The Tale of HORTON: Lessons Learned in a Decade of Scientific Software Development

Matthew Chan,[1, a)] Toon Verstraelen,[2, b)] Alireza Tehrani,[3] Michelle Richer,[1] Xiaotian Derrick Yang,[1] Taewon David Kim,[1] Esteban Vöhringer-Martinez,[4, c)] Farnaz Heidar-Zadeh,[3, d)] and Paul W. Ayers[1, e)]

[1)] *Department of Chemistry and Chemical Biology, McMaster University, Hamilton, Ontario, L8S-4L8, Canada*

[2)] *Center for Molecular Modeling (CMM), Ghent University, Technologiepark-Zwijnaarde 46, B-9052, Zwijnaarde, Belgium*

[3)] *Department of Chemistry, Queen's University, Kingston, Ontario, K7L-3N6, Canada*

[4)] *Departamento de Físico Química, Facultad de Ciencias Químicas, Universidad de Concepción, 4070371 Concepción, Chile*

Abstract: HORTON is a free and open-source electronic-structure package written primarily in Python 3 with some underlying C++ components. While HORTON's development has been mainly directed by the research interests of its leading contributing groups, it is designed to be easily modified, extended, and used by other developers of quantum chemistry methods or post-processing techniques. Most importantly, HORTON adheres to modern principles of software development, including modularity, readability, flexibility, comprehensive documentation, automatic testing, version control, and quality-assurance protocols. This article explains how the principles and structure of HORTON have evolved since we started developing it more than a decade ago. We review the features and functionality of the latest HORTON release (version 2.3) and discuss how HORTON is evolving to support electronic structure theory research for the next decade.

Keywords: quantum chemistry software, computational chemistry, Hartree-Fock method, model hamiltonians, Density Functional Theory (DFT) methods, numerical integration grids, periodic boundary conditions, Gaussian integrals, atoms-in-molecules partitioning schemes, Hirshfeld partitioning, population analysis, electrostatic potential fitting, parsing and converting computational chemistry file formats, theoretical chemistry Python library

[a)] Co-first authors.

[b)] Co-first authors.; Toon Verstraelen: Toon.Verstraelen@ugent.be

[c)] Esteban Vöhringer-Martinez: evohringer@udec.cl

[d)] Farnaz Heidar-Zadeh: farnaz.heidarzadeh@queensu.ca

[e)] Paul W. Ayers: ayers@mcmaster.ca

I. INTRODUCTION

HORTON is a **H**elpful **O**pen-source **R**esearch **TO**ol for **N**-fermion systems, written primarily in Python to ensure the readability, flexibility, and ease-of-use of the code. HORTON provides a platform for 1) testing new ideas for the quantum many-body problem at a reasonable computational cost and 2) accessing state-of-the-art quantum chemistry methods which have not yet entered the mainstream. HORTON is primarily focused on molecular electronic structure theory, but it has limited support for periodic boundary conditions and post-processing calculations, and it is being extended to include model Hamiltonians and nucleons.[1–3]

While there are many quantum chemistry packages, there are relatively few packages that adopt the free and open-source software development model. Examples include Erkale[4], Psi4[5], OpenMolcas[6], NWChem[7], Dalton[8], GQCP[9], Fermi.jl[10], Serenity[11], and JuliaChem[12]. Among these, only HORTON, PyQuante, UquanteChem, Slowquant, PySCF[13], and PyBEST[14,15] (a spin-off of an earlier version of HORTON) are primarily written in Python. While some of these packages (especially PySCF) have superior functionality and performance to HORTON, HORTON's emphasis on thorough documentation, comprehensive testing, and modular and readable source code—required because it has never had any dedicated staff/researchers and because it emerged from intercontinental collaboration, rather than a single group's research code—is unique. The precept of HORTON, encapsulated in its name, was to be a helpful research tool: fast enough to be a useful computational tool, but designed to be helpful to developers who wish to implement entirely new quantum chemistry approaches or interface with other, external, packages. This vision has been the lodestar of HORTON's developers from the very beginning.

Although HORTON has existed in some form for nearly a decade, it has constantly evolved and has recently been reenvisioned. After reviewing the history of HORTON in section II, section III presents our guiding principles, section IV covers the main modules of HORTON 2 with examples, and section V provides a perspective on the future of HORTON 3. We refrain from including installation guidelines and any other technical details that can easily become outdated over time and instead, we invite our readers to consult HORTON's website for further information.

II. HISTORY OF HORTON

HORTON emerged in the summer of 2011 as a result of the close collaboration between Toon Verstraelen and Paul Ayers. Specifically, parameterizing molecular mechanics force fields directly from *ab initio* calculations required building a flexible and easy-to-modify constrained density-functional theory (DFT) program,[16–23] and the available constrained-DFT software was difficult to use/modify and had substantial convergence difficulties.[7,24] This motivated the creation of HORTON 1.x, which was almost single-handedly written by T. Verstraelen in Python 2 and C++. Python was chosen as the primary language of HORTON to ensure the readability and flexibility of the program, and because the primary purpose of HORTON has always been to rapidly prototype new ideas. Even though many good software engineering practices (Git version control, automatic testing, and documentation) were already present in HORTON 1.x, it violated modularity (e.g., mainly through the `Molecule` class, which was used throughout the code).

HORTON 2 was a response to the expansion of the Ayers group's interests to include wavefunction-based approaches to electron correlation,[25–31] Supporting wavefunction-based methods led to a need for increased performance, and in early versions of HORTON 2, linear algebra factories were used to ensure that most computational bottlenecks were relegated to underlying C++ code. Unfortunately, the `LinearAlgebraFactory` class also reached its tentacles into all facets of the code, and in our zeal to rapidly prototype new methods, the `LinearAlgebraFactory` class became a dumping ground for linear algebraic operations, growing quickly beyond its original purpose, which made it difficult to retroactively repair. There was also an alarming tendency to write C++ or Cython code instead of redesigning algorithms for efficiency with NumPy[32] vectorization. This combination of languages made the code difficult to understand, compile and maintain, and they were aggravated by the paucity of code quality tools for Cython.

Because of its increased complexity and scope, to avoid compromising our design principles, HORTON 2 adopted continuous integration, continuous deployment, and protocols for code-quality. We improved our development practices by expanding HORTON's documentation and incorporating code review of pull requests. Some quality-assurance tasks were automated, including testing for style, documentation, test-coverage, and correctness. To make it accessible, we released HORTON 2 on various software repositories (e.g., `conda`). To-

day, the HORTON library is automatically built on Azure by the Conda-Forge project and users can install it with one command on the terminal.

The deprecation of Python 2.7 on January 1st, 2020 necessitated large-scale modifications to the code base. Thanks to the plethora of unit tests implemented, we could complete Python 3 porting relatively quickly with reasonable confidence that no bugs were introduced. In releasing HORTON 2.3, we also increased modularity by removing the `LinearAlgebraFactory` and replacing it directly with the underlying NumPy operations. This change has drawbacks (if developers are not careful, unnecessary copies of arrays can be formed) but we decided that the advantages (increased modularity and decreased potential for developer misuse) were worth the trade-off. As a part of this change, we simplified the code by removing some post-Hartree-Fock methods that relied heavily on bespoke features of the `LinearAlgebraFactory`, like the antisymmetric product of 1-electron reference-orbital geminals (AP1roG).[29] This pruning was necessary to ensure HORTON 2.3 was aligned with our design principles and to accommodate the more general framework for wavefunction-based methods in HORTON 3.[33]

III. OUR DESIGN PRINCIPLES

HORTON is designed to be a helpful framework for rapidly prototyping methods and testing ideas, together with utilities that ensure the resulting implementation is not too inefficient. Being a research tool, the current focus of HORTON is *ab initio* electronic structure theory methods and post-processing (conceptual) quantum chemistry calculations. Unlike quantum chemistry packages which are primarily intended to facilitate computational studies, HORTON is not designed to achieve bleeding-edge performance: readability, extensibility, and modifiability are often prioritized over computational efficiency and code compactness when trade-offs are essential. HORTON has a gentle learning curve, with sensible default parameters that can be overridden by advanced users. HORTON is well-documented and well-commented, both for users and developers. It includes tools to ensure that test coverage and code quality remain high while facilitating development by contributors who are less technically savvy.

HORTON is written primarily in Python because its simple syntax welcomes novice programmers. C++ is occasionally used for performance-critical features, but we strive to confine C++ code to pre-packaged libraries. In addition, HORTON strives to use simple

Pythonic coding styles wherever possible, and resorts to invasive performance optimization (caching, C++, ...), which might impair code transparency, only after profiling reveals a computational bottleneck. HORTON is, and always will be, free and open source. The HORTON development team welcomes and supports new contributions in accordance with our Code of Conduct.

HORTON can be used in two ways: as part of a larger Python program or as a stand-alone library. Retaining this flexibility is a fundamental design principle of HORTON. For example, when HORTON's functionality is limited or its computational requirements are prohibitive, HORTON can be used alongside other software packages.

IV. MODULES AND EXAMPLES

As described below, HORTON is organized into modules. Lower-level modules provide basic functionality and are combined with high-level modules like the modules for performing Hartree-Fock/Kohn-Sham density-functional theory (DFT) calculations, evaluating the electrostatic potential, and computing atomic charges and other chemical concepts. These modules are briefly exemplified below, but we delegate comprehensive documentation to HORTON's website.

A. IO: Input and Output

The input and output (IO) module is used for parsing, storing, and writing a wide variety of computational chemistry file formats, and is designed to be easily extensible. This module mediates HORTON's interoperability with external software, including other electronic-structure and post-processing software packages. In the simplest case, one can use HORTON to convert between different file formats, which can be directly done on a command line:

```
$ horton-convert.py water_hfs_321g.fchk water.xyz
```

Listing 1. Convert file formats using command-line functionality.

Using HORTON as a Python library, the IO module is used to parse supported file formats to conveniently access information therein and write alternative file formats.

```
from horton import *
```

```python

# Load molecule from an FCHK file
mol = IOData.from_file(context.get_fn("test/water_b3lyp_ccpvtz.fchk"))

# Get information contained in FCHK file
print("Energy (a.u.): ", mol.energy)
print("HOMO Energy: ", mol.orb_alpha.homo_energy)
print("LUMO Energy: ", mol.orb_alpha.lumo_energy)
print("Dipole Moment: ", mol.dipole_moment)
print("NPA Charges: ", mol.npa_charges)
print("ESP Charges: ", mol.esp_charges)
print("Atomic Numbers: ", mol.numbers)
print("Atomic Coordinates (a.u.):")
print(mol.coordinates)

# Compute molecular mass and store it as an IOData instance attribute
mol.mass = sum([periodic[number].mass for number in mol.numbers])
print(f"Molecular Mass (amu): {mol.mass / amu: .5f}")

# Write molecule into MOLDEN file
mol.to_file("water_b3lyp_ccpvtz.molden")
```

Listing 2. Parse and convert file formats.

B. `GBasis`: Gaussian Basis Evaluation and Analytical Integration

This module is responsible for evaluations and manipulations involving Gaussian basis functions. At the simplest level, `Gbasis` computes the values of basis functions and various molecular descriptors (including the electron density and its derivatives, electrostatic potential, and kinetic energy density) on specified arrays of (grid) points. `Gbasis` also computes integrals over Gaussian basis functions, including overlap, moment, and one- and two-electron integrals (including various integrals associated with range-separated DFT-

functionals). Underlying this implementation is Ed Valeev's widely used Obara-Saika code, LibInt[34]. This introduces a dynamically linked dependency into `HORTON` and we have interfaced with it using Cython.

```
import numpy as np
from horton import *

# Set up a molecule and define basis set
coordinates = np.array([[1.16, 0.0, 0.0], [0.0, 0.0, 0.0], [-1.16, 0.0,
    0.0]]) * angstrom
numbers = np.array([8, 6, 8])
obasis = get_gobasis(coordinates, numbers, "def2-tzvpd")

# compute electronic kinetic energy integrals and then add electron
    nuclear attraction itegrals
one_mo = obasis.compute_kinetic()
obasis.compute_nuclear_attraction(coordinates, numbers.astype(float),
    output=one_mo)
print("one_mo shape = ", one_mo.shape)

# Compute two-electron repulsion integrals
two_mo = obasis.compute_electron_repulsion()
print("two_mo shape = ", two_mo.shape)

# Compute nuclear-nuclear repulsion energy
core_energy = compute_nucnuc(coordinates, numbers.astype(float))

# Make an instance of IOData and write to a FCIDUMP file
data = IOData(one_mo=one_mo, two_mo=two_mo, core_energy=core_energy, nelec
    =20, ms2=0)
```

```
data.to_file("hamiltonian_ao.FCIDUMP")
```

Listing 3. Calculate and write 1- and 2-electron integrals of carbon dioxide in Def2-TZVPD basis.

C. `Grid`: Numerical Integration and Visualiztion

For molecular numerical integration, `HORTON` primarily uses atom-centered Becke-Lebedev[35,36] integration grids. This is specifically used in evaluating exchange-correlation functions in DFT (section IV D) and atoms-in-molecule analysis (section IV E). In this approach, the molecular grid is composed of the union of weighted atom-centered grids. The atomic grids are built from a one-dimensional radial grid and a Lebedev-Laikov angular grid; users choose an atomic grid specification through the `agspec` argument by:

1. A string with the `rname:rmin:rmax:nrad:nll` format. Here `rname` specifies the type of radial grid (`linear`, `exp`, or `power`), `rmin` and `rmax` specify the first and last radial grid point (in angstrom), `nrad` is the number of radial grid points, and `nll` is the number of points for angular Lebedev-Laikov grid. Alternatively, the user can provide `(rgrid, nll)` tuple where `rgrid` is an instance of `RadialGrid`. In this flexible framework, the user can even specify a different atomic integration grid for each element or implement additional radial/angular grids.

2. For ease of use, `HORTON` also provides built-in optimized (a.k.a. pruned) Becke-Lebedev atomic integration grids; these can be easily invoked by setting the `agspec` argument equal to `coarse`, `medium`, `fine`, `veryfine`, `ultrafine`, or `insane` ; the expected accuracy from these built-in grids is documented on `HORTON`'s website.

```
import h5py as h5
import numpy as np
from horton import *

# Load the Gaussian FCHK file from HORTON's test data directory
fn_fchk = context.get_fn("test/water_b3lyp_ccpvtz.fchk")
mol = IOData.from_file(fn_fchk)
```

```
8
9  # Initialize the numerical grid
10 grid = BeckeMolGrid(mol.coordinates, mol.numbers, mol.pseudo_numbers,
       agspec="fine")
11 print("Number of Grid Points = ", grid.size)
12
13 # Get the spin-summed density matrix
14 dm_full = mol.get_dm_full()
15
16 # Compute the density (integrand) on the grid points and integrate it
17 rho = mol.obasis.compute_grid_density_dm(dm_full, grid.points)
18 print("Integrate Electron Density: ", grid.integrate(rho))
19
20 # Compute the expectation value of |r| by integrating product of all
       arguments (integrand)
21 r = np.sum(grid.points**2, axis=1)**0.5
22 expt_r = grid.integrate(rho, r)
23 print(f"Expectation Value of |r|: {expt_r}")
24
25 # Save grid into an HDF5 file for future use
26 with h5.File("water_b3lyp_ccpvtz_molgrid", "w") as fn:
27     grid.to_hdf5(fn)
```

Listing 4. Integrate electron density on a Becke-Lebedev grid and compute expectation value of r.

The Becke-Lebedev integration grid can be saved as an HDF5 file to be passed as an argument to the command-line scripts (see the example in section IV E) or loaded in another script using `BeckeMolGrid.from_hdf5` constructor. `HORTON` also supports constructing Gaussian Cube grids for visualization (e.g., molecular iso-density surfaces) and electrostatic potential fitting in section IV F.

D. `Meanfield`: Self-Consistent-Field (SCF) calculations

This module performs Hartree-Fock and Kohn-Sham DFT calculations. As such, it is built on the previously-described low-level modules. `Meanfield` is organized into classes that represent various energy contributions. Density functional theory (DFT) functionals are provided by LibXC[37], which is dynamically linked just like LibInt2[34].

The simplest usage of this module is to perform a Hartree-Fock calculation. Because the Hamiltonian is constructed as a list of terms, each of which is implemented in an object-oriented style, it is easy to test new ideas. A variety of SCF solvers are implemented, including conventional (Pulay) DIIS[38], EDIIS[39], and optimal damping[40].

```
import numpy as np
from horton import *

# Construct a molecule from scratch (coordinates should be given in a.u.)
mol = IOData(title="carbon dioxide")
mol.coordinates = np.array([[1.16, 0.0, 0.0], [0.0, 0.0, 0.0], [-1.16,
    0.0, 0.0]]) * angstrom
mol.numbers = np.array([8, 6, 8])

# Create a Gaussian basis set
obasis = get_gobasis(mol.coordinates, mol.numbers, "def2-tzvpd")

# Compute Gaussian integrals in AO basis
olp = obasis.compute_overlap()
kin = obasis.compute_kinetic()
na = obasis.compute_nuclear_attraction(mol.coordinates, mol.pseudo_numbers
    )
er = obasis.compute_electron_repulsion()

# Construct the restricted HF effective Hamiltonian
external = {"nn": compute_nucnuc(mol.coordinates, mol.pseudo_numbers)}
```

```
terms = [
    RTwoIndexTerm(kin, "kin"),
    RDirectTerm(er, "hartree"),
    RExchangeTerm(er, "x_hf"),
    RTwoIndexTerm(na, "ne"),
]
ham = REffHam(terms, external)

# Create alpha molecular orbitals (MO)
orb_alpha = Orbitals(obasis.nbasis)

# Construct initial guess for MO coefficients and assign orb_alpha.coeffs
guess_core_hamiltonian(olp, kin + na, orb_alpha)

# Choose MO occupation model and assign orb_alpha.occupations
occ_model = AufbauOccModel(11)
occ_model.assign(orb_alpha)

# Construct the initial density matrix (needed for CDIIS)
dm_alpha = orb_alpha.to_dm()

# Conventional DIIS SCF solver
scf_solver = CDIISSCFSolver(1e-6)
scf_solver(ham, olp, occ_model, dm_alpha)

# Update mol object and print energy
mol.energy = ham.cache["energy"]
mol.obasis = obasis
mol.orb_alpha = orb_alpha
print(f"Energy (a.u.) = {mol.energy: .5f}")
```

```
50
51 # Write calculation output as a MOLDEN file
52 mol.to_file("carbon_dioxide_hf_def2tzvpd.molden")
```

Listing 5. Run HF/Def2-TZVPD calculation for carbon monoxide and write MOLDEN file.

One of the strengths of HORTON, which we have employed in various research projects,[41] is its ability to support unconventional DFT calculations. As a relatively simple example, for a hybrid Meta-GGA calculation, the Hamiltonian in the previous example can be replaced with:

```
1  # Define a numerical integration grid needed for the XC functionals
2  grid = BeckeMolGrid(mol.coordinates, mol.numbers, mol.pseudo_numbers)
3
4  # Construct XC terms
5  libxc_term_x = RLibXCHybridMGGA("x_m05")
6  libxc_term_c = RLibXCMGGA("c_m05")
7  terms = [
8      RTwoIndexTerm(kin, "kin"),
9      RDirectTerm(er, "hartree"),
10     RGridGroup(obasis, grid, [libxc_term_x, libxc_term_c]),
11     RExchangeTerm(er, "x_hf", libxc_term_x.get_exx_fraction()),
12     RTwoIndexTerm(na, "ne"),
13 ]
14 ham = REffHam(terms, external)
```

Listing 6. Constructing a Hybrid Meta-GGA Hamiltonian Operator.

E. Part: Atoms-in-Molecules Partitioning and Population Analysis

Part implements various methods for partitioning molecular electron density into atomic contributions. Specifically, it includes Becke[35], Hirshfeld[42], iterative Hirshfeld (HI)[43], iterative stockholder partitioning (ISA)[44], and minimal basis iterative stockholder (MBIS)[45–47] methods. It also includes the orbital-based Mulliken population analysis method. Part was

extensively used for the development of additive[48–50] and polarizable force fields.[18] It also exemplifies how HORTON can provide specialized functionality to a larger program (e.g., to update the atomic charges in a molecular dynamics simulation).[19,51–55]

```
import numpy as np
from horton import *

# Load the Gaussian FCHK output from HORTON's test data directory
fn_fchk = context.get_fn("test/water_b3lyp_ccpvtz.fchk")
mol = IOData.from_file(fn_fchk)

# Partition the density with the Minimal Basis Iterative Stockholder (MBIS) scheme
grid = BeckeMolGrid(mol.coordinates, mol.numbers, mol.pseudo_numbers, mode="only")
moldens = mol.obasis.compute_grid_density_dm(mol.get_dm_full(), grid.points)
wpart = MBISWPart(mol.coordinates, mol.numbers, mol.pseudo_numbers, grid, moldens)
wpart.do_charges()
wpart.do_moments()
print("Atomic Charges  = ", wpart["charges"])
print("Atomic Momenets = ", wpart["radial_moments"])

# Write the result to a file
np.savetxt("charges.txt", wpart["charges"])
```

Listing 7. Minimal basis iterative stockholder (MBIS) partitioning of electron density.

The same calculation can be done directly on the command-line using the following command:

```
$ horton-wpart.py test/water_b3lyp_ccpvtz.fchk output.h5 mbis
```

Listing 8. MBIS partitioning of electron density.

F. ESPFit: Electrostatic Potential Fitting

This module can either fit atomic charges or test atomic charges ability to reproduce the molecular electrostatic potential (ESP) on a set of user-defined points for isolated molecules and periodic systems. Similar to the Hu-Lu-Yang approach[56], ESPFit uses a quadratic cost function defined as an integral rather than a sum over sample points. The ESP cost function can be customized to accommodate different charge-fitting protocols, e.g., constraints or bond-charge increments. ESPFit can work with any program that generates a Gaussian Cube file; for periodic systems, this cube file can be generated by CP2K[57]. ESP fitting can be run directly from command-line.

```
# Generate cost function
$ horton-esp-cost.py esp.cube cost.h5 --wdens=rho.cube --pbc={000|111}

# Fit charges to the cost function
$ horton-esp-fit.py cost.h5 charges.h5
```

Listing 9. Fit atomic charges to an electrostatic potential.

V. REENVISIONING HORTON

While we still maintain and use HORTON 2.x, it has become clear that the monolithic style of HORTON 2 and its reliance on nonstandard home-built C++ libraries prevent us from achieving truly modular code and easy installation. In January 2019, the leading developers of HORTON conceived HORTON 3. Based on lessons learned from our mistakes, we are rewriting HORTON modules to *strictly* enforce modularity and exploit NumPy vectorized operations and other standard Python libraries to the fullest. Our goal is to make HORTON even easier to understand, install, maintain, and contribute to. In addition, to ensure that all HORTON modules have a clear scope, clean API, and minimal dependency, they are each maintained as separate (public) GitHub repositories so they can be installed and used independently. This also distributes the workload of developing a library and accommodates the reality that no one person can be an expert on the entire edifice of HORTON 3.

HORTON 3 is conceived as a set of workflows that bring together independent modules, providing a modular and flexible architecture with a rich set of utilities and customizable

features for quantum chemistry, thereby extending the functionality that was previously available in HORTON 2.x. While this structure is unusual for modern packages, it has a strong precedent in quantum chemistry, most notably in Ernest Davidson's MELD package.[58] We believe that distributing separate, interoperable, modules is still valuable for innovative method development research in quantum chemistry.

The first step of HORTON's modularization was to entirely split off its post-processing capabilities and incorporate them into the larger ChemTools[1,2] package, which shares the same principles of modularity and protocols for quality assurance as HORTON 3. The other HORTON modules are:

- **IOData**[59] is a module for reading, writing, and converting various file formats from molecular and solid-state electronic structure theory software packages, and provides functionality for writing input files for various quantum chemistry software packages.

- **Grid** is a module for performing operations on molecular integration grids including integration, differentiation, interpolation, and nearly-arbitrary differential equation solving, with support for periodic boundary conditions and for local grids.

- **GBasis** is a module for performing a wide variety of 1- and 2-electron integrals of Gaussian basis functions and evaluating molecular descriptors like the electron density, electrostatic potential, and other density-based quantities, with an emphasis on properties related to the kinetic-energy density.

- **MeanField** is a module for self-consistent-field calculations, chiefly (constrained) Hartree-Fock and Kohn-Sham density-functional theory (DFT).

- **GOpt** is a module for geometry optimization, including robust methods for finding transition states and reaction pathways using redundant internal coordinates.

- **Fanpy**[33] is a module for exploring new post-SCF methods, including geminal-product wavefunctions, tensor-product wavefunctions, and unconventional coupled-cluster wavefunctions.

Beyond these core modules, there are additional modules attuned to the specific interests of various contributing research groups, like BFit[60] for basis function fitting and Procrustes[61] for matrix similarity analysis. For example, DensPart is a post-processing

tool for electron-density partitioning. PyCI and Fanpy support selected configuration interaction and the development of new, more advanced, correlated wavefunction methods.[33] ModelHamiltonian allows quantum-chemistry packages like HORTON to be applied to many different types of model Hamiltonians, including those which are ubiquitous in condensed matter physics.

VI. SUMMARY

Since its inception in 2011, HORTON has aspired to provide useful functionality (chiefly related to the performing and post-processing of Hartree-Fock and Kohn-Sham DFT calculations) while adhering to the ever-evolving "best practices" for scientific software design, including version control, automated and manual quality-assurance checks, thorough documentation, and comprehensive automated testing. As shown through the above examples, HORTON 2.3 can be used not only as a stand-alone program, but also as module(s) within a larger workflow. We never intended HORTON to be a workhorse for routine electronic structure calculations. Instead, HORTON is intended to be a helpful open-source research tool for testing new ideas and teaching quantum chemistry. We believe that its clear documentation, thorough testing, and modular structure support this mandate.

HORTON is, and always will be, a work in progress: we continually strive to add features and improve its utility for both scientific exploration and pedagogical purposes. By breaking HORTON into modules, we hope to enhance its usability, increase its durability, and ensure its sustainability.

DATA AVAILABILITY STATEMENT

The data and code that support the findings of this study are either included in the paper or are free and openly available in the HORTON repository at https://github.com/theochem/horton. HORTON's last source code and any of its official releases can be directly downloaded from its GitHub repository. In addition, HORTON can be installed using the conda package manager from the theochem channel.

FUNDING INFORMATION

The primary funding for HORTON has been provided by a series of Natural Sciences and Engineering Research Council (NSERC) of Canada to PWA and his research team. TV's contributions have been partly supported by ongoing support from the Foundation of Scientific Research-Flanders (FWO) and the Research Board of Ghent University (BOF). FHZ's early contributions were subsidized by a series of fellowships from the Canadian Government (Vanier CGS, NSERC), the European Commission (Marie Curie individual postdoctoral fellowship), and the Flemish government (FWO postdoctoral fellowship), and her recent contributions have been funded by an NSERC Discovery Grant and Queen's University Research Initiation Grant. EVM has been supported primarily by FONDECYT No. 1200369 (Chile). Many of the students and postdocs who have contributed to HORTON over the years were supported by other fellowships from the Canadian government (chiefly from NSERC). Most of the computational resources needed to develop HORTON were provided by Compute Canada, a Research Tools and Instruments grant from NSERC, and the Flemish Supercomputer Center (VSC).

ACKNOWLEDGMENTS

Over the history of HORTON, there have been many contributors beyond those listed as authors. We are grateful for the help of these scientists in the development and improvement of HORTON, and especially wish to cite the contributions of the extended HORTON 2.x development team (Katharina Boguslawski, Pawel Tecmer, Cristina E. González-Espinoza, Stijn Fias, Marco Franco, Steven Vandenbrande, Diego Berrocal, Yilin Zhao, Peter Limacher, Carlos Cardenas, and Ali Malek, in addition to many of the coauthors of this paper), the people who have become core users of HORTON (including Paul Johnson and Ramon Alain Miranda-Quintana) and the people who have hosted HORTON workshops, hackathons, and training courses (including Carlos Cardenas, Julia Contreras, Shubin Liu, and Chunying Rong).

CONFLICT OF INTEREST

There is no conflict of interest.

REFERENCES

REFERENCES

[1]F. Heidar-Zadeh, M. Richer, S. Fias, R. A. Miranda-Quintana, M. Chan, M. Franco-Perez, C. E. Gonzalez-Espinoza, T. D. Kim, C. Lanssens, A. H. G. Patel, X. D. Yang, E. Vohringer-Martinez, C. Cardenas, T. Verstraelen, and P. W. Ayers, "An explicit approach to conceptual density functional theory descriptors of arbitrary order," Chemical Physics Letters **660**, 307–312 (2016).

[2]L. Pujal, A. Tehrani, and F. Heidar-Zadeh, "Chemtools: Gain chemical insight form quantum chemistry calculations," in *Conceptual Density Functional Theory: Towards a New Chemical Reactivity Theory*, edited by S. Liu (Wiley, 2022) 1st ed.

[3]V. Chuiko, W. Adams, A. Richards, G. Sanchez-Diaz, M. Richer, Y. Zhao, F. Heidar-Zadeh, and P. W. Ayers, "ModelHamiltonian," (2022).

[4]J. Lehtola, M. Hakala, A. Sakko, and K. Hämäläinen, "ERKALE–a flexible program package for x-ray properties of atoms and molecules," Journal of Computational Chemistry **33**, 1572–1585 (2012).

[5]R. M. Parrish, L. A. Burns, D. G. A. Smith, A. C. Simmonett, A. E. DePrince, 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, 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," Journal of Chemical Theory and Computation **13**, 3185–3197 (2017).

[6]I. Fdez. Galván, M. Vacher, A. Alavi, C. Angeli, F. Aquilante, J. Autschbach, J. J. Bao, S. I. Bokarev, N. A. Bogdanov, R. K. Carlson, L. F. Chibotaru, J. Creutzberg, N. Dattani, M. G. Delcey, S. S. Dong, A. Dreuw, L. Freitag, L. M. Frutos, L. Gagliardi, F. Gendron, A. Giussani, L. González, G. Grell, M. Guo, C. E. Hoyer, M. Johansson, S. Keller, S. Knecht, G. Kovačević, E. Källman, G. Li Manni, M. Lundberg, Y. Ma, S. Mai, J. P. Malhado, P. A. Malmqvist, P. Marquetand, S. A. Mewes, J. Norell, M. Olivucci, M. Oppel, Q. M. Phung, K. Pierloot, F. Plasser, M. Reiher, A. M. Sand, I. Schapiro, P. Sharma, C. J. Stein, L. K. Sørensen, D. G. Truhlar, M. Ugandi, L. Ungur, A. Valentini, S. Vancoillie,

V. Veryazov, O. Weser, T. A. Wesołowski, P.-O. Widmark, S. Wouters, A. Zech, J. P. Zobel, and R. Lindh, "OpenMolcas: From source code to insight," Journal of Chemical Theory and Computation **15**, 5925–5964 (2019).

[7]M. Valiev, E. Bylaska, N. Govind, K. Kowalski, T. Straatsma, H. V. Dam, D. Wang, J. Nieplocha, E. Apra, T. Windus, and W. de Jong, "NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations," Computer Physics Communications **181**, 1477–1489 (2010).

[8]K. Aidas, C. Angeli, K. L. Bak, V. Bakken, R. Bast, L. Boman, O. Christiansen, R. Cimiraglia, S. Coriani, P. Dahle, E. K. Dalskov, U. Ekström, T. Enevoldsen, J. J. Eriksen, P. Ettenhuber, B. Fernández, L. Ferrighi, H. Fliegl, L. Frediani, K. Hald, A. Halkier, C. Hättig, H. Heiberg, T. Helgaker, A. C. Hennum, H. Hettema, E. Hjertenæs, S. Høst, I.-M. Høyvik, M. F. Iozzi, B. Jansík, H. J. A. Jensen, D. Jonsson, P. Jørgensen, J. Kauczor, S. Kirpekar, T. Kjærgaard, W. Klopper, S. Knecht, R. Kobayashi, H. Koch, J. Kongsted, A. Krapp, K. Kristensen, A. Ligabue, O. B. Lutnæs, J. I. Melo, K. V. Mikkelsen, R. H. Myhre, C. Neiss, C. B. Nielsen, P. Norman, J. Olsen, J. M. H. Olsen, A. Osted, M. J. Packer, F. Pawlowski, T. B. Pedersen, P. F. Provasi, S. Reine, Z. Rinkevicius, T. A. Ruden, K. Ruud, V. V. Rybkin, P. Sałek, C. C. M. Samson, A. S. de Merás, T. Saue, S. P. A. Sauer, B. Schimmelpfennig, K. Sneskov, A. H. Steindal, K. O. Sylvester-Hvid, P. R. Taylor, A. M. Teale, E. I. Tellgren, D. P. Tew, A. J. Thorvaldsen, L. Thøgersen, O. Vahtras, M. A. Watson, D. J. D. Wilson, M. Ziolkowski, and H. Ågren, "The Dalton quantum chemistry program system," WIREs Computational Molecular Science **4**, 269–284 (2014).

[9]L. Lemmens, X. D. Vriendt, D. Tolstykh, T. Huysentruyt, P. Bultinck, and G. Acke, "GQCP: The ghent quantum chemistry package," The Journal of Chemical Physics **155**, 084802 (2021).

[10]G. J. R. Aroeira, M. M. Davis, J. M. Turney, and H. F. I. Schaefer, "Fermi.jl: A modern design for quantum chemistry," Journal of Chemical Theory and Computation **18**, 677–686 (2022).

[11]J. P. Unsleber, T. Dresselhaus, K. Klahr, D. Schnieders, M. B"ockers, D. Barton, and J. Neugebauer, "Serenity: A subsystem quantum chemistry program," J. Comput. Chem. **39**, 788–798 (2018).

[12]D. Poole, J. L. Galvez Vallejo, and M. S. Gordon, "A new kid on the block: Application of julia to hartree–fock calculations," Journal of Chemical Theory and Computation **16**,

5006–5013 (2020).

[13]Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, S. Wouters, and G. K.-L. Chan, "PySCF: the python-based simulations of chemistry framework," WIREs Computational Molecular Science **8**, e1340 (2018).

[14]K. Boguslawski, A. Leszczyk, A. Nowak, F. Brzęk, P. S. Żuchowski, D. Kędziera, and P. Tecmer, "Pythonic black-box electronic structure tool (PyBEST). an open-source python platform for electronic structure calculations at the interface between chemistry and physics," Computer Physics Communications **264**, 107933 (2021).

[15]K. Boguslawski, F. Brzęk, R. Chakraborty, K. Cieślak, S. Jahani, A. Leszczyk, A. Nowak, E. Sujkowski, J. Świerczyński, S. Ahmadkhani, D. Kędziera, M. H. Kriebel, P. S. Żuchowski, and P. Tecmer, "PyBEST: Improved functionality and enhanced performance," Computer Physics Communications **297**, 109049 (2024).

[16]Q. Wu and T. Van Voorhis, "Direct optimization method to study constrained systems within density-functional theory," Physical Review A **72**, 024502 (2005).

[17]T. Verstraelen, P. W. Ayers, V. Van Speybroeck, and M. Waroquier, "ACKS2: Atom-condensed kohn-sham DFT approximated to second order," Journal of Chemical Physics **138** (2013), 10.1063/1.4791569.

[18]T. Verstraelen, S. Vandenbrande, and P. W. Ayers, "Direct computation of parameters for accurate polarizable force fields," Journal of Chemical Physics **141** (2014), 10.1063/1.4901513.

[19]E. Vohringer-Martinez, T. Verstraelen, and P. W. Ayers, "The influence of ser-154, cys-113, and the phosphorylated threonine residue on the catalytic reaction mechanism of pin1," Journal of Physical Chemistry B **118**, 9871–9880 (2014).

[20]T. Verstraelen, D. Van Neck, P. W. Ayers, V. Van Speybroeck, and M. Waroquier, "The gradient curves method: An improved strategy for the derivation of molecular mechanics valence force fields from ab initio data," Journal of Chemical Theory and Computation **3**, 1420–1434 (2007).

[21]P. W. Ayers, "Charge transfer and polarization in force fields: An ab initio approach based on the (atom-condensed) kohn–sham equations, approximated by second-order perturbation theory about the reference atoms (ACKS2)," in *Conceptual Density Functional Theory: Towards a New Chemical Reactivity Theory*, edited by S. Liu (Wiley, 2022) 1st

ed., pp. 603–629.

[22]Q. Wu and T. Van Voorhis, "Constrained density functional theory and its application in long-range electron transfer," Journal of Chemical Theory and Computation **2**, 765–774 (2006).

[23]Q. Wu, P. W. Ayers, and Y. K. Zhang, "Density-based energy decomposition analysis for intermolecular interactions with variationally determined intermediate state energies," Journal of Chemical Physics **131**, 164112 (2009).

[24]Y. Shao, Z. Gan, E. Epifanovsky, A. T. B. Gilbert, M. Wormit, J. Kussmann, A. W. Lange, A. Behn, J. Deng, X. Feng, D. Ghosh, M. Goldey, P. R. Horn, L. D. Jacobson, I. Kaliman, R. Z. Khaliullin, T. Kús, A. Landau, J. Liu, E. I. Proynov, Y. M. Rhee, R. M. Richard, M. A. Rohrdanz, R. P. Steele, E. J. Sundstrom, H. L. Woodcock III, P. M. Zimmerman, D. Zuev, B. Albrecht, E. Alguire, B. Austin, G. J. O. Beran, Y. A. Bernard, E. Berquist, K. Brandhorst, K. B. Bravaya, S. T. Brown, D. Casanova, C.-M. Chang, Y. Chen, S. H. Chien, K. D. Closser, D. L. Crittenden, M. Diedenhofen, R. A. DiStasio Jr., H. Dop, A. D. Dutoi, R. G. Edgar, S. Fatehi, L. Fusti-Molnar, A. Ghysels, A. Golubeva-Zadorozhnaya, J. Gomes, M. W. D. Hanson-Heine, P. H. P. Harbach, A. W. Hauser, E. G. Hohenstein, Z. C. Holden, T.-C. Jagau, H. Ji, B. Kaduk, K. Khistyaev, J. Kim, J. Kim, R. A. King, P. Klunzinger, D. Kosenkov, T. Kowalczyk, C. M. Krauter, K. U. Lao, A. Laurent, K. V. Lawler, S. V. Levchenko, C. Y. Lin, F. Liu, E. Livshits, R. C. Lochan, A. Luenser, P. Manohar, S. F. Manzer, S.-P. Mao, N. Mardirossian, A. V. Marenich, S. A. Maurer, N. J. Mayhall, C. M. Oana, R. Olivares-Amaya, D. P. O'Neill, J. A. Parkhill, T. M. Perrine, R. Peverati, P. A. Pieniazek, A. Prociuk, D. R. Rehn, E. Rosta, N. J. Russ, N. Sergueev, S. M. Sharada, S. Sharmaa, D. W. Small, A. Sodt, T. Stein, D. Stück, Y.-C. Su, A. J. W. Thom, T. Tsuchimochi, L. Vogt, O. Vydrov, T. Wang, M. A. Watson, J. Wenzel, A. White, C. F. Williams, V. Vanovschi, S. Yeganeh, S. R. Yost, Z.-Q. You, I. Y. Zhang, X. Zhang, Y. Zhou, B. R. Brooks, G. K. L. Chan, D. M. Chipman, C. J. Cramer, W. A. Goddard III, M. S. Gordon, W. J. Hehre, A. Klamt, H. F. Schaefer III, M. W. Schmidt, C. D. Sherrill, D. G. Truhlar, A. Warshel, X. Xua, A. Aspuru-Guzik, R. Baer, A. T. Bell, N. A. Besley, J.-D. Chai, A. Dreuw, B. D. Dunietz, T. R. Furlani, S. R. Gwaltney, C.-P. Hsu, Y. Jung, J. Kong, D. S. Lambrecht, W. Liang, C. Ochsenfeld, V. A. Rassolov, L. V. Slipchenko, J. E. Subotnik, T. Van Voorhis, J. M. Herbert, A. I. Krylov, P. M. W. Gill, and M. Head-Gordon, "Advances in molecular

quantum chemistry contained in the q-chem 4 program package," Mol. Phys. **113**, 184–215 (2015).

[25]P. A. Johnson, P. A. Limacher, T. D. Kim, M. Richer, R. Alain Miranda-Quintana, F. Heidar-Zadeh, P. W. Ayers, P. Bultinck, S. De Baerdemacker, and D. Van Neck, "Strategies for extending geminal-based wavefunctions: Open shells and beyond," Computational and Theoretical Chemistry **1116**, 207–219 (2017).

[26]P. A. Limacher, T. D. Kim, P. W. Ayers, P. A. Johnson, S. De Baerdemacker, D. Van Neck, and P. Bultinck, "The influence of orbital rotation on the energy of closed-shell wavefunctions," Molecular Physics **112**, 853–862 (2014).

[27]P. Tecmer, K. Boguslawski, P. A. Johnson, P. A. Limacher, M. Chan, T. Verstraelen, and P. W. Ayers, "Assessing the accuracy of new geminal-based approaches," Journal of Physical Chemistry A **118**, 9058–9068 (2014).

[28]P. A. Johnson, P. W. Ayers, P. A. Limacher, S. De Baerdemacker, D. Van Neck, and P. Bultinck, "A size-consistent approach to strongly correlated systems using a generalized antisymmetrized product of nonorthogonal geminals," Computational and Theoretical Chemistry **1003**, 101–113 (2013).

[29]P. A. Limacher, P. W. Ayers, P. A. Johnson, S. De Baerdemacker, D. Van Neck, and P. Bultinck, "A new mean-field method suitable for strongly correlated electrons: computationally facile antisymmetric products of geminals," Journal of Chemical Theory and Computation **9**, 1394–1401 (2013).

[30]T. D. Kim, R. A. Miranda-Quintana, M. Richer, and P. W. Ayers, "Flexible ansatz for n-body configuration interaction," Computational and Theoretical Chemistry **1202**, 113187 (2021).

[31]K. Boguslawski, P. Tecmer, P. A. Limacher, P. A. Johnson, P. W. Ayers, P. Bultinck, S. De Baerdemacker, and D. Van Neck, "Projected seniority-two orbital optimization of the antisymmetric product of one-reference orbital geminal," Journal of Chemical Physics **140**, 214114 (2014).

[32]C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," Nature **585**, 357–362 (2020).

[33]T. D. Kim, M. Richer, G. Sánchez-Díaz, R. A. Miranda-Quintana, T. Verstraelen, F. Heidar-Zadeh, and P. W. Ayers, "Fanpy: A python library for prototyping multideterminant methods in ab initio quantum chemistry," Journal of Computational Chemistry **44**, 697–709 (2023).

[34]E. F. Valeev, "Libint: A library for the evaluation of molecular integrals of many-body operators over gaussian functions," http://libint.valeyev.net/ (2022), version 2.8.0.

[35]A. D. Becke, "A multicenter numerical integration scheme for polyatomic molecules," The Journal of Chemical Physics **88**, 2547–2553 (1988).

[36]V. I. Lebedev and D. N. Laikov, "A quadrature formula for the sphere of the 131st algebraic order of accuracy," Doklady Mathematics **59**, 477–481 (1999).

[37]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**, 1–5 (2018).

[38]P. Pulay, "Improved scf convergence acceleration," Journal of Computational Chemistry **3**, 556–560 (1982).

[39]K. N. Kudin, G. E. Scuseria, and E. Cancès, "A black-box self-consistent field convergence algorithm: One step closer," The Journal of Chemical Physics **116**, 8255–8261 (2002).

[40]E. Cancès and C. Le Bris, "Can we outperform the diis approach for electronic structure calculations?" International Journal of Quantum Chemistry **79**, 82–90 (2000).

[41]C. E. Gonzalez-Espinoza, P. W. Ayers, J. Karwowski, and A. Savin, "Smooth models for the coulomb potential," Theoretical Chemistry Accounts **135**, 256 (2016).

[42]F. L. Hirshfeld, "Bonded-atom fragments for describing molecular charge densities," Theor. Chim. Act. **44**, 129–138 (1977).

[43]P. Bultinck, C. Van Alsenoy, P. W. Ayers, and R. Carbó-Dorca, "Critical analysis and extension of the Hirshfeld atoms in molecules," J. Chem. Phys. **126**, 144–111 (2007).

[44]T. C. Lillestolen and R. J. Wheatley, "Redefining the atom: atomic charge densities produced by an iterative stockholder approach," Chem. Commun. **45**, 5909–5911 (2008).

[45]T. Verstraelen, S. Vandenbrande, F. Heidar-Zadeh, L. Vanduyfhuys, V. Van Speybroeck, M. Waroquier, and P. W. Ayers, "Minimal basis iterative stockholder: Atoms in molecules for force-field development," J. Chem. Theory Comp. **12**, 3894–3912 (2016).

[46]F. Heidar-Zadeh, P. W. Ayers, T. Verstraelen, I. Vinogradov, E. Vöhringer-Martinez, and P. Bultinck, "Information-theoretic approaches to atoms-in-molecules: Hirshfeld family of

partitioning schemes," The Journal of Physical Chemistry A **122**, 4219–4245 (2018).

[47]F. Heidar-Zadeh and P. W. Ayers, "How pervasive is the Hirshfeld partitioning?" J. Chem. Phys. **142**, 044–107 (2015).

[48]L. Vanduyfhuys, S. Vandenbrande, T. Verstraelen, R. Schmid, M. Waroquier, and V. Van Speybroeck, "Quickff: A program for a quick and easy derivation of force fields for metal-organic frameworks from ab initio input," Journal of Computational Chemistry **36**, 1015–1027 (2015).

[49]S. Vandenbrande, M. Waroquier, V. V. Speybroeck, and T. Verstraelen, "The monomer electron density force field (medff): A physically inspired model for noncovalent interactions," Journal of Chemical Theory and Computation **13**, 161–179 (2017).

[50]S. Vandenbrande, M. Waroquier, V. Van Speybroeck, and T. Verstraelen, "Ab initio evaluation of henry coefficients using importance sampling," Journal of Chemical Theory and Computation **14**, 6359–6369 (2018).

[51]D. A. Saez and E. Vöhringer-Martinez, "A consistent S-Adenosylmethionine force field improved by dynamic Hirshfeld-I atomic charges for biomolecular simulation." J Computer-Aided Molecular Design **29**, 951–961 (2015).

[52]A. Lara, M. Riquelme, and E. Vöhringer-Martinez, "Partition coefficients of methylated DNA bases obtained from free energy calculations with molecular electron density derived atomic charges." Journal of Computational Chemistry **93**, 1281 (2018).

[53]M. Riquelme, A. Lara, D. L. Mobley, T. Verstraelen, A. R. Matamala, and E. Vöhringer-Martinez, "Hydration Free Energies in the FreeSolv Database Calculated with Polarized Iterative Hirshfeld Charges." Journal of Chemical Information and Modeling **58** (2018).

[54]J. Oller, D. A. Saez, and E. Vöhringer-Martinez, "Atom-Condensed Fukui Function in Condensed Phases and Biological Systems and Its Application to Enzymatic Fixation of Carbon Dioxide," Journal of Physical Chemistry A **124**, 849–857 (2020).

[55]A. Gomez and E. Vöhringer-Martinez, "Conformational sampling and polarization of asp26 in pka calculations of thioredoxin," Proteins: Structure, Function, and Bioinformatics **87**, 467–477 (2019).

[56]H. Hu, Z. Lu, and W. Yang, "Fitting molecular electrostatic potentials from quantum mechanical calculations," Journal of Chemical Theory and Computation **3**, 1004–1013 (2007), pMID: 26627419.

[57]T. D. Kühne, M. Iannuzzi, M. Del Ben, V. V. Rybkin, P. Seewald, F. Stein, T. Laino, R. Z. Khaliullin, O. Schütt, F. Schiffmann, D. Golze, J. Wilhelm, S. Chulkov, M. H. Bani-Hashemian, V. Weber, U. Borštnik, M. Taillefumier, A. S. Jakobovits, A. Lazzaro, H. Pabst, T. Müller, R. Schade, M. Guidon, S. Andermatt, N. Holmberg, G. K. Schenter, A. Hehn, A. Bussy, F. Belleflamme, G. Tabacchi, A. Glöß, M. Lass, I. Bethune, C. J. Mundy, C. Plessl, M. Watkins, J. VandeVondele, M. Krack, and J. Hutter, "CP2K: An electronic structure and molecular dynamics software package - quickstep: Efficient and accurate electronic structure calculations," The Journal of Chemical Physics **152**, 194103 (2020).

[58]E. R. Davidson, "MELD: A many electron description," in *MOTECC-94: Methods and Techniques in Computational Chemistry*, Vol. B, edited by E. Clementi (STEF, Calgliari, Italy, 1993) pp. 209–274.

[59]T. Verstraelen, W. Adams, L. Pujal, A. Tehrani, B. D. Kelly, L. Macaya, F. Meng, M. Richer, R. Hernández-Esparza, X. D. Yang, *et al.*, "IOData: A python library for reading, writing, and converting computational chemistry file formats and generating input files," Journal of Computational Chemistry **42**, 458–464 (2021).

[60]A. Tehrani, J. S. M. Anderson, D. Chakraborty, J. I. Rodriguez-Hernandez, D. C. Thompson, T. Verstraelen, P. W. Ayers, and F. Heidar-Zadeh, "An information-theoretic approach to basis-set fitting of electron densities and other non-negative functions," Journal of Computational Chemistry **44**, 1998–2015 (2023).

[61]F. Meng, M. Richer, A. Tehrani, J. La, T. D. Kim, P. W. Ayers, and F. Heidar-Zadeh, "Procrustes: A python library to find transformations that maximize the similarity between matrices," Computer Physics Communications **276**, 108334 (2022).