

# ORBKIT: A Modular Python Toolbox for Cross-Platform Postprocessing of Quantum Chemical Wavefunction Data

Gunter Hermann,<sup>\*,[a]†</sup> Vincent Pohl,<sup>[a]†</sup> Jean Christophe Tremblay,<sup>[a]</sup> Beate Paulus,<sup>[a]</sup> Hans-Christian Hege,<sup>[b]</sup> and Axel Schild<sup>[c]</sup>

ORBKIT is a toolbox for postprocessing electronic structure calculations based on a highly modular and portable Python architecture. The program allows computing a multitude of electronic properties of molecular systems on arbitrary spatial grids from the basis set representation of its electronic wavefunction, as well as several grid-independent properties. The required data can be extracted directly from the standard output of a large number of quantum chemistry programs. ORBKIT can be used as a standalone program to determine standard quantities, for example, the electron density, molecular orbitals, and derivatives thereof. The cornerstone of ORBKIT is its modular structure. The existing basic functions can be arranged in an individual way and can be easily extended by user-written mod-

ules to determine any other derived quantity. ORBKIT offers multiple output formats that can be processed by common visualization tools (VMD, Molden, etc.). Additionally, ORBKIT possesses routines to order molecular orbitals computed at different nuclear configurations according to their electronic character and to interpolate the wavefunction between these configurations. The program is open-source under GNU-LGPLv3 license and freely available at <https://github.com/orbkit/orbkit/>. This article provides an overview of ORBKIT with particular focus on its capabilities and applicability, and includes several example calculations. © 2016 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.24358

## Introduction

In today's computational and theoretical chemistry, quantum chemical methods (or electronic structure methods) are routinely applied for the investigation of molecular systems. Modern quantum chemical programs are characterized by their general applicability, increasing functionality, and high efficiency due to methodological and numerical progress in the field. There exists a wide range of such program packages, covering different levels of theory and offering assorted features. The spectrum extends from open source packages, for example, GAMESS-US,<sup>[1]</sup> PSI4,<sup>[2]</sup> or Tonto,<sup>[3]</sup> over commercial closed source programs such as Gaussian,<sup>[4]</sup> Molpro,<sup>[5]</sup> Turbomole,<sup>[6]</sup> or Q-Chem,<sup>[7]</sup> to software freely available for academic usage such as ORCA<sup>[8]</sup> or NWChem.<sup>[9]</sup> Depending on the methodological requirements of a quantum chemical problem, the user has to deal with a multitude of differently formatted input and output data. In this context, projects such as the Atomic Simulation Environment (ASE),<sup>[10]</sup> the Basis Set Exchange library,<sup>[11,12]</sup> cclib,<sup>[13]</sup> and OpenBabel<sup>[14]</sup> contributed tremendous standardization efforts.

Despite this software diversity, most quantum chemical programs dealing with molecules share the same basic approach to the solution of the time-independent molecular Schrödinger equation: They solve it for clamped nuclei, and they use an atom-centered Gaussian basis set to represent the electronic wavefunction at the selected nuclear configuration. A typical output of a quantum chemical calculation contains not only the energy and other relevant properties of the molecular system, but also the expansion coefficients of the electronic wavefunction in the selected basis set. The latter allow the calculation of additional quantities for the system characterization. Conceivable quantities

include those based on the reconstructed electronic wavefunction, for example, the electron density, and others quantities, such as molecular orbitals (MO), that are constructed from various combinations of the basis set and the expansion coefficients. These quantities are typically represented on a grid in the configuration space of one electron, which facilitates their analysis and enables their visualization. The necessary postprocessing tools for the analysis are sparsely implemented in most of the quantum chemical program packages. Additionally, there is usually no possibility to adjust the postprocessing parameters, for example, grid parameters, or to request further quantities after finishing the electronic structure calculation.

To overcome this problem, two strategies can be pursued: the modification or extension of a quantum chemical program package, or the usage of standalone postprocessing programs. The first

[a] G. Hermann, V. Pohl, J. C. Tremblay, B. Paulus  
Institut für Chemie und Biochemie, Freie Universität Berlin, Takustraße 3,  
Berlin 14195, Germany  
E-mail: [gunter.hermann@fu-berlin.de](mailto:gunter.hermann@fu-berlin.de)

[b] H.-C. Hege  
Department of Visual Data Analysis, Zuse Institute Berlin, Takustraße 7,  
Berlin 14195, Germany

[c] A. Schild  
Max-Planck-Institut für Mikrostrukturphysik, Weinberg 2, Halle 06120,  
Germany

†These authors contributed equally to this work.

Contract grant sponsor: Deutsche Forschungsgemeinschaft (DFG) through the project TR1109/2-1 and Ma 515/25-1; Contract grant sponsor: Elsa-Neumann foundation of the Land Berlin; The authors acknowledge the Unit Scientific Computing Services of the Zentraleinrichtung für Datenverarbeitung (ZEDAT) at Freie Universität Berlin for allocation of computer time.

© 2016 Wiley Periodicals, Inc.

approach is practicable only for open source and well-documented programs; additionally it is a formidable, time-consuming work to understand, adapt, and extend the respective source code. For the second approach, a handful of specialized tools are available offering diverse functionalities. An easy visualization of the molecular structure, the MOs, the electron density, etc., based on the output of a quantum chemical program, can be carried out with programs such as Molden<sup>[15]</sup> or Avogadro.<sup>[16]</sup> To calculate properties from the electronic wavefunction (i.e., from the basis set used and the coefficients obtained in the electronic structure calculation) the programs Checkden,<sup>[17,18]</sup> DGrid,<sup>[19]</sup> Multiwfn,<sup>[20]</sup> or DensToolKit<sup>[21]</sup> are well-suited and provide an impressive number of features. However, if the desired feature is not already available, extending these codes may become prohibitively difficult.

For such situations, we have developed the Python toolbox ORBKIT, which meets all common requirements of postprocessing electronic wavefunctions. ORBKIT stands out by its broad applicability in terms of postprocessing electronic structure data. It offers similar features such as Checkden, DGrid, or Multiwfn, and it can be employed as a standalone program for investigating in detail the characteristics of a molecular system and for visualizing important position space quantities. Its modular design allows the user to combine the individual functions in any manner. Furthermore, ORBKIT also comes with a library and application programming interface. This can be used to create new programs, without in-depth knowledge of the internal structure. Additionally, the library can be extended by user-written functions. Programming is greatly facilitated using Python as major language with its user-friendly syntax and large number of function libraries. Consequently, also non-standard or new problems can be quickly solved by adding new user-written functions to the standalone program ORBKIT, or by combining existing functions and new functions in a user-written program. In this article, we want to present the capabilities and several selected applications of ORBKIT.

The paper is structured as follows. Section “Methodology” briefly introduces the theoretical background, and Section “Program” describes the structure and main aspects of ORBKIT. Then, we present several “Practical Applications” that illustrate the features of ORBKIT, followed by a conclusion.

## Methodology

In this section, we present the main functions implemented in ORBKIT for postprocessing results from electronic structure calculations. Quantities that can be constructed from these fundamental components and that are included in ORBKIT are not listed here but presented in Section “Program”. All theoretical aspects and quantities considered in Section “Practical Applications” will be briefly discussed there, in the respective subsection. We use atomic units throughout the article.

### The Electronic Wavefunction

For the solution of the time-independent molecular Schrödinger equation for clamped nuclei, most molecular quantum chemistry methods introduce localized one-electron basis set functions to expand the many-body electronic wavefunction. Accordingly,

a standard output of a quantum chemical program provides the data to reconstruct the electronic wavefunction, that is, the expansion coefficients of the wavefunction, the selected atom-centered basis set (the atomic orbitals), and the nuclear configuration (position and type of the nuclei). Gaussian-type orbitals are by far the most commonly used atom-centered basis sets, and they can be handled with ORBKIT.

In general, the many-body electronic wavefunction is arranged in the form of a single Slater determinant or a linear combination of multiple Slater determinants. These are defined as antisymmetrized products of  $N$  one-electron functions, which correspond to orthonormal MOs. In the MO-LCAO (Molecular Orbital—Linear Combination of Atomic orbitals) ansatz, each MO  $\varphi_a$  can be reconstructed by the linear expansion of a finite set of contracted Gaussian basis functions<sup>[22]</sup>

$$\varphi_a(\mathbf{r}) = \sum_A^{N_A} \sum_i^{N_{AO}} C_{ia} \psi_i(\mathbf{r} - \mathbf{R}_A), \quad (1)$$

where  $C_{ia}$  is the  $i$ th MO coefficient for the MO  $a$ ,  $\psi_i$  is the respective atomic orbital centered at atom  $A$ ,  $\mathbf{r}$  are the Cartesian coordinates of one electron,  $\mathbf{R}_A$  denotes the spatial coordinates of nucleus  $A$ ,  $N_{AO}$  represents the number of atomic orbitals, and  $N_A$  is the number of atoms.

The atomic orbitals correspond to the real-valued contracted Gaussian basis functions, which are defined by the linear combination of primitive Gaussian functions<sup>[23]</sup>

$$\psi_i(\mathbf{r}_A) = \sum_p^L d_p g_p(\mathbf{r}_A, \alpha_p, l_p, m_p, n_p), \quad (2)$$

where  $d_p$  labels the contraction coefficients,  $L$  is the length of the contraction, and  $\mathbf{r}_A = \mathbf{r} - \mathbf{R}_A$  is the position vector of an electron relative to the nucleus  $A$ .

A primitive Cartesian Gaussian function  $g_p$  has the form<sup>[23]</sup>

$$g_p(\mathbf{r}, \alpha_p, l_p, m_p, n_p) = N_p(\alpha_p, l_p, m_p, n_p) x^{l_p} y^{m_p} z^{n_p} \exp(-\alpha_p r^2), \quad (3)$$

where  $x$ ,  $y$ , and  $z$  are Cartesian coordinates,  $r = \sqrt{x^2 + y^2 + z^2}$  is the magnitude of  $\mathbf{r}$ ,  $\alpha_p$  labels the Gaussian orbital exponents, and  $l_p$ ,  $m_p$ , and  $n_p$  are declared as exponents whose sum determines the angular momentum and, thus, the type of the orbital, for example,  $l = l_p + m_p + n_p = 0$  for an s-orbital.

The normalization constant  $N_p$  is given by<sup>[23]</sup>

$$N_p(\alpha_p, l_p, m_p, n_p) = \sqrt{\left(\frac{2\alpha_p}{\pi}\right)^{\frac{3}{2}} \frac{(4\alpha_p)^{l_p + m_p + n_p}}{(2l_p - 1)!! (2m_p - 1)!! (2n_p - 1)!!}}. \quad (4)$$

In addition to the Cartesian Gaussians, ORBKIT can process spherical harmonic Gaussians using the transformation described by Schlegel and Frisch.<sup>[24]</sup>

As a position space one-electron quantity, the experimentally observable electron density can provide further insights for the analysis of electronic and chemical characteristics of the system, for instance, bonding properties. For a single Slater determinant ansatz, the one-electron density reads

$$\rho_e(\mathbf{r}) = \sum_a^{N_{\text{occ}}} n_a |\varphi_a(\mathbf{r})|^2, \quad (5)$$

where  $N_{\text{occ}}$  is the number of singly or doubly occupied MOs with the respective occupation number  $n_a$ . For multideterminant wavefunctions, as obtained from configuration interaction methods or coupled cluster methods, the one-electron density can be constructed, for instance, using the Slater–Condon rules or by converting the wavefunction into a single Slater determinant representation built from natural orbitals. These orbitals possess non-integer occupation numbers  $n_a$  between zero and two.<sup>[22]</sup> The default version of ORBKIT supports all single-determinant wavefunctions. Hence, it can directly be used for the results of a Hartree–Fock or Density Functional Theory calculation as well as of a Post-Hartree–Fock calculation in natural orbital representation. The evaluation of quantities directly from a multideterminant wavefunction can be accomplished using the Slater–Condon rules mentioned above. Depending on the complexity of the wavefunction, the level of difficulty and effort for this task varies significantly. An essential prerequisite is a profound understanding of the underlying basis set expansion of the wavefunction.<sup>[25]</sup> With that, it is feasible to implement self-written modules in ORBKIT that extract the required data from quantum chemical outputs and evaluate one- and two-body operators over the multideterminant wavefunction. This has been accomplished with ORBKIT, for example, for a configuration interaction singles and for a multireference wavefunction.<sup>[26,27]</sup>

### Analytical Derivatives and Integrals

Further basic components, which can be derived from an electronic structure calculation and are relevant for the determination of other postprocessing quantities, include analytical derivatives and integrals of the basis functions, MOs, and of the electron density.

One of these components is the gradient of the electron density with respect to the electronic coordinates, which has the general form

$$\vec{\nabla} \cdot \rho_e(\mathbf{r}) = 2 \sum_a^{N_{\text{occ}}} n_a \left( \varphi_a(\mathbf{r}) \vec{\nabla} \cdot \varphi_a(\mathbf{r}) \right) \quad (6)$$

for real-valued functions.  $\vec{\nabla} \cdot \varphi_a(\mathbf{r})$  is constructed from the analytical gradients of the primitive Gaussian functions within the MO-LCAO ansatz

$$\vec{\nabla} \cdot g_p(\alpha_p, l_p, m_p, n_p, \vec{r}) = \begin{pmatrix} l_p x^{-1} & -2\alpha_p x \\ m_p y^{-1} & -2\alpha_p y \\ n_p z^{-1} & -2\alpha_p z \end{pmatrix} g_p(\alpha_p, l_p, m_p, n_p, \vec{r}). \quad (7)$$

Note that the contraction coefficients  $d_p$  from eq. (2) and the MO expansion coefficients  $C_{ia}$  from eq. (1) are independent of the electronic coordinates and thus not affected by the derivative operator. The gradients can be used to calculate, for example, the transition electronic flux density, as shown in an application below.

Closely related to the gradient is the Laplacian of the electron density, which is defined as

$$\nabla^2 \rho_e(\mathbf{r}) = \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) \rho_e(\mathbf{r}). \quad (8)$$

By revealing information about the local depletion and concentration of the electron density, the Laplacian plays a key role in the specification of bonding properties, and therefore, it is also used in the Atoms in Molecules (AIM) theory.<sup>[28–31]</sup>

To complete the set of essential functions incorporated in ORBKIT, we introduce the MO overlap matrix

$$\langle \varphi_a | \varphi_b \rangle = \sum_i^{N_{\text{AO}}} \sum_j^{N_{\text{AO}}} C_{ia} C_{jb} \langle \psi_i | \psi_j \rangle \quad (9)$$

and the atomic orbital overlap matrix

$$\langle \psi_i | \psi_j \rangle = \sum_p^L \sum_q^L d_{pi} d_{qj} \langle g_p | g_q \rangle. \quad (10)$$

Here,  $\langle g_p | g_q \rangle$  is calculated based on the article of Hô and Hernández-Pérez.<sup>[32]</sup> Possible quantities that can be derived from these overlap matrices involve, for example, Mulliken and Löwdin atomic populations or total and transition dipole moments.<sup>[22]</sup>

### Program

For the development of ORBKIT, we pursued the goal to make it practically useful for a large group of users, ranging from end-users who are interested in a simple and straightforward calculation of standard quantities, via university teachers who want to illustrate the computational ideas of quantum chemistry, to developers looking for a toolbox that provides core functions to build on. To this end, we made a number of design decisions: First, we chose Python as a programming language because of its user-friendliness, its vast amount of standard libraries, and its cross-platform portability. The stable version of ORBKIT has been tested on Linux, Mac OS X, and Windows platforms. Furthermore, we tried to retain a readily comprehensible modular structure, that is, we implemented a broad set of functions that can be separately called in a user-assembled driver program. This design facilitates the execution of the single components of ORBKIT and opens up the opportunity to implement self-written features in combination with the already existing functions. Besides, there is a standalone version of ORBKIT which can calculate a selected number of quantities, such as the one-electron density or the MOs on a user-defined rectilinear grid. Its simple handling allows quickly getting started with ORBKIT. In general, we attempt to ensure a universal applicability that comprises the readability of standard quantum chemistry programs, the writing of output files which are easy to handle, and an adequate number of standard quantities. Table 1 gives an overview of the possible input and output file formats and lists the computable quantities.

**Table 1.** Available input and output formats as well as computable quantities and other features of ORBKIT.

Input	Output
Molden files AOMix files GAMESS-US output files Gaussian log-files Gaussian formatted checkpoint files wfn and wfx files cclib library	HDF5 files Gaussian cube files VMD script files ZIBAmiraMesh files and network files Mayavi visualization XYZ and PDB files
Standard Quantities	Additional Feature
Electron (spin) densities Atomic and molecular orbitals Orbital derivatives Gross atomic densities Molecular orbital transition flux densities Total dipole moments Mulliken and Löwdin charges	Input of real-space grid as regular grid or as point set Order molecular orbitals of different nuclear structures Interpolate between different nuclear structures Symmetry transformations of the Cartesian grid Center grid around specified nuclei Adaptive integration with Cubature

Concerning the efficiency of ORBKIT, we use the highly scalable NumPy<sup>[33]</sup> and SciPy<sup>[34]</sup> Python libraries for processing large, multidimensional arrays. Moreover, computationally expensive operations are implemented in C and Cython<sup>[35]</sup> and can be run on multiple processors using the Python package multiprocessing. Position space quantities are then calculated by dividing the grid into slices and distributing them on the requested number of processors, thus offering linearly scaling parallelization.

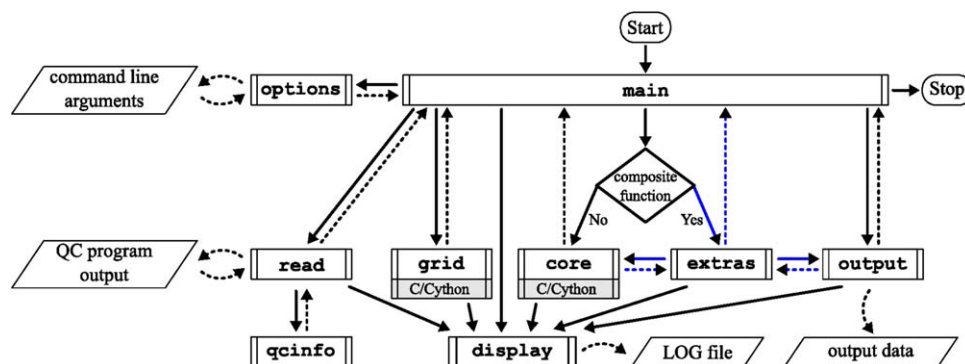
To understand how to use ORBKIT, it is recommended to read the detailed documentation and to work through the example applications in this article or in the ORBKIT example package. The documentation also contains function references for advanced usage. In addition, a flowchart visualizing the internal design of ORBKIT is depicted in Figure 1.

### Input and Output

In general, ORBKIT requires as main input the data of a single determinant wavefunction. To this end, it extracts the expansion coefficients of the MOs, the selected atomic basis set, and

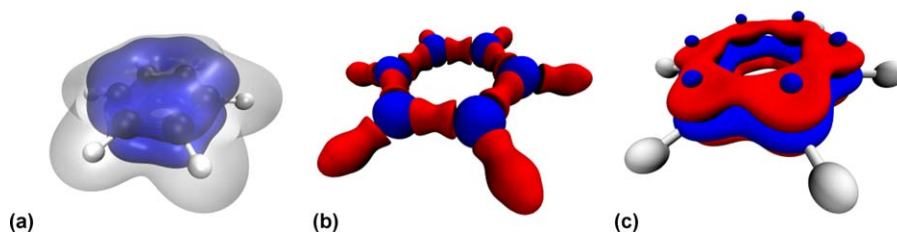
the specification of the molecular structure from the output of a quantum chemical calculation. The data files of almost all major quantum chemical programs can be handled with ORBKIT (cf. Table 1). Besides output files, such as Gaussian log-files, the Molden file format<sup>[36]</sup> is our main input format. This file format can be both directly written by some programs, such as Molpro,<sup>[5]</sup> and transformed with Molden<sup>[15]</sup> from output files of other program packages, such as GAMESS-US. In addition, there exists an interface to the library cclib,<sup>[13]</sup> which is a package-independent platform for parsing and extracting information of several computational chemistry programs. With this extension, many further file formats can be read. However, it is also straightforward to read the output of any other electronic structure program with a self-written Python routine and transfer it into the specific ORBKIT data structure. The data formats used by ORBKIT are described in detail in the documentation.

Based on the extracted data, ORBKIT can compute all standard position space functions (cf. Table 1) on an equidistant zero- to three-dimensional Cartesian grid with user-defined grid parameters. Besides, the calculation can be performed on



**Figure 1.** Schematic flowchart of ORBKIT's modular organization. Modules are illustrated as rectangles with double-struck vertical edges. Solid arrows represent a module call, dashed arrows signify data communication between two modules, and curved dashed lines represents Input/Output. The overall procedure is controlled by the program `main` and includes the following steps: (1) `options` reads command line arguments, for example, grid parameters or calculation settings, (2) `read` extracts the QC program output and `qinfo` stores the data, (3) `grid` initializes the grid with the user-defined parameters, (4) if composite functions are requested `extras` controls the ensuing operations (blue arrows), (5) `core` computes all grid-based quantities, and (6) `output` saves the data in a user-defined format. The calculation progress is written in a LOG file with `display`.





**Figure 2.** a) The electron densities of benzene for all electrons (gray) and for the  $\pi$ -electrons (blue). The isocontour value for the electron densities is  $0.01 a_0^{-3}$ . b) Laplacians of the molecular electron density and c) of the  $\pi$ -electron density for benzene. The isocontour values for the Laplacian of the total electron density are  $-0.5 a_0^{-5}$  (red) and  $0.5 a_0^{-5}$  (blue) and  $-0.1 a_0^{-5}$  (red) and  $0.1 a_0^{-5}$  (blue) for the  $\pi$ -electron density. The visualization was performed with VMD.

a list of  $(x, y, z)$  coordinates, which we call vector grids. This includes equidistant spherical coordinates, random grids, or user-defined arbitrary point sets. It is also possible to adapt the Cartesian grid to the molecular structure or to perform symmetry transformations on it.

During an ORBKIT calculation, a LOG file is written to record the basic information concerning the selected computational parameters, for example, grid type, grid parameter, chosen input or output file, or the progress of the calculation. Subsequently, the results, that is, the electron density, the MOs, etc., given on the user-defined grid, are typically saved as HDF5 files.<sup>[37]</sup> This hierarchical data format can efficiently store and organize numerical data with a small need of disk space. Furthermore, there are many programs and tools that support this data type.

For 3D visualizations with standard molecular graphics programs such as VMD,<sup>[38]</sup> ORBKIT provides the option to save the data as Gaussian Cube files. These plain text files contain the volumetric data, the grid parameters, and the atomic positions of the molecular system. Besides, ORBKIT can create VMD script files, which are directly callable with VMD for a quick depiction of any position space function. It is also possible to use a simple interface to Mayavi,<sup>[39]</sup> which enables an immediate and interactive visualization. In addition to the output of grid-dependent quantities, there exists the possibility to create XYZ and PDB<sup>[40]</sup> files with ORBKIT.

## Features

Apart from the usual wavefunction analysis (cf. “Standard Quantities” in Table 1), several quantum chemical outputs can be compared simultaneously to highlight for instance the influence of the change in the nuclear positions on the electronic structure. In this context, the ordering routine of ORBKIT should be mentioned, which enables the correct arrangement of the MOs for different nuclear configurations according to their overlap [cf. eq. (9)]. This procedure may be useful when the MOs change their symmetry and/or energetic ordering with the nuclear configuration. An example for the usage is given below. Another valuable feature of ORBKIT is the adaptive integration of multidimensional functions using a Python wrapper<sup>[41]</sup> for the C package Cubature.<sup>[42]</sup> This module integrates numerically vector-valued integrands over hypercubes. In the implemented  $h$ -adaptive integration scheme,<sup>[43,44]</sup> the integrands are iteratively evaluated by gradually adding more

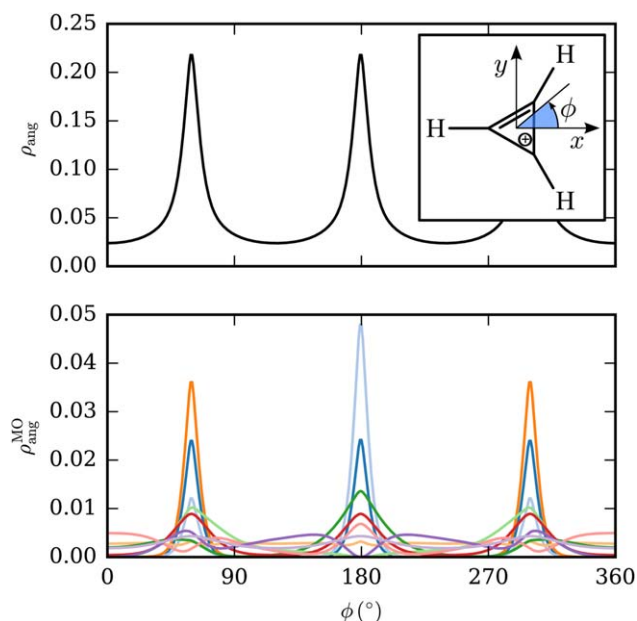
grid points until a user-defined tolerance is fulfilled. Especially for functions with localized sharp features such as the one-electron density, this is a well-suited technique. The basic procedure for the multidimensional integration with Cubature in ORBKIT can be outlined as follows: First, ORBKIT evaluates a given function on a predefined coarse grid. Subsequently, Cubature numerically computes the integral for this function and refines the grid in the subregions with largest estimated errors. This is followed by a function evaluation of ORBKIT for the added grid points and a further integral calculation with Cubature. These steps are iterated until either the absolute or relative error, evaluated using different orders of the cubature rules, falls below a user-defined threshold. In general, Cubature opens up the opportunity to evaluate the expectation value of any operator solely on the basis of the data from a quantum chemistry program output. This could include any one- and two-electron operator, like, for instance, Coulomb integrals. Additionally, multiple functions can be integrated simultaneously, reducing the number of calls to `core` functions (cf. Fig. 1).

## Practical Applications

In this section, we present applications to five molecules: benzene  $C_6H_6$ , the 2-cyclopropenyl cation  $(C_3H_3)^+$ , the hydrogen molecule ion  $H_2^+$ , carbon dioxide  $CO_2$ , and the water dimer. For all examples, the Molden data file and the execution command or an extensively commented Python execution code are provided in the ORBKIT package. Thus, the reader is encouraged to follow these examples interactively. Additional examples can be found in the “Supporting Information” and in the ORBKIT package. All electronic structure calculations are performed with Molpro<sup>[5]</sup> using the Hartree–Fock method and a cc-pVDZ basis set.<sup>[45]</sup> There are also some applications of ORBKIT in the literature, see Refs. [26,27,46–48].

### Electron Density of Benzene

In the first application, the usage of ORBKIT as a standalone program via the terminal interface and the subsequent visualization of grid-based one-electron quantities are demonstrated with the example of benzene. To characterize the nature of a chemical bond in a given quantum system, the calculation and analysis of the electron density and its Laplacians, as well as the partitioning of the electron density in certain subsets of electrons, are useful tools.<sup>[30,31,49–51]</sup> In the benzene molecule,



**Figure 3.** Angular electron density  $\rho_{\text{ang}}$  (upper panel) and integrated molecular orbital densities  $\rho_{\text{ang}}^{\text{MO}}$  (lower panel) for angular segments of  $\Delta\phi = 1^\circ$  for the 2-cyclopropenyl cation ( $\text{C}_3\text{H}_3^+$ ) using a Cubature interface. The inset in the upper panel shows the Lewis structure of the 2-cyclopropenyl cation and the orientation of the polar angle  $\phi$ . [Color figure can be viewed in the online issue, which is available at [wileyonlinelibrary.com](http://wileyonlinelibrary.com).]

for example, we can look at the  $\pi$ -electron density. The respective MOs can be identified by their nodal plane being the plane spanned by the nuclei. Hence, the associated  $\pi$ -electron density is distributed above and below the benzene ring. The electron density  $\rho_e(\mathbf{r})$  [cf. eq. (5)] for all electrons and for the specified set of electrons ( $\pi$ -electrons) and the respective Laplacians  $\nabla^2\rho_e(\mathbf{r})$  [cf. eq. (8)] are calculated with ORBKIT and visualized (cf. Fig. 2) with VMD.<sup>[38]</sup>

Tasks such as the selection of groups of MOs and the subsequent calculation of grid-based one-electron quantities are straightforward with ORBKIT. The associated data can be stored in output formats like Gaussian cube files. This facilitates its visualization with graphical programs such as VMD, etc. In addition, a direct visualization in ORBKIT is feasible also with Python packages, for example, matplotlib<sup>[52]</sup> or Mayavi.<sup>[39]</sup>

### Angular Electron Density of ( $\text{C}_3\text{H}_3$ )<sup>+</sup>

Our second application illustrates the ease of utilizing the ORBKIT library with an existing program. In this case, the interface of ORBKIT to Cubature<sup>[43,44]</sup> is introduced, and its virtue for integrating the density in a region of space is demonstrated. The example at hand shows the integration of the three-dimensional electron density  $\rho(x, y, z)$  of ( $\text{C}_3\text{H}_3$ )<sup>+</sup> to obtain the angular electron density  $\rho_{\text{ang}}(\phi)$ . The angle  $\phi$  is defined as the polar angle in the plane of the nuclei (cf. inset of Fig. 3). Thus, an integration over  $z$  and over  $r = \sqrt{x^2 + y^2}$  has to be performed. Additionally, to obtain a smooth angular density close to the positions of the nuclei, an averaging along  $\phi$  has to be done. Hence for each discrete point  $\phi_i$  on our

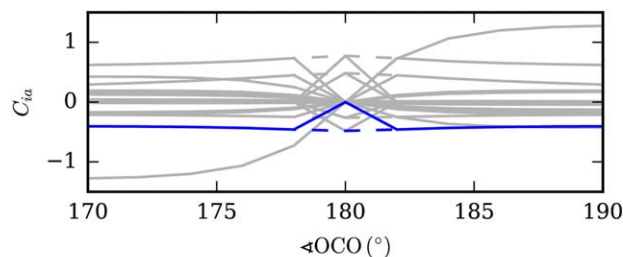
grid, we have to integrate over  $z \in [-\infty, \infty]$ ,  $r \in [0, \infty]$ , and  $\phi \in [\phi_i - \Delta\phi/2, \phi_i + \Delta\phi/2]$ .

The choice of grid coordinates is not as straightforward as it may seem. While cylindrical coordinates are a natural choice, they have a major disadvantage if used equidistantly: Because of the fixed number of points along  $\phi$ , there will be a high density of points for small  $r$  but very few for larger  $r$ . Thus, the number of points needed to accurately integrate the density around the nuclei quickly becomes prohibitively large. Calculating the density on a grid in Cartesian coordinates is not an alternative. While the point density in space does not change, the number of points per angle varies significantly. As a consequence, integration using Cartesian coordinates needs far too many points to calculate the angular density efficiently and can yield artifacts nevertheless (see chapter 3.4 of Ref. [53] for an example).

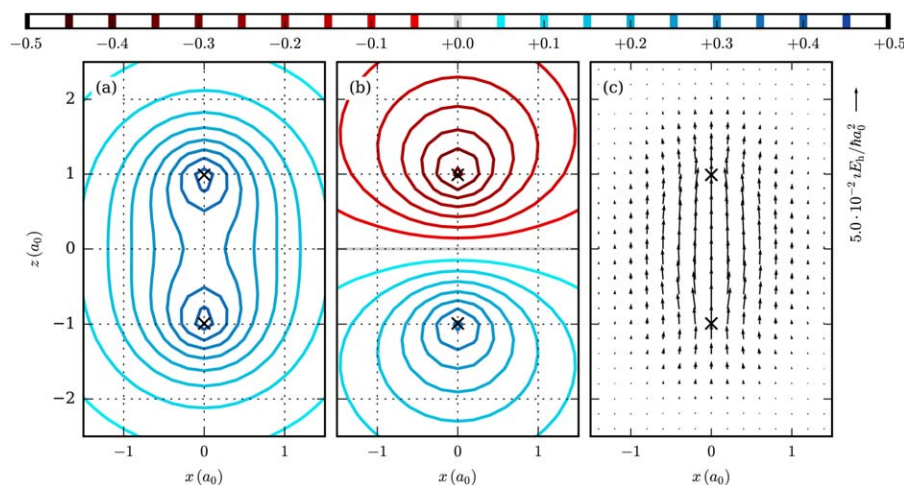
Thus, we implemented an interface to a Python wrapper<sup>[41]</sup> for Cubature.<sup>[42]</sup> This program can integrate multidimensional functions with moderate dimensionality adaptively to a specified error. A function for integrating in cylindrical coordinates converts the points asked for by Cubature (which are assumed to be cylindrical coordinates  $r, \phi, z$ ) into Cartesian coordinates, calls ORBKIT using the list of Cartesian vectors as input, and returns the density multiplied with  $r$  to account for the volume element of the integration. This implementation is straightforward, as ORBKIT is designed to be used as both a standalone program and a function library.

The upper and lower panel of Figure 3 show the angular density and the integrated molecular orbital densities, respectively. The integration of the regions of strong localization around the nuclear positions are well-converged, which would have been difficult to achieve using simple integration schemes on equidistant grids.

In general, the adaptive integration by Cubature of grid-based quantities computed in ORBKIT, in any user-defined



**Figure 4.** Upper panel: Molecular orbital coefficients  $C_{ia}$  of the MO (initially the LUMO) as a function of the  $\angle\text{OCO}$  angle before (solid line) and after (dashed line) sorting by ORBKIT. Lower panel: Isosurface plots of the lowest unoccupied molecular orbital (LUMO) of  $\text{CO}_2$  at an  $\text{O}-\text{C}-\text{O}$  angle of  $170^\circ$  (left), of the LUMO and LUMO + 1 (degenerate) at  $180^\circ$ , and of the LUMO at  $190^\circ$ . Solid and dashed arrows correspond to the solid and dashed lines in the upper panel. The isosurface value is  $\pm 0.1 a_0^{-3}$ . The isosurface plots were visualized with VMD. [Color figure can be viewed in the online issue, which is available at [wileyonlinelibrary.com](http://wileyonlinelibrary.com).]



**Figure 5.** Contour plots of selected molecular orbitals (MO) of the hydrogen molecule ion  $\text{H}_2^+$  for an internuclear distance of  $R = 1.4 a_0$ : a) MO  $1\sigma_g$  and b) MO  $1\sigma_u$ . c) Vector plot of the stationary transition electronic flux density (STEFED)  $J_{e,1\sigma_g,1\sigma_u}^{\text{STEFED}}$  for the transition between the state  $1\sigma_g$  and the state  $1\sigma_u$ . The nuclear positions are marked with black crosses. [Color figure can be viewed in the online issue, which is available at [wileyonlinelibrary.com](http://wileyonlinelibrary.com).]

volume, opens up a wide spectrum of conceivable applications, for example, the determination of Voronoi deformation density atomic charges, etc.

### Ordering Molecular Orbitals along a Reaction Coordinate

In quantum chemistry, it can be of interest to follow the change of molecular properties during the variation of the nuclear structure, for example, along a reaction coordinate. While the comparison of most properties is straightforward but possibly cumbersome because data may have to be extracted from several output files, the comparative analysis of the molecular orbitals can be a problem: Most quantum chemistry programs are sorting the orbitals according to their energy and additionally, if applicable, according to their symmetry. However, often the energetic order of the orbitals changes with the nuclear configuration, and it is necessary to order them according to a different criterion. For these issues, ORBKIT offers a set of useful functions to simultaneously handle multiple files, and it provides several MO ordering routines.

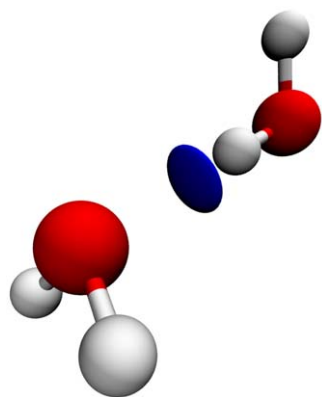
The most useful ordering routine sorts the MOs of different nuclear configurations and the associated signs according to their overlap. It starts from the first two structures in a list, computes the analytical overlap between all the orbitals [cf. eq. (9)] of both structures, and sorts them accordingly. Thereafter, it proceeds with the second and third structure, etc. Subsequently, the expansion coefficients for the ordered orbitals data can be interpolated using B-Splines to approximate intermediate nuclear configurations. B-Splines (cf. Ref. [54] and references therein) are piecewise polynomial functions with many useful properties, one of which is their analytical differentiability. This substantiates their usage for the accurate determination of non-adiabatic coupling terms.

To illustrate the functionality of this ordering routine, we performed a set of quantum chemical calculations for  $\text{CO}_2$ , varying the  $\angle \text{OCO}$  angle from  $170^\circ$  to  $190^\circ$  with a step size of  $2^\circ$ . For a sorted set of molecular orbitals, one expects

smooth curves for the coefficients as a function of the slightly modified nuclear configurations. In Figure 4 (upper panel), the MO coefficients are displayed for a selected orbital of  $\text{CO}_2$ . Solid lines correspond to the MO coefficients  $C_{ia}$  for the lowest unoccupied MO (LUMO) of the energetically ordered list of the quantum chemical output, and dashed lines signify the coefficients of the orbital after sorting them according to the MO overlap. To illustrate the difference between a sorted and unsorted MO list, the curve of one chosen coefficient is marked in blue. For linear  $\text{CO}_2$  ( $\angle \text{OCO} = 180^\circ$ ), the selected orbital, the LUMO, is energetically degenerate with another one, the LUMO + 1, which leads to an interchange of both in the energetically ordered list of the quantum chemistry program. Following the procedure described above, ORBKIT can sort all orbitals according to their overlap and in groups according to their symmetry properties, if this information is available. The lower panel of Figure 4 shows the orbitals that were incorrectly assigned to each other (solid arrows) and those assigned to each other after ordering (dashed arrows). Note that the results of the ordering routine depend on the validity of the overlap as a measure of the character of the orbital. Hence, the prerequisites for a successful MO sorting are moderate structural variations between the quantum chemical calculations and the identical orientation of the molecule. Nonetheless, a failure of the ordering routine can be easily detected and corrected by inspecting the graphs of, for example, the orbital energies or orbital coefficients, and using the manual ordering function of ORBKIT. To the best of our knowledge ORBKIT is the only postprocessing program that provides such an orbital ordering function. This is complemented by a number of convenience functions to save, load, and plot the computed quantities.

### Transition Electronic Flux Density of $\text{H}_2^+$

The example of this section illustrates the computation of the transition electronic flux density (TEFD) between two selected



**Figure 6.** Isosurface plot of the dimensionless reduced density gradient for the water dimer. The isocontour value is 0.5. The data visualization was performed with VMD. [Color figure can be viewed in the online issue, which is available at [wileyonlinelibrary.com](http://wileyonlinelibrary.com).]

electronic Born–Oppenheimer states in the hydrogen molecular ion  $H_2^+$  with ORBKIT.<sup>[26]</sup> In general, the TEFD is the nonvanishing component of the electronic flux density in the framework of the Born–Huang expansion and is defined for the transition from the electronic state  $\lambda$  to the electronic state  $\nu$  as

$$\mathbf{J}_{e,\lambda\nu}^{\text{TEFD}}(\mathbf{r}, t) = \int d\mathbf{R} \rho_{n,\lambda\nu}(\mathbf{R}, t) \cdot \mathbf{J}_{e,\lambda\nu}^{\text{TEFD}}(\mathbf{r}; \mathbf{R}) \quad (11)$$

with the nuclear transition density  $\rho_{n,\lambda\nu}(\mathbf{R}, t) = \chi_\lambda^*(\mathbf{R}, t)\chi_\nu(\mathbf{R}, t)$  and the purely imaginary static transition electronic flux density (STEFED)

$$\mathbf{J}_{e,\lambda\nu}^{\text{STEFED}}(\mathbf{r}; \mathbf{R}) = -\frac{i}{2} \left( \Psi_\lambda(\mathbf{r}; \mathbf{R}) \nabla_e \Psi_\nu(\mathbf{r}; \mathbf{R}) - \Psi_\nu(\mathbf{r}; \mathbf{R}) \nabla_e \Psi_\lambda(\mathbf{r}; \mathbf{R}) \right), \quad (12)$$

where  $\Psi_\lambda$ ,  $\Psi_\nu$  are the real-valued electronic wavefunctions, and the gradient  $\nabla_e$  is taken with respect to the electronic coordinates.<sup>[55]</sup> The TEFD is a crucial quantity to analyze the contributions to infrared absorption or vibrational circular dichroism spectra,<sup>[56,57]</sup> as a complement to the study of the adiabatic electronic flux density,<sup>[26]</sup> or for the visualization of electron processes, for example.

The hydrogen molecular ion  $H_2^+$  is the simplest diatomic molecule consisting of two protons and one electron. The fact that the molecular orbitals of  $H_2^+$  correspond to its electronic states simplifies the calculations of the TEFD. However, it is nonetheless feasible to determine this quantity for more complicated quantum systems with ORBKIT. This can be accomplished with the help of the Slater–Condon rules, but it is necessary to take into account the underlying basis set expansion of the wavefunction (e.g., single Slater determinant, configuration state functions, multireference configuration interaction representation, etc.). The required modules for this computational task can be easily incorporated into the modular structure of ORBKIT. In addition, the simple and platform-independent parallelization techniques within Python can be used to enhance the efficiency of the implemented code. Currently, ORBKIT is being used to investigate the ultrafast photoelectron transfer in a Dye-Sensitized Solar Cell by analyzing the corresponding time-dependent TEFD constructed from configuration interaction wavefunctions.

For the TEFD of  $H_2^+$ , the transition between the electronic ground state  $1\sigma_g$  and the first excited state  $1\sigma_u$  is selected, since it is experimentally accessible.<sup>[58]</sup> Contour plots of both electronic states (MOs) are displayed in Figures 5a and 5b showing their gerade and ungerade symmetry properties. The stationary TEFD  $\mathbf{J}_{e,1\sigma_g 1\sigma_u}^{\text{STEFED}}$  for the transition between the ground state  $1\sigma_g$  and the excited state  $1\sigma_u$  as a function of the x- and z-coordinate for an internuclear distance of  $R = 1.4 a_0$  is depicted in Figure 5c. As expected, this imaginary vector field shows a gerade parity for the transition between a gerade and an ungerade state.

### Exemplary Implementation of a Grid-Based Quantity

The following example provides a short tutorial for the implementation of a real space quantity into ORBKIT using the existing modules. The particular focus is on the creation and technical description of the Python code that is required for this task. As an example, the reduced density gradient is selected, which is an essential quantity for the description of

```

1 from numpy import sqrt, pi
2 from orbkit import read, grid, core, output
3 qc = read.main_read('h2o_dimer.molden', itype='molden')
4 grid.adjust_to_geo(qc, extend=1.0, step=0.1)
5 grid.grid_init()
6 R, dR = core.rho_compute(qc, drv=['x', 'y', 'z'], numproc=4)
7 s = sqrt((dR**2).sum(axis=0)) / (2*(3*pi**2)*(1/3.)*R**(4/3.))
8 s[R>0.05] = 1e3
9 output.main_output(s, qc.geo_info, qc.geo_spec, outputname='out',
10                   otype=['cb', 'vmd'], iso=(-0.5, 0.5))

```

**Figure 7.** Complete script for the implementation of the reduced density gradient on the basis of modules existing in ORBKIT.



intramolecular interactions. Its general form defined by Johnson et al.<sup>[59]</sup> reads

$$s(\mathbf{r}) = \frac{1}{2(3\pi^2)^{1/3}} \frac{|\nabla\rho(\mathbf{r})|}{\rho(\mathbf{r})^{4/3}}, \quad (13)$$

where  $|\nabla\rho(\mathbf{r})|$  is the norm of the gradient of the electron density. Exemplary, the reduced density gradient is calculated for a water dimer in accordance with Ref. [59] on the basis of a Hartree–Fock calculation with a cc-pVDZ basis set.

The general procedure of a postprocessing calculation in ORBKIT and the involved modules can be seen in Figure 1. To implement a new quantity into ORBKIT, it is necessary to build up a user-written script which controls all other modules and eventually determines and saves the desired quantity. This script undertakes all tasks of the `main` module (cf. Fig. 1) in the standalone version of ORBKIT. To start with, the required modules are imported via

```
from orbkit import read, grid, core, output
```

Note that the ORBKIT directory needs to be defined in the Python environment variables. Subsequently, the data of the quantum chemical calculation is extracted with the module `read`,

```
qc = read.main_read('h2o_dimer.molden',  
                  itype='molden')
```

Here, `qc` is an instance of the Python class `QCInfo`, which stores the specification of the molecular structure, the atomic basis set, and the expansion coefficients of the MOs. Next, the spatial grid is adapted to the nuclear configuration and initialized with `grid`

```
grid.adjust_to_geo(qc, extend=1.0, step=0.1)  
grid.grid_init()
```

The essential elements for the calculation of the reduced density gradient are the electron density and the derivatives thereof with respect to  $x$ ,  $y$ , and  $z$ . Their parallel computation is accomplished using the module `core`,

```
R, dR = core.rho_compute(qc, drv=['x', 'y', 'z'],  
                        numproc=4)
```

Referring to eq. (13), the following expression is implemented into the script,

```
from numpy import sqrt, pi  
s = sqrt((dR**2).sum(axis=0))  
s /= (2*(3*pi**2)**(1/3.))*R**(4/3.)  
s[ R>0.05 ] = 1e3
```

The last line of the code sets a density cutoff to focus solely on the noncovalent interaction region in the system. Finally, this low reduced density gradient is saved as Gaussian cube files using the `output` routine

```
output.main_output(s, qc.geo_info, qc.geo_spec,  
                  outputname='out',  
                  otype=['cb', 'vmd'],  
                  iso=(-0.5, 0.5))
```

In addition, a VMD script is created for immediate visualization. The respective isosurface plot is depicted in Figure 6. The complete script can be found in Figure 7 and in the ORBKIT package.


## Conclusion

ORBKIT is a modular designed Python toolbox that allows an individualized cross-platform postprocessing of quantum chemical data from electronic structure calculations. The variety of position space one-electron functions and fundamental quantities that has been implemented serves as the basis for sophisticated analyses of molecular wavefunctions. Thus, it is useful for a wide range of applications. In addition, in its current state of development ORBKIT offers multiple options and features for postprocessing issues. For the calculation of one-electron quantities on arbitrary grids, there exists a standalone version which is easy to handle and can be executed in parallel to speed up the computation. The results can be directly visualized with a simple and interactive viewer (Mayavi). This is complemented by the interoperability of ORBKIT with various external visualization programs. Besides the standalone execution, the functions existing in ORBKIT can be individually combined to enable a problem-specific wavefunction analysis. Furthermore, the possibility to add user-written Python functions into ORBKIT can foster the development of own postprocessing programs. For this purpose, only a basic understanding of the design and features of ORBKIT is required. The first steps into the program are facilitated by a detailed documentation and several application examples. Hence, ORBKIT appeals to novices as well as experienced theoretical chemists.

As a wavefunction analysis toolkit, ORBKIT differs from similar projects by its portability and its simple modular structures. They enable the user to build own application programs on top of ORBKIT with minimal effort and without recompiling. In conclusion, ORBKIT is a fairly mature open-source program that provides the basis for many possible further developments.

**Keywords:** quantum chemical calculation · electronic structure · molecular visualization · electron density · grid representation of one-electron quantities · molecular orbital ordering

How to cite this article: G. Hermann, V. Pohl, J. C. Tremblay, B. Paulus, H.-C. Hege, A. Schild. *J. Comput. Chem.* **2016**, *37*, 1511–1520. DOI: 10.1002/jcc.24358

 Additional Supporting Information may be found in the online version of this article.

- [1] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, J. A. Montgomery Jr, *J. Comput. Chem.* **1993**, *14*, 1347.
- [2] J. M. Turney, A. C. Simmonett, R. M. Parrish, E. G. Hohenstein, F. A. Evangelista, J. T. Fermann, B. J. Mintz, L. A. Burns, J. J. Wilke, M. L. Abrams, N. J. Russ, M. L. Leininger, C. L. Janssen, E. T. Seidl, W. D. Allen, H. F. Schaefer, R. A. King, E. F. Valeev, C. D. Sherrill, T. D. Crawford, *WIREs Comput. Mol. Sci.* **2012**, *2*, 556.
- [3] D. Jayatilaka, D. Grimwood, In *Computational Science - ICCS 2003*, Vol. 2660 of *Lecture Notes in Computer Science*; P. Sloot, D. Abramson, A.

- Bogdanov, Y. Gorbachev, J. Dongarra, A. Zomaya, Eds.; Springer: Berlin, Heidelberg, **2003**; pp. 142.
- [4] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, et al., Gaussian'09 Revision D.01, Gaussian Inc.: Wallingford, CT, 2009.
- [5] H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, M. Schütz, P. Celani, W. Györfy, D. Kats, T. Korona, R. Lindh, A. Mitrushenkov, G. Rauhut, K. R. Shamasundar, T. B. Adler, R. D. Amos, A. Bernhardsson, A. Berning, D. L. Cooper, M. J. O. Deegan, A. J. Dobyn, F. Eckert, E. Goll, C. Hampel, A. Hesselmann, G. Hetzer, T. Hrenar, G. Jansen, C. Köppl, Y. Liu, A. W. Lloyd, R. A. Mata, A. J. May, S. J. McNicholas, W. Meyer, M. E. Mura, A. Nicklaß, D. P. O'Neill, P. Palmieri, D. Peng, K. Pflüger, R. Pitzer, M. Reiher, T. Shiozaki, H. Stoll, A. J. Stone, R. Tarroni, T. Thorsteinsson, M. Wang, Molpro, Version 2012.1, A Package of Ab Initio Programs, **2012**. Available at: <http://www.molpro.net>.
- [6] TURBOMOLE V6.5, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007, **2013**. Available at: <http://www.turbomole.com>
- [7] Y. Shao, Z. Gan, E. Epifanovsky, A. T. Gilbert, M. Wormit, J. Kusmann, A. W. Lange, A. Behn, J. Deng, X. Feng, D. Ghosh, M. Goldedy, P. R. Horn, L. D. Jacobson, I. Kaliman, R. Z. Khaliullin, T. Kus, 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. Diefenbach, R. A. DiStasio Jr., H. Do, 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. Khistyayev, J. Kim, J. Kim, R. A. King, P. Klunzinger, D. Kosenkov, T. Kowalczyk, C. M. Krauter, K. U. Lao, A. D. 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, E. Neuscamman, C. M. Oana, R. Olivares-Amaya, D. P. O'Neill, J. A. Parkhill, T. M. Perrine, R. Peverati, A. Prociuk, D. R. Rehn, E. Rosta, N. J. Russ, S. M. Sharada, S. Sharma, D. W. Small, A. Sodt, T. Stein, D. Stück, Y.-C. Su, A. J. W. Thom, T. Tsuchimoto, V. Vanovschi, L. Vogt, O. Vydrov, T. Wang, M. A. Watson, J. Wenzel, A. White, C. F. Williams, J. Yang, S. Yeganeh, S. R. Yost, Z.-Q. You, I. Y. Zhang, X. Zhang, Y. Zhao, 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. Xu, 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. V. Voorhis, J. M. Herbert, A. I. Krylov, P. M. W. Gill, M. Head-Gordon. *Mol. Phys.* **2015**, *113*, 184.
- [8] F. Neese, *WIREs Comput. Mol. Sci.* **2012**, *2*, 73.
- [9] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T. L. Windus, W. A. de Jong. *Comput. Phys. Commun.* **2010**, *181*, 1477.
- [10] S. R. Bahn, K. W. Jacobsen, *Comput. Sci. Eng.* **2002**, *4*, 56.
- [11] D. Feller, *J. Comput. Chem.* **1996**, *17*, 1571.
- [12] K. L. Schuchardt, B. T. Didier, T. Elsethagen, L. Sun, V. Gurumoorthi, J. Chase, J. Li, T. L. Windus, *J. Chem. Inf. Model* **2007**, *47*, 1045.
- [13] N. M. O'Boyle, A. L. Tenderholt, K. M. Langner, *J. Comput. Chem.* **2008**, *29*, 839.
- [14] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, G. R. Hutchison, *J. Cheminform.* **2011**, *3*, 33.
- [15] G. Schaftenaar, J. Noordik, *J. Comput. Aided Mol. Des.* **2000**, *14*, 123.
- [16] M. Hanwell, D. Curtis, D. Lonie, T. Vandermeersch, E. Zurek, G. Hutchison, *J. Cheminform.* **2012**, *4*, 17.
- [17] L. F. Pacios, *Comput. Biol. Chem.* **2003**, *27*, 197.
- [18] L. F. Pacios, A. Fernandez, *J. Mol. Graph. Model.* **2009**, *28*, 102.
- [19] M. Kohout, DGrid, Version 4.6, Radebeul, **2011**.
- [20] T. Lu, F. Chen, *J. Comput. Chem.* **2012**, *33*, 580.
- [21] J. Solano-Altamirano, J. M. Hernández-Pérez, *Comput. Phys. Commun.* **2015**, *196*, 362.
- [22] F. Jensen, Introduction to Computational Chemistry; Wiley, **2007**.
- [23] S. Huzinaga, J. Andzelm, Gaussian Basis Sets for Molecular Calculations, Physical sciences data; Elsevier, **1984**.
- [24] H. B. Schlegel, M. J. Frisch, *Int. J. Quantum Chem.* **1995**, *54*, 83.
- [25] A. Szabo, N. Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory; Dover Publications, **1989**.
- [26] G. Hermann, B. Paulus, J. F. Pérez-Torres, V. Pohl, *Phys. Rev. A* **2014**, *89*, 052504
- [27] G. Hermann, J. C. Tremblay, *J. Phys. Chem. C* **2015**, *119*, 25606.
- [28] R. Bader, Atoms in Molecules: A Quantum Theory; Oxford University Press: New York, **1994**.
- [29] R. F. W. Bader, *Chem. Rev.* **1991**, *91*, 893.
- [30] R. Gillespie, P. Popelier, Chemical Bonding and Molecular Geometry: From Lewis to Electron Densities; Oxford University Press, **2001**.
- [31] P. Popelier, P. Popelier, Atoms in Molecules: An Introduction; Prentice Hall, **2000**.
- [32] M. Hö, J. M. Hernández-Pérez, *Math. J.* **2012**, *14*, 1.
- [33] S. van der Walt, S. C. Colbert, G. Varoquaux, *Comput. Sci. Eng.* **2011**, *13*, 22.
- [34] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open Source Scientific Tools for Python, **2001**. Available at <http://www.scipy.org/> (accessed on 28 September, 2015).
- [35] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn, K. Smith, *Comput. Sci. Eng.* **2011**, *13*, 31.
- [36] Molden File Format, Description, **2000**. Available at: [http://www.cmbi.ru.nl/molden/molden\\_format.html](http://www.cmbi.ru.nl/molden/molden_format.html) (accessed on 28 September, 2015).
- [37] The HDF Group, Hierarchical Data Format Version 5, **2000–2010**. Available at: <http://www.hdfgroup.org/HDF5> (accessed on 28 September, 2015).
- [38] W. Humphrey, A. Dalke, K. Schulten, *J. Mol. Graph.* **1996**, *14*, 33.
- [39] P. Ramachandran, G. Varoquaux, *Comput. Sci. Eng.* **2011**, *13*, 40.
- [40] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, *Nucleic Acids Res.* **2000**, *28*, 235.
- [41] S. Castro, Python Wrapper for the Cubature Algorithm, **2015**. Available at: <https://github.com/saullocaastro/cubature> (accessed on 28 September, 2015).
- [42] S. G. Johnson, Cubature (Multi-Dimensional Integration), **2015**. Available at <http://ab-initio.mit.edu/wiki/index.php/Cubature> (accessed on 28 September, 2015)
- [43] A. Genz, A. Malik, *J. Comput. Appl. Math.* **1980**, *6*, 295.
- [44] J. Bernsten, T. O. Espelid, A. Genz, *ACM Trans. Math. Softw.* **1991**, *17*, 437.
- [45] T. H. Dunning, *J. Chem. Phys.* **1989**, *90*, 1007.
- [46] T. Gomez, G. Hermann, X. Zarate, J. Pérez-Torres, J. C. Tremblay, *Molecules* **2015**, *20*, 13830.
- [47] V. Pohl, J. C. Tremblay, *Phys. Rev. A* **2016**, *93*, 012504.
- [48] M. Berg, B. Paulus, T. Bredtmann, *Mol. Phys.* **2015**, *1*.
- [49] A. Schild, D. Choudhary, V. D. Sambre, B. Paulus, *J. Phys. Chem. A* **2012**, *116*, 11355.
- [50] T. Lu, F. Chen, *J. Phys. Chem. A* **2013**, *117*, 3100.
- [51] L. Zhang, F. Ying, W. Wu, P. Hiberty, S. Shaik, *Chem. Eur. J.* **2009**, *15*, 2979.
- [52] J. D. Hunter, *Comput. Sci. Eng.* **2007**, *9*, 90.
- [53] A. Schild, Electron Fluxes During Chemical Processes in the Electronic Ground State; Freie Universität Berlin, **2013**.
- [54] H. Bachau, E. Cormier, P. Decleva, J. E. Hansen, F. Martín, *Rev. Prog. Phys.* **2001**, *64*, 1815.
- [55] L. A. Nafie, *J. Phys. Chem. A* **1997**, *101*, 7826.
- [56] T. B. Freedman, M. L. Shih, E. Lee, L. A. Nafie, *J. Am. Chem. Soc.* **1997**, *119*, 10620.
- [57] L. A. Nafie, Vibrational Optical Activity: Principles and Applications; Wiley, **2011**.
- [58] P. Bolognesi, L. Avaldi, M. MacDonald, C. Lopes, G. Dawber, C. Brion, Y. Zheng, G. King, *Chem. Phys. Lett.* **1999**, *309*, 171.
- [59] E. R. Johnson, S. Keinan, P. Mori-Sánchez, J. Contreras-García, A. J. Cohen, W. Yang, *J. Am. Chem. Soc.* **2010**, *132*, 6498.

Received: 18 December 2015  
Revised: 16 February 2016  
Accepted: 22 February 2016  
Published online on 4 April 2016