauss, J. *J. Chem. Phys.* **2011**, *135*, 084114.
(186) Foldy, L. L.; Wouthuysen, S. A. *Phys. Rev.* **1950**, *78*, 29−36.
(187) Evangelista, F. A.; Shushkov, P.; Tully, J. C. *J. Phys. Chem. A* **2013**, *117*, 7378−7392.
(188) Verma, P.; Derricotte, W. D.; Evangelista, F. A. *J. Chem. Theory Comput.* **2016**, *12*, 144−156.
(189) Psi4 Manual. http://psicode.org/psi4manual/master/index.html (accessed May 8, 2017).
(190) Sphinx, Python Documentation Generator. http://www.sphinx-doc.org/en/1.5.2/index.html (accessed May 8, 2017).
(191) Psi4 YouTube videos. https://www.youtube.com/user/psitutorials (accessed May 8, 2017).
(192) Psi4 User Forum. http://forum.psicode.org (accessed May 8, 2017).
(193) Psi4 Website. http://www.psicode.org (accessed May 8, 2017).
(194) Psi4Education. http://www.psicode.org/labs.php (accessed May 8, 2017).
(195) Fortenberry, R. C.; McDonald, A. R.; Shepherd, T. D.; Kennedy, M.; Sherrill, C. D. In *The Promise of Chemical Education: Addressing our Students' Needs*; Daus, K., Rigsby, R., Eds.; American Chemical Society: Washington, D.C., 2015; Vol. *1193*; pp 85−98.

DOI: 10.1021/acs.jctc.7b00174
J. Chem. Theory Comput. 2017, 13, 3185−3197