CCBDB.py: Automated and Accessible Computational Chemistry Benchmarking

William Adams

Candidate for the degree

Bachelor of Sciences

Submitted in partial fulfilment of the requirements for

College Honors

Departmental Distinction in Chemistry

Matt

Matthew Sonntag, Ph.D.

Nicholas Piro, Ph. D.

David Nawrocki

David Nawrocki, Ph. D.

F. Wilbur Gingrich Library
Special Collections Department
Albright College

Release of Senior Thesis

I hereby grant to the Special Collections Department of the F. Wilbur Gingrich Library at Albright College the nonexclusive right and privilege to reproduce, disseminate, or otherwise preserve the Senior Honors Thesis described below in any noncommercial manner that furthers the educational, research or public service purposes of Albright College. Copyright privileges remain with me, the author.

Title: CCBDB.py: Automated and Accessible Computational Chemistry Benchmarking

Signature of Author: ______________________ Date: 14 Dec 2018

Printed Name of Author: William H. Adams

Street Address: 107 Meadowlark Rd

City, State, Zip Code: Reading PA 19606

Abstract

Early approaches in computational chemistry attempted to develop a workhorse density functional theory (DFT) functional applicable for use in a variety of chemical applications. However, modern improvements have led to the development of numerous functionals, necessitating an automated method of benchmarking to determine appropriate functionals for different applications. An area where DFT struggles to attain chemical accuracy is in predicting the energy for non-covalent interactions (NCIs), an important class of interactions throughout every chemical discipline. One method of accounting for these interactions is by using an empirical dispersion correction. CCBDB.py (Computational Chemistry Benchmark DataBases) is a program implemented in the Python programming language and designed to benchmark NCI energies and optimize dispersion correction parameters using an automated workflow. This program is relevant to non-specialists looking to determine an appropriate functional for specific applied calculations and also for specialists formulating new functionals and dispersion corrections.

Introduction

Density functional theory has become a valuable tool for theorists and experimentalists alike. Due to the approximations inherent in DFT, an ever-expanding number of density functionals have been introduced, each with a range of strengths and weaknesses in describing particular chemical phenomena. While most DFT methods excel in describing short-range interactions such as covalent bonding, a consistent failure of DFT is in the description of non-covalent interactions, which act over longer distances. A common approach to address this deficiency is the application of an empirical dispersion correction to the energy calculation. In order to assess the performance of DFT methods and dispersion corrections, numerous benchmark databases have been developed, each containing systems of interest, with energies calculated using a high-level method for comparison.

In the following section, the foundations of DFT will be described, and the different classes of functionals will be outlined, highlighting specific popular functionals in each class. Next, the Grimme D3 dispersion correction will be presented, and each damping function will be presented. An example demonstrating the poor performance in DFT for noncovalent systems and the importance of dispersion corrections will be fully elucidated. Finally, a series of relevant NCI benchmark databases will be detailed, including both small benchmark sets and large supersets.

DFT methods

In 1964, Pierre Hohenberg and Walter Kohn proved that the ground state electronic energy of a system is fully dependent on the electron density (*1*). This proof was fundamental for the development of density functional theory (DFT). One major advantage of DFT over approaches such as a wave function approach is the computational efficiency of a density approach. While wave functions for an *N* electron system contain *4N* variables (three spatial and one spin coordinate for each electron), the electron density only depends on three spatial coordinates, regardless of the system size (*2*). However, the Hohenberg-Kohn Theorem does not specify the functional that links the electron density to the energy, and as such a large amount of effort has been focused towards developing approximate functionals to describe the ground states of many-electron systems (*3*).

Early DFT models attempted to find an energy functional in an orbital-free setting, in order to maintain the efficient three-variable system; these models proved to be fairly effective in certain metallic systems but failed to predict fundamental phenomena such as bonding. By introducing orbitals as a method of representing the electron density, Kohn and Lu Jeu Sham produced a more representative kinetic energy term, the main issue in orbital-free DFT (*4*). However, this approach increases the complexity of the system from *3* to *3N*, sacrificing part of the efficiency of DFT in exchange for better accuracy. In Kohn-Sham DFT, the total energy is decomposed into the form $E_{DFT}[\rho] = T_s[\rho] + E_{ne}[\rho] + J[\rho] + E_{xc}[\rho]$, where T is the total kinetic energy (assuming non-interacting electrons), E_{ne} and J are potential energy terms, and E_{xc} is the exchange-correlation term. The existence of E_{xc} is necessitated by the assumption of non-interacting electrons, and in essence serves as a catch-all term for the energetic effects that are ignored in T. These effects include electron correlation, where the movement of one electron is altered by interaction with another electron, and electron exchange, where identical electrons are repelled as a result of the Pauli Exclusion Principle. For most systems, even simple approximations of E_{xc} yield accurate results for most systems, however, the accurate description of effects such as dispersion, which are driven by electron interaction, will be highly dependent on the strength of this approximation (*2*).

Many popular functionals use a generalized gradient approximation (GGA) approach, making the exchange and correlation energies dependent on both the electron density and derivatives of the density. Some of these functionals, such as LYP (*5*), are parameterized using empirical data, while others, such as PBE (*6*), are derived from expected properties of the exact functional (*2*). Hybrid methods incorporate the exchange energy from Hartree-Fock theory and are typically parameterized using experimental data. The quintessential example of a hybrid method is B3LYP (*7*). In 2006, B3LYP was the functional of choice in 81% of the publications where a functional was mentioned in the title or abstract (*8*).

Range-separated methods are another class of DFT functionals that are gaining popularity in modern uses. These methods developed in order to limit self-interaction errors, which are particularly troublesome for Rydberg-type excited states and anionic systems with low electron affinities. In DFT, self-interaction errors manifest when loosely bound electrons are erroneously treated as unbound electrons; these electrons are bound by the most diffuse orbital and are therefore confined to the correct physical space by a well-chosen basis set but would drift away from the atom for an extended basis set. Range-separated methods address this issue by partitioning the electron-electron operator of the exchange energy into short-range and long-range parts, which leads to the correct description of loosely bound electrons due to correct theoretical methodology, not error cancellation (*2*); one popular functional of this type is ωB97X (*9*).

Dispersion Corrections

Hydrogen bonding and London dispersion forces are omnipresent throughout biological and macromolecular systems. Thus, an efficient and accurate treatment of these and other non-covalent interactions (NCIs) would be highly applicable to modern research in every chemical discipline. Coupled-cluster methods such as CCSD(T), generally considered to be the "gold standard" of chemical accuracy, provide highly accurate results for NCI systems but scale on the order of N^7, making these and similar methods inaccessible for even medium-sized systems ($N>20$). DFT methods scale on the order of $N^3 - N^5$, a much more modest computational cost, however, DFT consistently fails to accurately describe dispersion forces. This failure is due to the exchange-correlation approximations used in the design of each DFT functional (*10*). While the energies of short-range (local) interactions (<2 Å) are generally accurate within DFT, no component of the functional can properly describe the behavior of electrons through the medium-and long-range (non-local) scales. London dispersion is particularly challenging for DFT: the London dispersion force is generated by non-local electron interactions, but the exchange-correlation functional only acts locally (*11*).

The most popular method of dispersion correction is the additive empirical correction proposed by Stefan Grimme (*12*). This correction, denoted D/D2/D3 (D2 and D3 corresponding to methodological improvements over the original D correction) proposes a damped pairwise interatomic potential

$$\Delta E_{\text{disp.}} = -\sum_{AB}\sum_{n=6,8}\frac{C_{n,AB}}{r_{AB}^{n}}f_{d,n}(r_{AB}) \tag*{[1]}$$

where $C_{n,AB}$ are dispersion coefficients derived for every pair of atoms AB, r_{AB} is the interatomic distance, and $f_{d,n}$ is the damping function that ensures convergence of the dispersion correction. The r^{-6} distance-dependence of this correction is meant to mirror the non-local scale of London dispersion forces

This correction can take one of several forms, each tunable through a series of parameters specific to each functional and typically optimized at the def2-QZVP basis set. The original damping function approach, called zero damping (Equation 2), utilizes a cutoff radius R_0^{AB} tabulated for every atomic pair, a radius scaling parameter $s_{R,n}$, a global scaling parameter s_n, and steepness parameters α_n (*12*).

$$f_{d,n}(r_{AB}) = \frac{s_n}{1+6\left(\frac{r_{AB}}{s_{R,n}R_0^{AB}}\right)^{-\alpha_n}} \tag*{[2]}$$

Becke and Johnson proposed a dispersion correction model that explicitly incorporates the actual electronic structure of the system (*13*). BJ-damping (Equation 3) modifies the damping function by defining the cutoff radius as $R_0^{AB} = \sqrt{C_{8,AB}/C_{6,AB}}$ and introducing two free fit parameters, a_1 and a_2 (*14*).

$$f_{d,n}(r_{AB}) = \frac{s_n r_{AB}^n}{r_{AB}^n+\left(a_1R_0^{AB}+a_2\right)^n} \tag*{[3]}$$

Other damping functions have been proposed as modifications to the D3 methodology. The optimized power (OP) damping function attempts to generalize BJ-damping by adding the additional parameter β_6, such that $n = \beta_n$ in Equation 3 with $\beta_8 = \beta_6 + 2$ (*15*). Although OP-damping adds additional computational complexity during the optimization process due to the additional parameter, OP-damped corrections show slightly higher accuracy in comparison to BJ-damping. An opposite approach, C-six-only (CSO) damping, attempts to reduce the number of empirical parameters by employing a sigmoidal interpolation function incorporating only a single adjustable parameter; this approach yields results similar to BJ-damping (*16*).

Applying dispersion corrections to standard DFT methods drastically improves the chemical accuracy of DFT in non-covalent systems. As an example, we can consider the stability of the π-stacked adenine-thymine dimer (Figure 1). This system models the interaction between these two nucleobases, a common interaction within DNA, the genetic polymer that provides the basis for life on Earth. The benchmark calculations are calculated by the "gold standard" high-accuracy method CCSD(T), extrapolated to the complete basis set limit.

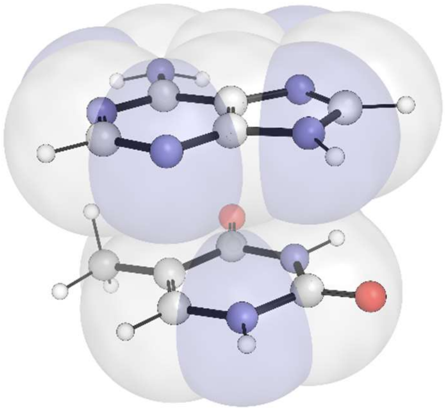


Figure 1 π-stacked adenine-thymine dimer at equilibrium intermolecular distance

Functional/Basis Set	Stabilization Energy (kcal/mol)	Difference, % Error (kcal/mol)
CCSD(T)/CBS	-11.730	--
B3LYP/def2-QZVP	+1.061	+ 12.729 (109%)
B3LYP-D3(BJ)/def2-QZVP	-11.950	– 0.220 (1.9%)

Table 1 Adenine-thymine dimer stabilization energy

Table 1 gives the stabilization energy (given as $E_{\mathrm{monomers}} - E_{\mathrm{dimer}}$) of the adenine-thymine dimer as calculated by B3LYP with and without the D3 correction, in comparison to the gold-standard energy calculation. This system is expected to be highly attractive, as DNA base-stacking interactions are important to the stability of DNA strands (*17*). The uncorrected B3LYP energy would predict a repulsive interaction, with an overall error of over 100%. The corrected B3LYP-D3 energy is well within the 1 kcal/mol threshold desired for a chemically accurate description of these systems.

Improving the accuracy of DFT in systems involving non-covalent interactions (NCI) is a desirable alternative to relying upon high-level calculations such as CCSD(T)/CBS, due to the considerable computational cost of performing these calculations. A variety of datasets, using high-accuracy calculations using coupled-cluster methods such as CCSD(T), have allowed for benchmarking the performance of computational methods including DFT. The following section will detail a variety of NCI benchmark sets that can be used to test the effectiveness of dispersion correction methods.

NCI Databases

NCI databases are designed to provide benchmarks for various non-covalent phenomena in chemical systems. Generally, these datasets present multiple phenomena relevant to a specific chemical discipline, such as biochemistry or materials science. While many earlier benchmark sets are composed entirely of equilibrium-geometry (van der Waals minimum) structures, later work has produced datasets containing radial curves, such that the benchmarks can test performance at non-equilibrium geometries. Reparameterization of dispersion corrections using radial curve datasets has yielded improved performance over the equilibrium-only parameters.

S22 and its Extensions

The S22 benchmark set (Figure 2) consists of 22 complexes ranging in size from small (6 atoms) to large (30 atoms) systems. The non-covalent interactions in the S22 set roughly fall into three categories: hydrogen bonded (H-bond) complexes, dispersion-stabilized complexes, and mixed (combination H-bond and dispersion) complexes. The basis for the design of this data set is as a screening set for the JSCH-2005 benchmark set, a larger set of biochemical benchmarks. While the extended set contains a large number of DNA base pair and amino acid pair complexes, and is mainly intended for the study of biomacromolecules, the S22 training set provides a broad baseline that is more broadly applicable to testing the general treatment of dispersion by different computational methods (*17*).

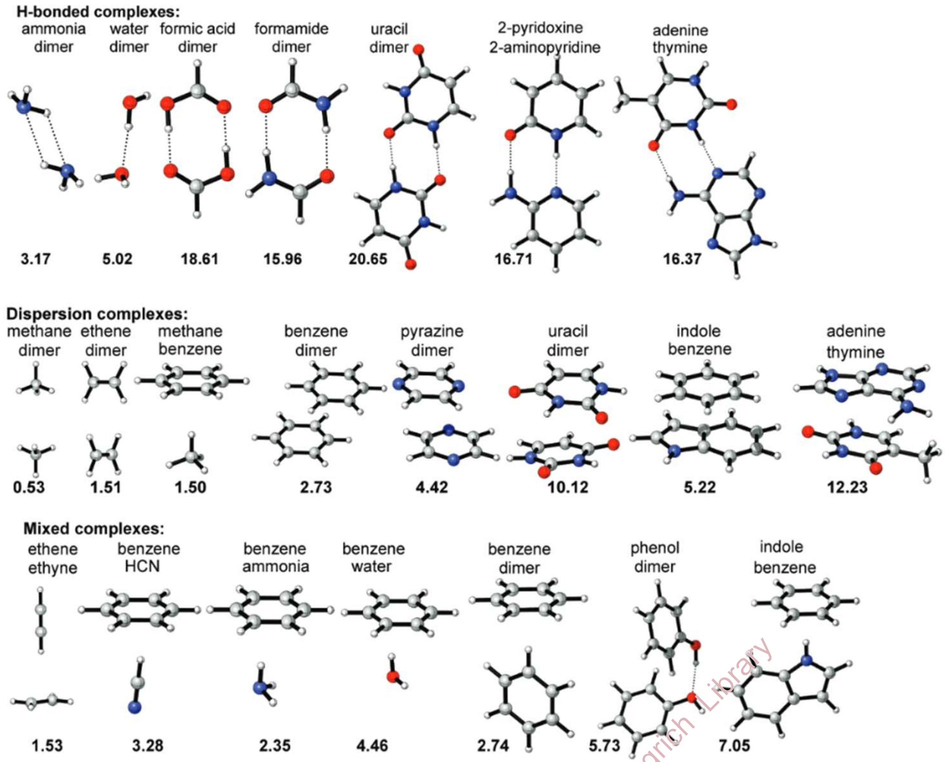


Figure 2 The S22 benchmark database complexes (*18*)

Řezáč, Riley, and Hobza introduced the S66 database as a balanced expansion to the S22 set, focusing on NCIs in bioorganic molecules. A special focus was the incorporation of aromatic-aliphatic and aliphatic-aliphatic dispersion interactions and single hyrdogen bonds (*19*). Several radial curves have been developed for these sets. The S22x5 radial curve set (*20*) expands the S22 set to add one shortened (0.9*r*) and three elongated (1.2*r*, 1.5*r*, 2.0*r*) geometries. The S66x8 radial curve set (*19*) was introduced at the same time as the S66 set, and incorporates two shortened (0.90*r*, 0.95*r*) and five elongated (1.05*r*, 1.10*r*, 1.25*r*, 1.50*r*, 2.00*r*) geometries. The S22x7 and S66x10 extensions (*21*) added two additional shortened geometries (0.7*r*, 0.8*r*) to the radial curves of S22x5 and S66x8, respectively, in order to balance the number of shortened and elongated geometries.

X40

The X40 dataset (Figure 3) contains 40 non-covalent complexes involving halogens participating in a variety of interaction types (*23*). The complexes are grouped into several interaction types: London Dispersion, induction, dipole-dipole interaction, π-stacking, halogen-π interactions, halogen bonds, and hydrogen bonds. This dataset is produced by the same group that designed the S66 dataset and is constructed analogously to that set such that the two sets are fully compatible. The X40x10 radial curve set is composed of four shortened (0.80*r*, 0.85*r*, 0.90*r*, 0.95*r*) and five elongated (1.05*r*, 1.10*r*, 1.25*r*, 1.50*r*, 2.00*r*) geometries. The X40 includes the halogens F, Cl, Br, and I; however, iodine cannot be calculated using small basis sets, so an analogous set excluding all iodine compounds, X31, can also be constructed (*21*).

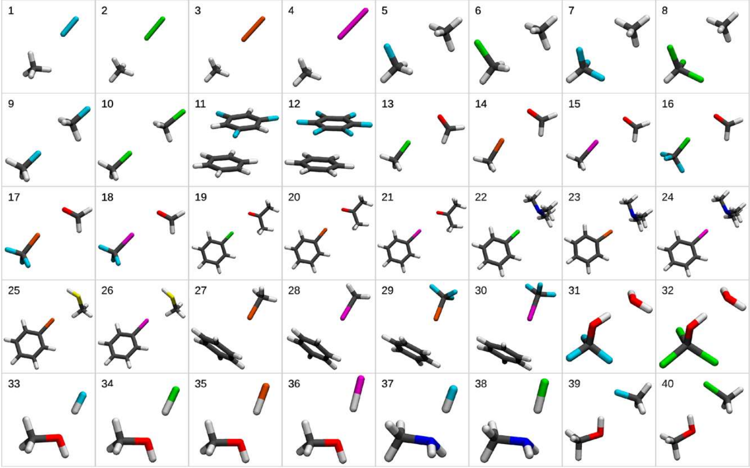


Figure 3 The X40 benchmark database complexes (*23*)

L7

Many of the benchmark databases available focus on the performance of DFT functionals and dispersion corrections for small and medium systems, few datasets exist for evaluating the same performance for large systems. The L7 database consists of seven large systems ranging from 48 to 112 atoms, representative of major dispersion-dominated biochemical motifs (*24*). The importance of this database is clear for biochemical systems, where noncovalent interactions accumulate for larger systems and impact macromolecular structure.

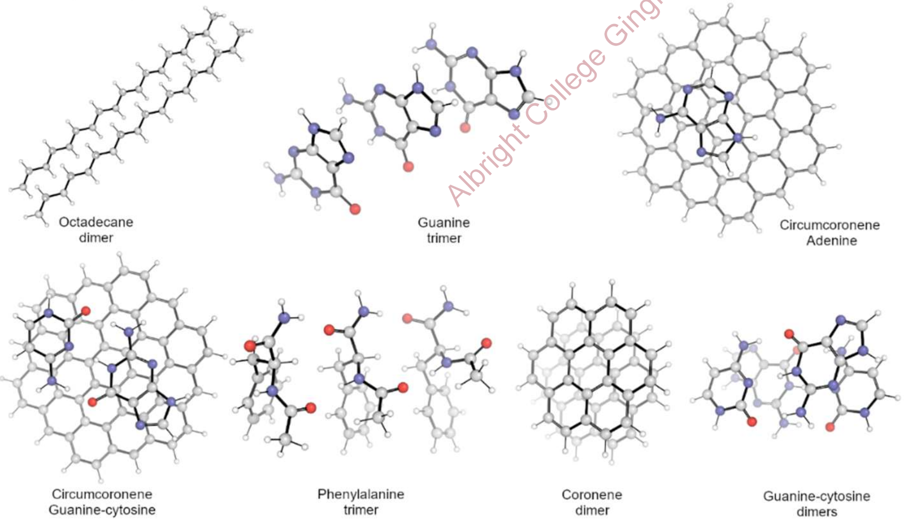


Figure 4 The L7 database complexes

NCIBLIND10

The NCIBLIND10 dataset contains radial curves for 10 dimers ranging in size from water (6 atoms) to ethylenedinitramine (EDNA, 32 atoms) (*22*). While most benchmark sets are mainly composed as a mix of dispersion-dominated or hydrogen-bond-dominated dimers, the compounds in this set were chosen to prove a spectrum ranging between these two extremes, with the dipole moment strength roughly corresponding to the difficulty of the system. The NCIBLIND10 radial curves are not of uniform size and are generated by specifically varying the interaction/bond length, instead of scaling the distance between the monomers as in the radial curves in S66x8.

In order to successfully parameterize dispersion corrections, it is desirable to have as representative of a benchmarking set as possible. To that end, multiple efforts have been focused on developing benchmarking supersets composed of many smaller datasets, in an effort to form a comprehensive database appropriate for many applications. While the sweeping scale of these databases does allow for highly representative training and testing sets in the parameterization process, the unwieldy amount of data encompassed in these supersets drastically increases the computational cost of performing an optimization. Compromises between the analytical power of large subsets and the computational efficiency of smaller sets have been found through various efforts to create representative subsets that preserve the behavior of the supersets, but with a reduced number of systems.

GMTKN and Diet GMTKN55

Goerigk and Grimme proposed a definitive database for general main group thermochemistry, kinetics, and non-covalent interactions (GMTKN24), the first major example of a superset database, intended for benchmarking DFT functionals (*25*). This super set is composed of 731 data points from 24 different benchmark datasets, including four new sets proposed in that work. Benchmarking in GMTKN is aided by the use of weighted total mean absolute deviation (WTMAD), specifically the weighting scheme WTMAD-2, which scales the subset error depending on the magnitude of the energies in that subset. The original database was expanded to GMTKN30 the following year, adding six new datasets and modifying three from the existing superset, increasing the total number of data points to 841 (*26*). The newest superset in this lineage is GMTKN55, a complete overhaul of the previous database (*27*). Many of the subsets were updated to include new reference values or incorporate extensions to the original and only three subsets were left unchanged. The total number of data points has expanded to 1505, more than double the number of original datapoints. 2462 single-point calculations are required to obtain benchmarks against the entire GMTKN55 database, presenting a significant computational expense for using this superset to parameterize dispersion corrections.

The 'Diet GMTKN' subsets proposed by Tim Gould, chosen using a stochastic genetic approach, are meant to reproduce the ranking of functionals by the full GMTKN55 superset using a fraction of the computational power (*28*). These subsets, containing 30, 100, and 150 systems from the full superset, reproduce the quantitative errors found through full-set benchmarking and are increasingly a representative sample of the full set: in the 100-system subset, 75% of the sets comprising the superset are represented. While these subsets are highly suitable for pre-screening operations, as well as dispersion correction parameterization, they are not as powerful as the full GMTKN55 database and are not meant for deep analysis.

Sherrill’s D3 Database

In 2016, Smith et al. proposed revised damping parameters for the D3 dispersion correction (*21*); these parameters are the ones most commonly used today. Their training (1526 systems) and validation (6773 systems) sets were developed with the recognition that fully focusing on equilibrium-geometry accuracy does not ensure accuracy across the entire potential energy system. To this end, both the training and validation sets are composed of radial curve sets, several of which (S22x7 and S66x10, as already presented examples) were extended to include more shortened geometries for this work. The training set, larger than the full GMTKN55 database, provided a powerful tool for reparameterizing the D3 dispersion correction, and the expansive validation set allows for extensive analysis of the accuracy of any computational method in describing dispersion.

MGCDB82 and MG8

Mardirossian and Head-Gordon proposed the MGCDB84 superset in their extensive review of density functional theory (*29*). The MGCDB84 set is composed of databases of NCI, isomerization energy, thermochemistry, and barrier height benchmarks, and consists of 4986 total systems. The NCI benchmarks alone comprise 2078 systems, comparable to the size of the entire GMTKN55 superset. While the deep analysis of 200 density functionals that was facilitated by the size of MGCDB84, the extreme computational complexity of this set lends itself well to a subsetting approach similar to that used to develop the ‘Diet GMTKN’ set. Bun Chan recently generated such a subset, MG8, containing just 60 systems from the total superset (*30*). The systems comprising MG8 were determined using lasso regression and forward selection to produce the subsets in each category that were most highly correlated with the overall mean absolute deviation (MAD) the superset. Next, systems were trimmed from each set until a small-but-representative subset of systems was produced. The MG8 set is intended as a qualitative measure of performance: MADs below the ~4 kJ/mol (~1 kcal/mol) threshold are likely to indicate a robust computational method.

Experimental

CCBDB is a program designed to automate the benchmarking of computational methods and optimize the parameters of the D3 dispersion correction. The program is fully implemented in Python, designed to be fully compatible with open data principles and accessible to chemists with a broad range of coding abilities. The dispersion correction code easily allows for the addition of new damping functions, and the program interface is being developed to be increasingly program-agnostic, to allow for broader use as a computational chemistry tool. Figure 5 shows the full automated workflow of the program. After specifying the desired databases to be benchmarked for a specific computational method, no further input is required throughout the entire benchmarking workflow, making CCBDB invaluable as a benchmarking tool.

The original framework of CCBDB allowed for a computational method to be compared to a single database, as a proof of concept of the automation process within the program. The program was able to successfully generate Gaussian input files, submit Gaussian jobs to a server-based implementation, import the calculation results, and process the results for presentation in the program readout. After this system was fully functional, the next step was to develop a method of applying Grimme’s D3 correction to the results. Rob Paton and Kelvin Jackson’s Python implementation of D3 (pyDFTD3) was a suitable starting point, but only allowed for calculation of a single calculation per function call (*31*). The original program was modified to allow for high-throughput calculation by developing an interface that allowed direct access via CCBDB. The D3 dispersion correction was implemented to support both zero-damping and BJ-damping methods; other damping functions could be added to the program with less than fifty added lines of code.

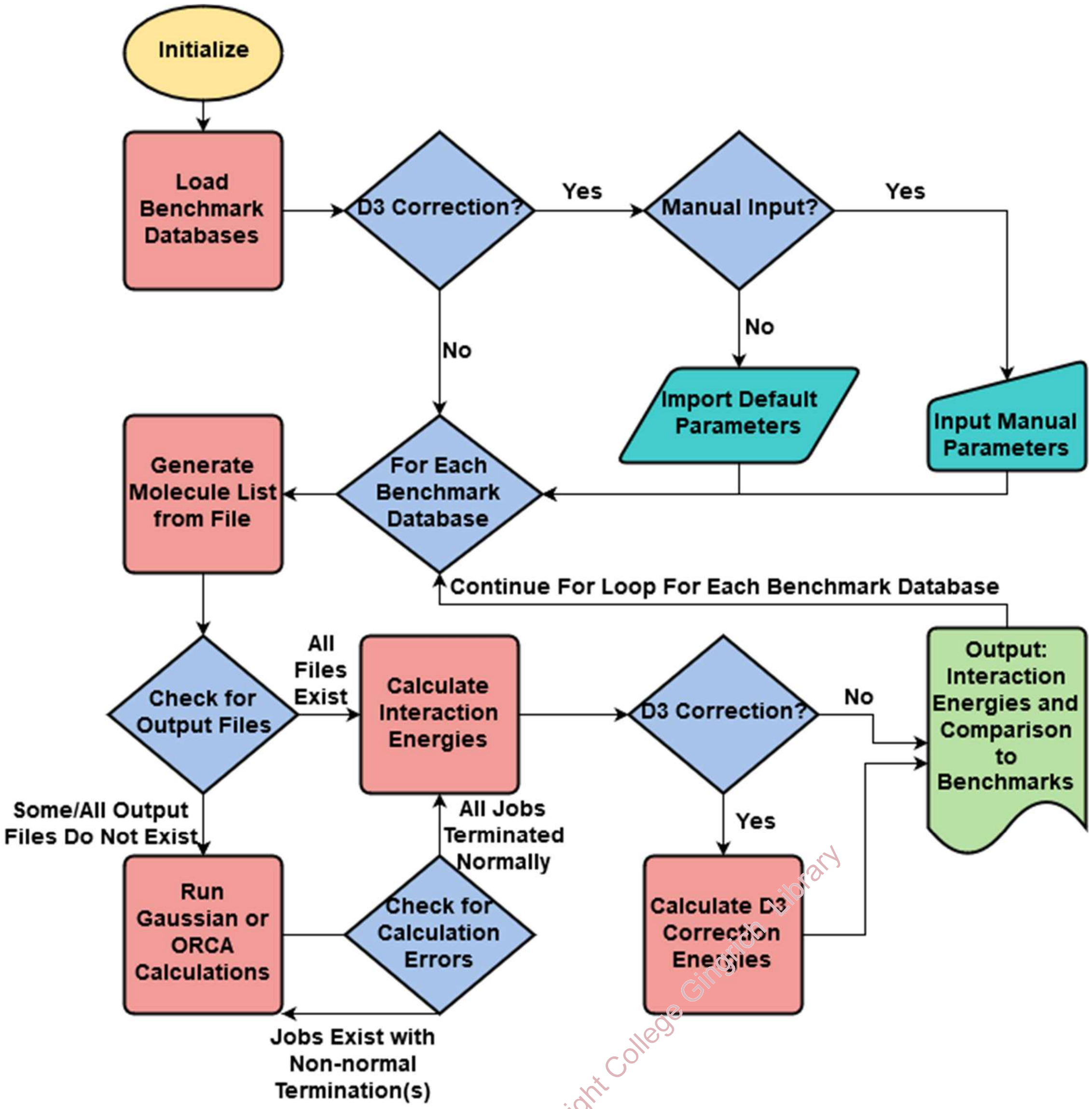


Figure 5 Schematic of the CCBDB automated workflow

After enabling dispersion corrections, the next outstanding issue was the one-database limit of the program. Large amounts of body code were generalized to allow for multi-database calculations, and a suite of single-database and multi-database descriptive statistics were introduced into the program readout. After this recoding, the only limit on the possible number of databases used in CCBDB is the number implemented in the program. In total, 10 databases are currently supported: S22, S22x7, S66x10, X31, X31x10, X40, NCIBLIND10, MG8, L7, and CHBDE (a small set of experimental bond dissociation energies that is useful for testing). The main bottleneck in adding additional databases is the highly heterogeneous nature of data available from different groups, which hinders the ability to uniformly process the data into the necessary form to be used in CCBDB. Parallel job submission was introduced in order to speed up the data generation process for systems with sufficient capacity, and an interface with ORCA was developed, to improve the versatility of the program. The input/output files will be converted to JSON format in an upcoming update, in order to reduce the overall data storage size requirements and make CCBDB more program-agnostic, which should easily allow

The next major feature introduced in CCBDB was the ability to optimize dispersion correction parameters for a specific computational method. The option of inputting manual parameters was introduced during the initial implementation of the dispersion correction but was not intended for use in optimization contexts. A Nelder-Mead simplex optimization was selected for the optimization in order to avoid calculation of the gradient, however, it may be necessary to incorporate a gradient-based method to speed up convergence. Currently, CCBDB begins optimization at the literature parameters if they exist for the functional being used or at zero for functionals without existing parameters. A planned improvement to the optimization method is to apply a hyper-parameter optimization procedure to determine a better starting point, in order to encourage quicker optimization convergence (*32*).

CCBDB Usage

Initiating CCBDB requires a single function call to begin the automated workflow. In this function call, the method (functional and basis set), databases, and dispersion correction options are specified. In this section, the input and output of two CCBDB function calls are described to demonstrate the functionality of the program.

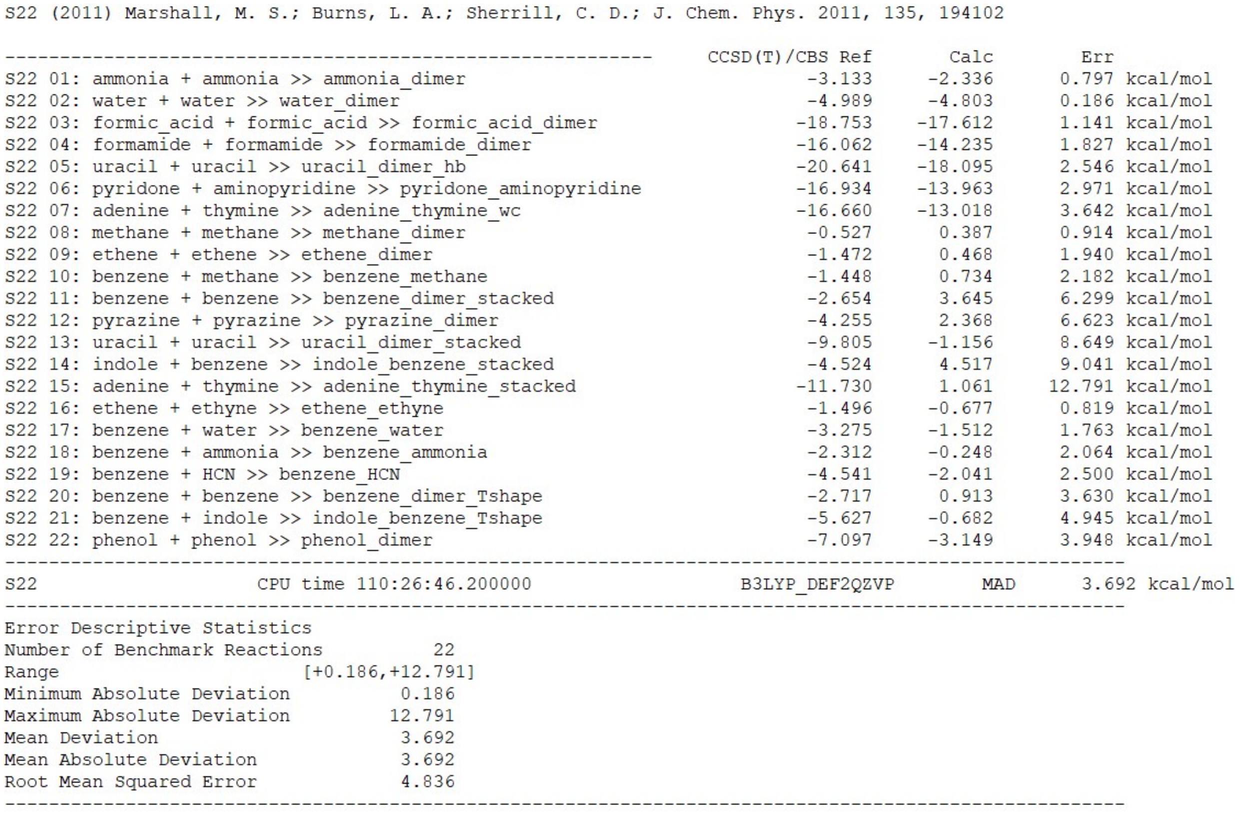
Basic Benchmarking

Initiating the program for basic benchmarking requires nothing more than specifying a method and desired datasets (Figure 6).

```
CCBDB.py -method B3LYP/def2qzvp -db S22
```

Figure 6 CCBDB input for benchmarking B3LPY/def2-QZVP for the S22 database

The program will then attempt to load the energy calculations from the system files. If the calculations do not yet exist, the program will generate input files for the chosen quantum chemistry program (the current default is Gaussian) and submit each input file to that program; on super-computing clusters, it is possible to submit several jobs in parallel through CCBDB. The final output (Figure 7) shows the reference values, calculated energies, and a number of descriptive statistics.



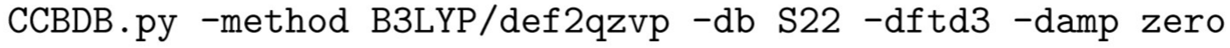
```
S22 (2011) Marshall, M. S.; Burns, L. A.; Sherrill, C. D.; J. Chem. Phys. 2011, 135, 194102

-----------------------------------------------------------    CCSD(T)/CBS Ref       Calc        Err
S22 01: ammonia + ammonia >> ammonia_dimer                               -3.133     -2.336      0.797 kcal/mol
S22 02: water + water >> water_dimer                                     -4.989     -4.803      0.186 kcal/mol
S22 03: formic_acid + formic_acid >> formic_acid_dimer                  -18.753    -17.612      1.141 kcal/mol
S22 04: formamide + formamide >> formamide_dimer                        -16.062    -14.235      1.827 kcal/mol
S22 05: uracil + uracil >> uracil_dimer_hb                              -20.641    -18.095      2.546 kcal/mol
S22 06: pyridone + aminopyridine >> pyridone_aminopyridine              -16.934    -13.963      2.971 kcal/mol
S22 07: adenine + thymine >> adenine_thymine_wc                         -16.660    -13.018      3.642 kcal/mol
S22 08: methane + methane >> methane_dimer                               -0.527      0.387      0.914 kcal/mol
S22 09: ethene + ethene >> ethene_dimer                                  -1.472      0.468      1.940 kcal/mol
S22 10: benzene + methane >> benzene_methane                             -1.448      0.734      2.182 kcal/mol
S22 11: benzene + benzene >> benzene_dimer_stacked                       -2.654      3.645      6.299 kcal/mol
S22 12: pyrazine + pyrazine >> pyrazine_dimer                            -4.255      2.368      6.623 kcal/mol
S22 13: uracil + uracil >> uracil_dimer_stacked                          -9.805     -1.156      8.649 kcal/mol
S22 14: indole + benzene >> indole_benzene_stacked                       -4.524      4.517      9.041 kcal/mol
S22 15: adenine + thymine >> adenine_thymine_stacked                    -11.730      1.061     12.791 kcal/mol
S22 16: ethene + ethyne >> ethene_ethyne                                 -1.496     -0.677      0.819 kcal/mol
S22 17: benzene + water >> benzene_water                                 -3.275     -1.512      1.763 kcal/mol
S22 18: benzene + ammonia >> benzene_ammonia                             -2.312     -0.248      2.064 kcal/mol
S22 19: benzene + HCN >> benzene_HCN                                     -4.541     -2.041      2.500 kcal/mol
S22 20: benzene + benzene >> benzene_dimer_Tshape                        -2.717      0.913      3.630 kcal/mol
S22 21: benzene + indole >> indole_benzene_Tshape                        -5.627     -0.682      4.945 kcal/mol
S22 22: phenol + phenol >> phenol_dimer                                  -7.097     -3.149      3.948 kcal/mol
-----------------------------------------------------------------------------------------------------------
S22                  CPU time 110:26:46.200000              B3LYP_DEF2QZVP        MAD      3.692 kcal/mol
-----------------------------------------------------------------------------------------------------------
Error Descriptive Statistics
Number of Benchmark Reactions          22
Range                            [+0.186,+12.791]
Minimum Absolute Deviation          0.186
Maximum Absolute Deviation         12.791
Mean Deviation                      3.692
Mean Absolute Deviation             3.692
Root Mean Squared Error             4.836
-----------------------------------------------------------------------------------------------------------
```

Figure 7 Output file for benchmarking B3LYP/def2-QZVP with the S22 database

Benchmarking with Dispersion Corrections

Initiating the program for benchmarking with dispersion correction requires the same specifications as the basic benchmarking, in addition to selecting a damping function (Figure 8).



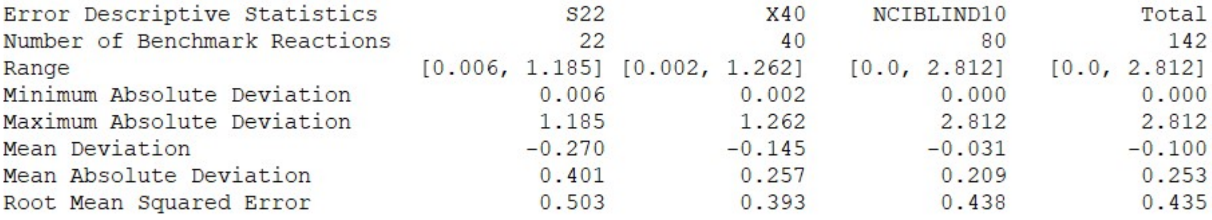
```
CCBDB.py -method B3LYP/def2qzvp -db S22 -dftd3 -damp zero
```

Figure 8 CCBDB input for benchmarking B3LYP-D3(0)/def2-QZVP with the S22 database

The program will follow the same calculation steps as before, however, the D3 corrections are recalculated each time. The final output shows similar information to the basic output, but with the corrected energies and the D3 parameters used.

Benchmarking with Multiple Databases

CCBDB allows for benchmarking with several databases, using similar syntax to the basic function calls, but adding additional databases (comma-separated) to the -db specifier. The program follows the same calculation steps as with one database, but the final output includes descriptive statistics for each database as well as for the combined sets (Figure 9).



```
Error Descriptive Statistics                S22             X40       NCIBLIND10           Total
Number of Benchmark Reactions                22              40               80             142
Range                            [0.006, 1.185] [0.002, 1.262]   [0.0, 2.812]     [0.0, 2.812]
Minimum Absolute Deviation                0.006           0.002            0.000           0.000
Maximum Absolute Deviation                1.185           1.262            2.812           2.812
Mean Deviation                           -0.270          -0.145           -0.031          -0.100
Mean Absolute Deviation                   0.401           0.257            0.209           0.253
Root Mean Squared Error                   0.503           0.393            0.438           0.435
```

Figure 9 Descriptive statistics output for multi-database benchmarking

Discussion

CCBDB is a versatile program with many functions relevant to either specialists, nonspecialists, or both. The implementation of sets such as MG8 allows for quick benchmarking that is highly suitable for groups developing new functionals. In particular, the transition to JSON formatting for input/output files should allow for fully quantum-chemistry-program-agnostic performance, very suitable for functionals not implemented in a popular program such as Gaussian. CCBDB is fully compatible with large supersets such as the GMTKN or MGCDB databases, provided that the required subsets have already been implemented in the program. In fact, once the CCBDB database has been sufficiently expanded, it should be possible to develop and test new supersets, using either full databases or components from them, in addition to testing new datasets of any type.

A main focus of this program is the D3 dispersion correction. This does not preclude the choice of using a different dispersion correction, however, this modification would require a significant effort. At this point in time, the D3 correction is considered to be the top-of-the-line empirical correction of its type and should continue to be for some time. The incorporation of new damping functions (such as the OP or CSO functions outlined earlier) is a much easier undertaking, and is one of high relevance, due to some initial findings that these functions perform as well as the currently implemented functions. Thus, CCBDB is an excellent option for groups developing new damping functions and would allow for speedy comparison to benchmark data.

Dispersion corrections are generally parameterized for use with the def2-QZVP basis set, so chosen because its large size limits issues such as the basis set superposition error (BSSE). In practice, however, this basis set is prohibitively large for use with larger systems and would require months of computing time. Using a slightly smaller basis set in conjunction with D3's literature parameters increases the error of the system but may be preferable in some situations where the computational cost would otherwise be much too high. A compromise is proposed by Hostaš and Řezáč, wherein a smaller basis set, DZVP-DFT, with a small BSSE for its size, is used to reparameterize D3 for small-basis-set applications (*33*). Their work produced new parameters for D3, using all four common damping functions, for five common functionals, and yielded some promising results for decreasing error when used with small basis sets. CCBDB provides an excellent tool for producing parameters for any other functional using similar methodology.

The same automation that makes CCBDB highly suitable for specialists allows for it to be successful in broader use as well. CCBDB is fully implemented in Python, one of the fastest growing and most popular programming languages worldwide. This makes the program highly accessible to chemists of all stripes in a way that code implemented in languages such as Fortran cannot. Using CCBDB to benchmark a specific computational method requires only a single function call, and the data is presented in a highly organized manner, allowing for quick analysis.

CCBDB is designed with open data in mind. The Python implementation of this code allows for broad collaboration once the program is introduced, with each improvement increasing the impact of this program. After calculations are run, the output data is stored, to prevent reduntant calculations. To this end, all of those calculations will be made fully available online, such that nonspecialists who hope to benchmark several common functionals and/or basis sets likely will not even need to perform their own calculations, and can use the stored data, further reducing the computational demands for using CCBDB.

Conclusion

Density functional theory is an increasingly popular topic for theoreticians and experimentalists alike. While DFT methods are successful in their treatment of many chemical phenomena, one particular failure is in describing noncovalent interactions. The use of dispersion corrections such as the D3 correction allow for much more accurate descriptions of the energy of these systems. For this purpose, a variety of benchmark databases exist to test the accuracy of computational methods. CCBDB is a Python program designed to automate the benchmarking process and optimize D3 parameters for any DFT functional. This program is relevant to both specialists developing functionals and nonspecialists looking to find a suitable functional for a specific application. The open-data-focused design of the program will allow for broad collaboration in designing a highly valuable computational chemistry tool.

References

1. Hohenburg, P.; Kohn, W. *Physical Review B* **1964,** *136* (3B), B864.
2. Jensen, F. *Introduction to Computational Chemistry;* John Wiley & Sons, Ltd.: Chichester, 2017.
3. Kryachko, E. S.; Ludeña, E. V. *Physics Reports - Review Section of Physics Letters* **2014,** *544* (2), 123-239.
4. Kohn, W.; Sham, L. J. *Phys. Rev.* **1965,** *140* (4A), A1133.
5. Lee, C. T.; Yang, W. T.; Parr, R. G. *Phys. Rev. B* **1986,** *37* (2), 785-789.
6. Perdew, J. P.; Burke, K.; Ernzerhof, M. *Phys. Rev. Lett.* **1996,** *77* (18), 3865-3868.
7. Becke, A. D. *J. Chem. Phys.* **1993,** *98* (7), 5648-5652.
8. Sousa, S. F.; Fernandes, P. A.; Ramos, M. J. *J. Phys. Chem. A* **2007,** *111* (42), 10439-10452.
9. Chai, J.-D.; Head-Gordon, M. *Phys. Chem. Chem. Phys.* **2008,** *10* (44), 6615-6620.
10. Burns, L. A.; Vázquez-Mayagoitia, A.; Sumpter, B. G.; Sherrill, C. D. *J. Chem. Phys* **2011,** *134,* 084107.
11. Riley, K. E.; Pitoňák, M.; Jurečka, P.; Hobza, P. *Chem. Rev.* **2010,** *110,* 5023-5063.
12. Grimme, S.; J., A.; Ehrlich, S.; Krieg, H. *J. Chem. Phys.* **2010,** *132* (15), 154104.
13. Becke, A. D.; Johnson, E. R. *J. Chem. Phys.* **2007,** *127* (15), 154108.
14. Grimme, S.; Ehrlich, S.; Goerigk, L. *J. Comput. Chem* **2011,** *32,* 1456-1465.
15. Witte, J.; Mardirossian, N.; Neaton, J. B.; Head-Gordon, M. *J. Chem. Theory Comput.* **2017,** *13,* 2043-2052.
16. Schröder, H.; Creon, A.; Schwabe, T. *J. Chem. Theory Comput.* **2015,** *11,* 3163-3170.
17. Jurečka, P.; Špooner, J.; Černý, J.; Hobza, P. *Phys. Chem. Chem. Phys.* **2006,** *8,* 1985-1993.
18. Paton, R. S.; Goodman, J. M. *J. Chem. Inf. Model.* **2009,** *49,* 944-955.
19. Řezáč, J.; Riley, K. E.; Hobza, P. *J. Chem. Theory Comput.* **2011,** *7,* 2427-2438.
20. Gráfová, L.; Pitoňák, M.; Řezáč, J.; Hobza, P. *J. Chem. Theory Comput.* **2010,** *6* (8), 2365-2376.

21. Smith, D. G. A.; Burns, L. A.; Patkowski, K.; Sherrill, C. D. *J. Phys. Chem. Lett.* **2016,** *7,* 2197-2203.
22. Taylor, D. E.; Ángyán, J. G.; Galli, G.; Zhang, C.; Gygi, F.; Hirao, K.; Song, J. W.; Rahul, K.; von Lilienfeld, O. A.; Podeszwa, R.; Bulik, I. W.; Henderson, T. M.; Scuseria, G. E.; Toulouse, J.; Peverati, R.; Truhlar, D. G.; Szalewicz, K. *J. Chem. Phys.* **2016,** *145,* 124105.
23. Řezáč, J.; Riley, K. E.; P., H. *J. Chem. Theory Comput.* **2012,** *8,* 4285-4292.
24. Sedlak, R.; Janowski, T.; Pitoňák, M.; Řezáč, J.; Pulay, P.; Hobza, P. *J. Chem. Theory Comput.* **2013,** *9* (8), 3364-3374.
25. Goerigk, L.; Grimme, S. *J. Chem. Theory Comput.* **2010,** *6,* 107-126.
26. Goerigk, L.; Grimme, S. *J. Chem. Theory Comput.* **2011,** *7,* 291-309.
27. Goerigk, L.; Hansen, A.; Bauer, C.; Ehrlich, S.; Najibi, A.; Grimme, S. *Phys. Chem. Chem. Phys.* **2017,** *19,* 32184.
28. Gould, T. *Phys. Chem. Chem. Phys.* **2018,** Advance Article.
29. Mardirossian, N.; Head-Gordon, M. *Molecular Physics* **2017,** *115* (19), 2315-2372.
30. Chan, B. *J. Chem. Theory Comput.* **2018,** *14* (8), 4254-4262.
31. Paton, R.; Jackson, K. pyDFTD3.py. https://github.com/bobbypaton/pyDFTD3.
32. Jones, D.; Schonlau, M.; Welch, W. *Journal of Global Optimization* **1998,** *13* (4), 455-492.
33. Hostaš, J.; Řezáč, J. *J. Chem. Theory Comput.* **2017,** *13* (8), 3575-3585.