Working with benchmark datasets in the Cuby framework

Jan Řezáč,* Outi Vilhelmiina Kontkanen, and Martin Nováček

Institute of Organic Chemistry and Biochemistry, Czech Academy of Sciences, 160 00 Prague, Czech Republic

***e-mail: rezac@uochb.cas.cz**

February 13, 2024

Abstract

The development and benchmarking of computational chemistry methods relies on comparison with benchmark data. More and larger benchmark datasets are becoming available, and working efficiently with them is a necessity. The Cuby framework provides rich functionality for working with datasets, comes with many ready-to-use predefined benchmark sets, and interfaces with a wide range of computational chemistry software. Here we review the tools Cuby provides for working with datasets and provide examples of more advanced workflows, such as handling large numbers of computations on HPC resources and reusing previously computed data. Cuby has also been extended recently to include two important benchmark databases, NCIAtlas and GMTKN55.

Introduction

Cuby is a software framework that provides unified access to a wide range of computational chemistry methods implemented in different software packages. It can be used to build com-

plex computational protocols that combine multiple individual computations and to simplify the execution of computations in high performance computing (HPC) environments. Cuby is written in the Ruby programming language,[1] but its use does not require programming skills; even the most complex tasks can be defined using structured input files. Cuby is open source software, freely available at `http://cuby4.molecular.cz`, where you can also find documentation and extensive collections of sample input files. The design of the framework and a broader overview of its possible applications have been described in Ref. 2, and the most important information about the software's architecture, capabilities, and usage is summarized in the next section.

In this paper, we focus on a specific use of Cuby in benchmarking computational methods on predefined datasets. This was one of the main applications for which Cuby was developed, and it offers very useful features that simplify all steps of the process. The use of established benchmark datasets has become the norm in both the development and validation of approximate computational methods, improving the transparency and reproducibility of the studies. As the field continues to evolve, both the number and size of available datasets increase, and it is clear that a high degree of automation is required to work efficiently with them. For example, developments of new density functional theory (DFT) functionals involve benchmarking of tens of computational methods, often implemented in different software packages, applied to tens of datasets, resulting in hundreds of thousands of individual calculations.[3–6] Automated benchmarking is also useful on a smaller scale, allowing more application-oriented computational chemists to validate their methods on smaller subsets of data covering their topic of interest.

The Cuby framework is modular and provides all the building blocks needed to create a complete workflow for handling benchmark datasets. These modules fall into two categories: 'Protocols' that control the calculation workflow and 'Interfaces' that perform the individual elementary calculations, usually by calling external software. Every protocol can be coupled to any interface that implements the calculation method of choice. There are multiple

protocols available; this paper focuses mainly on the 'dataset' protocol, which automates the calculation of a predefined dataset and processes the results. The dataset itself is defined in a 'dataset definition file', and Cuby includes a large collection of predefined, ready-to-use datasets. Performing a calculation on the dataset is as simple as selecting the dataset and specifying the computational method to be applied to it. More advanced features can be used to manipulate the data, distribute calculations across a cluster, and reuse previously computed results. This is described in the corresponding sections of the paper.

Cuby comes with more than 80 predefined datasets, including the recently added Non-Covalent Interactions Atlas (NCIAtlas) datasets[7–11] and the GMTKN55 database,[12] which will be discussed later in this paper. The collection also includes widely used benchmarks for non-covalent interactions previously developed in our group, such as the S66 and X40 sets,[13–15] as well as datasets from other authors covering various other quantities collected from the literature. Some of the datasets contain additional data and metadata that allow a more detailed analysis of the results. Adding new datasets is also straightforward, and with Cuby's open source license, contributions of other high quality data are welcome. However, it should be noted that Cuby is not designed to work with very large databases (with more than thousands of entries), such as those used in the development of machine learning models.

The Cuby framework

The Cuby framework consists of several code layers. First, there are libraries for the object-oriented representation of the chemical data it works with (molecular geometries, calculated properties, etc.) and functions for working with them. Then, the core of the framework manages the setup and execution of the calculations. The last layer exposed to the user are the interchangeable modules that define what to calculate (the 'protocol' modules) and how to calculate it (the 'interface' modules). The Cuby calculation is defined by a structured input file based on the YAML format,[16] which is easy to read and write, but allows to express

complex structures if needed. A simple example of an input file for calculation of interaction energy with counterpoise correction of the basis set superposition error is shown in Listing 1. The '`interaction`' protocol automatically identifies the two molecules in the geometry, prepares the calculations with the additional basis functions, runs them and computes the interaction energy from their outputs. Both the specification of the molecules in the dimer, and many other options, can be configured from the input if the user wishes to change the defaults provided by Cuby. The implementation of MP2 in the Psi4 package[17] is used, but the same calculation can be carried out in any other software that implements it and is interfaced to Cuby just by changing the '`interface`' keyword. This input, as well as all other examples listed here, assume that all used interfaces are configured in Cuby's global configuration file which tells Cuby where to find the external software and how to run it (the details are described in the documentation[18]).

Listing 1: *Cuby input example, MP2/aug-cc-pVTZ interaction energy calculation with automatic detection of the two molecules in the geometry, and with counterpoise correction of basis set superposition error. In such a simple case, the input is just a list of keywords and values, with comments optional starting with the '#' character.*

```
# Interaction energy calculation, the protocol automatically
# recognizes the two molecules in the geometry
job: interaction
bsse_corretion: on
interface: psi4
method: MP2
basisset: aug-cc-pVTZ
geometry: water_dimer.xyz
charge: 0
```

More details on how Cuby handles the calculations internally can be found in the previous paper on Cuby,[2] and more examples can be found in the documentation online.[18] Cuby currently provides interfaces to the following external software: AmberTools,[19] CFOUR,[20] CHARMM,[21] deMon,[22] DFTB+,[23] GAMESS,[24] Gaussian,[25] Molpro,[26] Mopac,[27] MRCC,[28]

Orca,[29] Psi4,[17] Turbomole[30] and Xtb.[31] The interfaces may not cover the full functionality of each software package, but adding more features to an existing interface usually requires only small modifications to the code. An up-to-date list, including the interfaces to calculations implemented directly in Cuby, is available on the corresponding page of the documentation.[32]

However, the main power of Cuby is not in simplifying already simple tasks, but in the possibility to combine and layer the elementary protocols into more complex but fully automated workflows. For example, Cuby can iterate over a dataset using the '`dataset`' protocol but instead of a single-point calculation on its each entry, it is possible to perform multiple steps using different protocols and methods implemented in different external software before the results are handed back to the dataset protocol for processing and analysis.

Working with datasets in Cuby

The Dataset protocol. This protocol performs calculations on a dataset and analyzes the results. The dataset is specified in a separate file what makes it easy to reuse the datasets. Each dataset definition file also contains the setup of the calculation applied to each item in the dataset. For example, the interaction energy datasets enforce the use of the '`interaction`' protocol, and the database of geometries used in the dataset may contain the definition of the molecular fragments needed for such calculations. For the default calculation on the dataset, as intended by the authors of the data, only the calculation method needs to be specified, and no additional input on the specific protocol applied to the dataset entries is required. As a result, a calculation on a predefined dataset requires only very simple input, as demonstrated in the Listing 2.

Listing 2: *Example of a Cuby input file performing calculation of the S66 dataset using the PM6 semiempirical method*[33] *in Mopac.*[27]

```
# Calculation of the S66 dataset packaged with Cuby
job: dataset
dataset: S66
interface: mopac
method: PM6
```

Cuby not only performs the calculations, but also processes the results. On the output, Cuby prints a table of the computed results, along with the reference values and the error relative to the reference, as well as the overall error measures for the entire dataset. The most important parts of the output of the example calculation described above are shown in the Listing 3.

Listing 3: *Key parts of the output of the dataset calculation invoked by the input shown in Listing 2.*

```
================================================================================================
name                                                E      Eref     error   error(%)
------------------------------------------------------------------------------------------------
01 Water ... Water                               -3.912    -5.011     1.099    21.938
02 Water ... MeOH                                -4.236    -5.701     1.465    25.698
03 Water ... MeNH2                               -4.055    -7.036     2.981    42.365
04 Water ... Peptide                             -6.281    -8.220     1.939    23.586
...

================================================================================================
RMSE                3.069   kcal/mol
MUE                 2.702   kcal/mol
MSE                 2.702   kcal/mol
...

================================================================================================
RMSE               60.365   %
MUE                57.000   %
...
```

Built-in datasets. The Cuby package includes more than 80 predefined datasets. They are listed in the documentation at page `http://cuby4.molecular.cz/datasets.html`. They include the NCIAtlas and GMTKN55 databases discussed below, as well as a diverse collection of standalone datasets, covering mostly non-covalent interactions. These

datasets are listed in Table 1. Calculations on these datasets can be invoked by a simple input analogous to the example provided in the Listing 2.

Table 1: Datasets included in Cuby installation, except the NCIAtlas and GMTKN55 databases which are discussed separately later in the paper.

Quantity	Datasets
Interaction energies	3B69, A24, Bauza2013, Charge_transfer, HB104, Ionic_H-bonds, L7, Pecina2015, PLFrag547, R160x6, S12L, S66, S66a8, S66x8, Sulfur_x8, X40, X40x10
Three-body interactions	3B69_dimers
Conformation energies	MPCONF196, Peptide_FGG, Peptide_GFA, Peptide_GGF, Peptide_WG, Peptide_WGG
Atomization energies	W4-17
Dipole moments	Dipoles152

Custom datasets. Creating a new dataset is not difficult either. The dataset definition file is a structured YAML file that describes the dataset and lists its elements. The best way to start is to use an existing dataset as a template (you can find them in the directory `cuby4/data/datasets`). The calculations can refer to molecular geometries in separate files, in the dataset file itself, or in the Cuby geometry database[34]. The calculation of the dataset is then invoked by passing the path to the dataset definition file to the '`dataset`' keyword.

Predefined groups and other metadata. The Cuby datasets can contain metadata that makes it easier to work with the datasets and analyze the results. The first feature is the groups defined within a dataset. When groups are defined, the statistics are not only evaluated for the entire dataset, but also for each group within it. Predefined datasets with author-defined groups make it easy to reproduce the original analysis. Additional, arbitrary metadata can be included in the 'tags' field of the dataset entries.

Subset selection. Often only a certain part of the dataset needs to be calculated. Cuby provides several keywords to select these subsets. The selections can be made using the names of the records, the groups, the tags, and the elemental composition of the system. The selections can be both positive (using the keywords '`dataset_select_...`') and negative (using '`dataset_skip_...`'). All options are listed in the table 2. These keywords can be

combined in order to create more complex selections. An example is provided in Listings 4.

Table 2: Cuby keywords used to select a subset of records from a data set.

Selection by	Keywords
entry name[a]	'`dataset_select_name`', '`dataset_skip_name`'
group[a]	'`dataset_select_group`', '`dataset_skip_group`'
tags[a]	'`dataset_select_tag`', '`dataset_skip_tag`'
elemental composition	'`dataset_select_elements`', with multiple possible selection modes possible: '`no_more_than(list)`', '`at_least_one(list)`' and '`at_least_all(list)`' where '`list`' is comma-separated list of element names

[a] supports regular expressions

Listing 4: *Example of selections applied to the Non-covalent Interactions Atlas SH250×10 dataset of σ-hole interactions. The comments in the input (starting with '#') describe the individual selections being made.*

```
job: dataset
dataset: NCIA_SH250x10
interface: mopac
method: PM6

# Selections:
# The dataset has groups by the element involved in
# the interactions, we select only halogens  using
# a regular expression operator "|"
dataset_select_group: "Cl|Br|I"

# The dataset contains dissociation curves, we select only
# the equilibrium  geometries marked by a tag "equilibrium"
dataset_select_tag: "equilibrium"

# Exclude systems with atoms of other elements
dataset_select_elements: "no_more_than(H,C,N,O,Cl,Br,I)"
```

Custom processing of the dataset entries

As mentioned above, the dataset definition file also contains the setup of the calculations to be applied to the entries of the dataset. For example, in a dataset of interaction energies, the '`interaction`' protocol is automatically applied, and additional setup may be present to ensure that the calculated quantity corresponds the benchmark data. The user provides only the setup of the method used in these calculations. However, there are cases where it may be useful to replace or modify this default setup.

This can be achieved by defining the '`calculation_overwrite`' block in the input file (the blocks are separate, named parts of the input that form a tree structure defined by their indentation). The contents of this block specify the calculation to be performed on the dataset entries instead of the default calculation. The results of these calculations are then processed and evaluated as usual. Obviously, it makes no sense to calculate a quantity for which the dataset is not intended, but the strength of this feature lies in the possibility of applying more complex calculation protocols to the dataset entries before calculating the final result. The modular structure of Cuby allows to define such a workflow from a single hierarchically structured input file.

An example we often use in practice is the preprocessing of the geometries of the systems before the final computation of the benchmarked quantity. This can be done using the '`multistep`' protocol, which allows chaining multiple calculations and returns the result of the last one. The input file provided in Listing 5 performs a geometry optimization of each of the systems (molecular dimers of the S66 set) and computes the interaction energy on the optimized geometry. Note that for the '`multistep`' protocol, we first define the steps to be applied and then define each of them in a separate block of the input. The setup shared by all steps, in this case the setup of the method, can only be provided once in the '`calculation_common`' block (although it is of course possible to use a different method in each step). This workflow implements a very useful test if the benchmarked method describes well not only the interaction energies (which are commonly tested only in fixed geometries),

but also the geometries themselves.

Listing 5: *Example of custom processing of the dataset items. The 'multistep' protocol is used to to first optimize and subsequently compute the interaction energy with the PM6-D3H4 semiempirical QM method in the S66 dataset.*

```
job: dataset
dataset: S66

calculation_overwrite:
  job: multistep
  steps: opt, int

  # Common setup for all calculations
  calculation_common:
    interface: mopac
    method: pm6
    mopac_corrections: d3h4

  # Optimize geometry of each item in dataset
  calculation_opt:
    job: optimize
    # The geometry is read from a parent block,
    # to which it is passed by the dataset protocol
    geometry: parent_block

  # Calculate interaction energy in the optimized geometry
  calculation_int:
    job: interaction
    geometry: optimized.xyz
```

Handling large datasets on HPC clusters

By default, calculations on dataset entries are performed sequentially, making calculations on larger datasets time-consuming. There are several ways to mitigate this and make more efficient use of parallel HPC computing resources.

First, the individual calculations can take advantage of the parallelization implemented in the external software used (using the '`parallel`' keyword). The execution of a dataset

computation then results in a series of sequentially executed parallel calculations, which is useful for more demanding computations on small datasets.

Second, Cuby implements an internal parallelization mechanism for running multiple calculations simultaneously (accessible using the '`cuby_threads`' keyword). This option is limited to parallelizing the computations within a single computer and works best for running large numbers of very fast computations. The advantage of these two approaches is that the preparation, execution, and analysis of the calculations are performed as a single job.

For more demanding calculations on large datasets, these options are not sufficient. Here, the best option is to perform the individual computations that make up the datasets independently and distribute them across multiple nodes of a cluster. This can also be automated with Cuby, but the workflow is now divided into three steps:

1) All calculations can be prepared with the usual input for a Cuby dataset calculation, but with an additional keyword '`prepare_only: yes`' that stops Cuby before executing the calculations. At this stage, Cuby has prepared the individual calculations to be performed by the external software as separate directories.

2) These individual calculations can be submitted to the cluster independently. The user must make sure that these calculations generate the output files expected by Cuby in each of the calculation directories.

3) Once all calculations are finished, Cuby can be invoked again to read and analyze the results. This time, additional keywords '`existing_calc_dir: read_results`' and '`job_cleanup: no`' are needed.

Step 2) can be further simplified by avoiding the direct call to the external software, which would have to be managed by the user. Cuby has a somewhat surprising feature, an interface to itself,[35] which is very useful here. Using it as a proxy layer between the dataset protocol and the actual calculations forces Cuby to generate separate Cuby inputs (in their own directories) for all entries of the dataset. Combined with Cuby's support for job management systems used on HPC clusters,[36] it is possible to build and submit the

computations without any custom scripting. An example of this workflow is shown in Listing 6.

Listing 6: *Cuby input preparing and reading the dataset broken down to individual calculations executed independently.*

```
job: dataset
dataset: S66

# Distributed calculations of dataset in three steps:

# 1) Prepare calculations only by running cuby once
#    with the following keyword:
prepare_only: yes

# 2) Run the individual calculations
#    and wait for them to finish

# 3) Once the calculations finished, remove the keyword above
#    and read the results in a separate run with:
existing_calc_dir: read_results
job_cleanup: no

# Proxy interface is used to generate Cuby inputs
# for the entries
interface: cuby

# The actual calculation is defined in a separate block
calculation:
  interface: psi4
  method: MP2
  basisset: aug-cc-pVTZ
  # Here, setup for submitting the calculations to
  # a job management system at a cluster can be added:
  queue_submit: yes
  # ...
```

Reuse of previously computed data

One of the main advantages of using established benchmark datasets is the ability to reuse previously calculated data. By using data available in the literature or in public databases,

it is possible to avoid repeating the calculations altogether. Also, it is often useful to reuse newly generated data, changing only parameters that do not require repeating the entire calculation, or analyzing the same data in different ways. Cuby offers several ways of achieving this.

The simplest way is to keep the calculations already built and calculated by Cuby (these are deleted by default, keyword '`job_cleanup: no`' has to be used) and read the results without recalculation in the following calls of Cuby with keyword '`existing_calc_dir: read_results`' added to the input file. For example, an entire dataset can be computed first, and then the results can be analyzed in user-defined subsets (as shown above).

A more permanent solution is to store the additional data in the dataset definition file itself in the 'alternative reference' section. Some of the datasets that are bundled with Cuby already contain a variety of data in this section, which are listed in the documentation of each dataset. User data can be easily added as a list of results in YAML format, preferably to a local copy of the dataset file. The primary purpose of this feature is to provide an alternative reference against which the computed results are evaluated (using '`dataset_reference`'), but they can also be used to replace or modify the benchmarked calculations using the '`dataset_add_ref_to_result`' keyword. This keyword supports mathematical expressions combining multiple reference values.

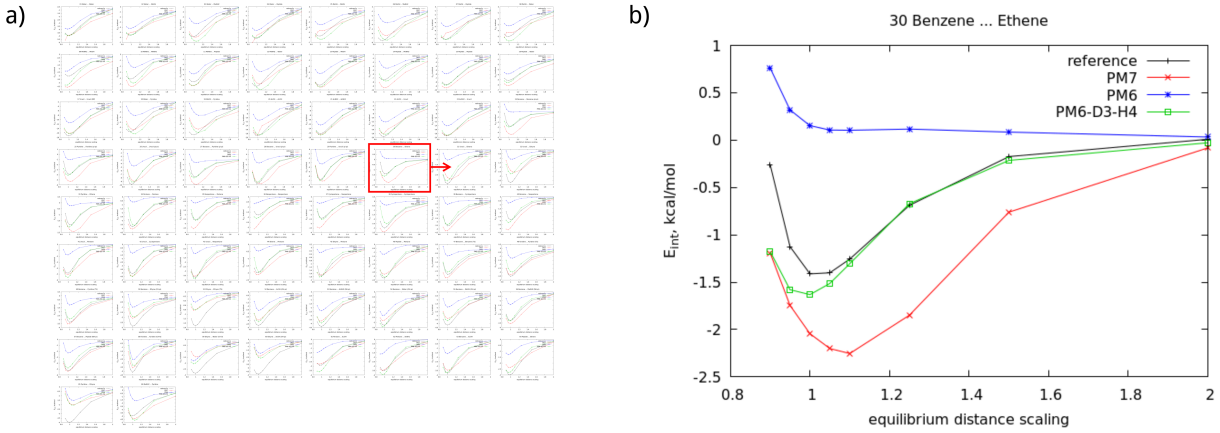
Plotting the data

Some of the datasets packaged with Cuby are dissociation curves of non-covalent complexes, and it is useful to present the results calculated on the dataset as plots. Once the series representing the curves are listed in the dataset definition file (which is the case for all predefined dissociation curve datasets), Cuby can automatically generate plots in several different formats. In the background, the gnuplot tool is used to generate the plots either as images or as gnuplot scripts to generate publication-quality figures. In the simplest case,

adding the keyword '`dataset_save_plots: gnuplot_tiled`' to the input will generate a panel of plots of all curves in the dataset. More options related to this feature are described in the dataset protocol documentation.[37]

It is also possible to combine the results of multiple calculations using different methods into a single plot. An example, a plot of multiple semiempirical methods in the S66×8 set of dissociation curves, is shown in Figure 1. The input file used to generate the plots is provided in the Supporting Information as Listings S1.

Figure 1: Dissociation curves with multiple semiempirical methods are plotted together for the S66×8 dataset with a single input file (Listing S1 in SI). a) shows a panel of plots and b) one of the plots enlarged.



New benchmark databases included in Cuby

Non-Covalent Interactions Atlas datasets

We have recently added two large databases of benchmark datasets to Cuby. The first is the Non-Covalent Interactions Atlas database, a collection of seven datasets covering different classes of non-covalent interactions developed in our group. The NCIAtlas database aims to provide the highest quality benchmark interaction energies, systematically covering much more diverse chemistry than the previous smaller datasets. The NCIAtlas datasets are listed in Table 3. These datasets can also be browsed at the website `http://www.nciatlas.org`

and downloaded from a GitHub repository referenced therein.

Table 3: The Non-Covalent Interactions Atlas datasets added to Cuby. The notation '$\times N$' in the names of the datasets denotes N-point dissociation curves.

Dataset	Description	Ref.
NCIA_HB375×10	Hydrogen bonds in neutral organic compounds and related decoys	7
NCIA_IHB100×10	Ionic hydrogen bonds of organic molecules.	7
NCIA_HB300SPX×10	Hydrogen bonds featuring S, P and halogens	8
NCIA_Rep739×5	Artificial repulsive contacts in extended chemical space	9
NCIA_SH250×10	σ-hole interactions (halogen, chalcogen and pnictogen bonds)	10
NCIA_D1200	London dispersion in extended chemical space	11
NCIA_D442×10	Dissociation curves for selected systems from the D1200 set	11

In addition to the size of this database (with over 19 thousand data points) and the quality of the benchmark calculations (composite CCSD(T)/CBS estimate extrapolated from large basis sets), a unique feature of this database is the rich metadata that can be easily leveraged in Cuby. Separate datasets cover different classes of non-covalent interactions, and within each class the systems are described by the elements of the atoms that define the interactions. More detailed information about each system is recorded in the 'tags' field and can also be used for selections. An example of Cuby input using this metadata to filter the NCIA_SH250 dataset[10] is shown above in Listing 4.

A special feature of the NCIAtlas datasets is the definition of smaller subsets recorded in the metadata. These subsets have been generated using a clustering analysis that maximizes the diversity of the selected systems.[7] For the applications where smaller datasets are needed, the predefined subsets offer a practical solution, with the added advantage of better reproducibility of such an analysis. In Cuby, these subsets can be selected very easily, e.g. the following input '`dataset_select_tag: cluster050`' selects the subset of 50 entries from the larger dataset.

The GMTKN55 database

The second newly added collection of datasets is the GMTKN55 database, a widely used general benchmark for DFT and other computational methods.[12] It contains 55 different sets covering general main-group thermochemistry, kinetics, and non-covalent interactions (see Table 4 for details). It is an upgraded version of the earlier GMTKN30 database[38] of 30 datasets which was already available in Cuby (and is kept there for backward compatibility, although we do not count it in the total number of datasets).

The authors of GMTKN55 added new datasets (with emphasis on reaction barriers), and also modified major part of the datasets included already in GMTKN30. In particular, they have i) updated the reference values based on their own calculations and from the literature in order to improve the accuracy of the benchmark, ii) updated the optimized geometries where applicable, and iii) extended some of the existing sets, which is also reflected in the names of the datasets, e.g. ISO24 being an extension of ISO22. These changes are documented in the original paper.[12] Only 3 out of the old 30 sets (G21EA, G21IP, ACONF) were kept unchanged.

For use in Cuby, the GMTKN55 datasets and the geometries of the systems they contain have been downloaded from the website maintained by the authors of the database[39] and converted to the dataset format used by Cuby. In order to accommodate the different quantities used in the database, Cuby uses the '`reaction`' protocol which allows the definition arbitrary relative energies. This conversion has been validated by performing DFT calculations on all the datasets in Cuby and comparing them with the published results; the agreement is either perfect or within a reasonable allowance for numerical differences between the different implementation and setup of the calculations. A table of these results is provided as a Supporting Information.

Table 4: Overview of the GMTKN55 datasets newly added to Cuby. N is the number of dataset entries (relative energies) in the group.

Group	Sets	N
Basic properties and reaction energies for small systems	W4-11, G21EA, G21IP, DIPCS10, PA26, SIE4×4, ALKBDE10, YBDE18, AL2X6, HEAVYSB11, NBPRC, ALK8, RC21, G2RC, BH76RC, FH51, TAUT15, DC13	473
Reaction energies for large systems and isomerisation reactions	MB16-43, DARC, RSE43, BSR36, CDIE20, ISO34, ISOL24, C60ISO, PArel	243
Reaction barrier hights	BH76, BHPERI, BHDIV10, INV24, BHROT27, PX13, WCPT18	194
Intermolecular non-covalent interactions	RG18, ADIM6, S22, S66, HEAVY28, WATER27, CARBHB12, PNICO23, HAL59, AHB21, CHB6, IL16	304
Intramolecular non-covalent interactions	IDISP, ICONF, ACONF, Amino20×4, PCONF21, MCONF, SCONF, UPU23, BUT14DIOL	291

Conclusions

The paper describes the application of the Cuby software framework to automate calculations on datasets used in benchmarking computational chemistry methods. The modular nature of the framework allows calculations implemented in various external software to be combined with advanced computational protocols provided by Cuby. All this can be achieved using structured input files, without the need for programming.

Cuby includes a collection of more than 80 predefined datasets that can be computed by changing a single keyword in the input. This collection has recently been expanded to include datasets from the Non-Covalent Interactions Atlas and GMTKN55 databases, and this update is also discussed here.

We hope that this paper demonstrates that the Cuby framework makes the use of benchmark datasets easy and accessible to everyone, and we encourage the readers to give it a try.

Acknowledgements

We acknowledge the support from the Czech Science Foundation, Grant No. 22-17063S.

Data availability statement

The data sets used in this study are openly available as a part of the Cuby4 open-source software package at `http://cuby4.molecular.cz`.

Supporting Information Available

The Supporting Information contains a table of DFT results in the GMTKN55 database comparing the original errors reported by the authors of the database and the errors computed with Cuby, and an additional listing of a sample Cuby input file.

References

(1) Ruby Programming Language. `http://www.ruby-lang.org/en/`.

(2) Řezáč, J. Cuby: An integrative framework for computational chemistry. *J. Comput. Chem.* **2016**, *37*, 1230–1237.

(3) Najibi, A.; Goerigk, L. The Nonlocal Kernel in van der Waals Density Functionals as an Additive Correction: An Extensive Analysis with Special Emphasis on the B97M-V and B97M-V Approaches. *J. Chem. Theory Comput.* **2018**, *14*, 5725–5738.

(4) Santra, G.; Sylvetsky, N.; Martin, J. M. L. Minimally Empirical Double-Hybrid Functionals Trained against the GMTKN55 Database: revDSD-PBEP86-D4, revDOD-PBE-D4, and DOD-SCAN-D4. *J. Phys. Chem. A* **2019**, *123*, 5129–5143.

(5) Bursch, M.; Neugebauer, H.; Ehlert, S.; Grimme, S. Dispersion corrected r2SCAN based global hybrid functionals: r2SCANh, r2SCAN0, and r2SCAN50. *J. Chem. Phys.* **2022**, *156*, 134105.

(6) Fürst, S.; Haasler, M.; Grotjahn, R.; Kaupp, M. Full Implementation, Optimization, and Evaluation of a Range-Separated Local Hybrid Functional with Wide Accuracy for Ground and Excited States. *J. Chem. Theory Comput.* **2023**, *19*, 488–502.

(7) Řezáč, J. Non-Covalent Interactions Atlas Benchmark Data Sets: Hydrogen Bonding. *J. Chem. Theory Comput.* **2020**, *16*, 2355–2368.

(8) Řezáč, J. Non-Covalent Interactions Atlas Benchmark Data Sets 2: Hydrogen Bonding in an Extended Chemical Space. *J. Chem. Theory Comput.* **2020**, *16*, 6305–6316.

(9) Kříž, K.; Nováček, M.; Řezáč, J. Non-Covalent Interactions Atlas Benchmark Data Sets 3: Repulsive Contacts. *J. Chem. Theory Comput.* **2021**, *17*, 1548–1561.

(10) Kříž, K.; Řezáč, J. Non-covalent interactions atlas benchmark data sets 4: -hole interactions. *Phys. Chem. Chem. Phys.* **2022**, *24*, 14794–14804.

(11) Řezáč, J. Non-Covalent Interactions Atlas benchmark data sets 5: London dispersion in an extended chemical space. *Phys. Chem. Chem. Phys.* **2022**, *24*, 14780–14793.

(12) Goerigk, L.; Hansen, A.; Bauer, C.; Ehrlich, S.; Najibi, A.; Grimme, S. A look at the density functional theory zoo with the advanced GMTKN55 database for general main group thermochemistry, kinetics and noncovalent interactions. *Phys. Chem. Chem. Phys.* **2017**, *19*, 32184–32215.

(13) Řezáč, J.; Riley, K. E.; Hobza, P. S66: A Well-balanced Database of Benchmark Interaction Energies Relevant to Biomolecular Structures. *J. Chem. Theory Comput.* **2011**, *7*, 2427–2438.

(14) Řezáč, J.; Riley, K. E.; Hobza, P. Extensions of the S66 Data Set: More Accurate Interaction Energies and Angular-Displaced Nonequilibrium Geometries. *J. Chem. Theory Comput.* **2011**, *7*, 3466–3470.

(15) Řezáč, J.; Riley, K. E.; Hobza, P. Benchmark calculations of noncovalent interactions of halogenated molecules. *J. Chem. Theory Comput.* **2012**, *8*, 4285–4292.

(16) The Official YAML Web Site. `http://yaml.org/`.

(17) Turney, J. M.; Simmonett, A. C.; Parrish, R. M.; Hohenstein, E. G.; Evangelista, F. A.; Fermann, J. T.; Mintz, B. J.; Burns, L. A.; Wilke, J. J.; Abrams, M. L.; Russ, N. J.; Leininger, M. L.; Janssen, C. L.; Seidl, E. T.; Allen, W. D.; Schaefer, H. F.; King, R. A.; Valeev, E. F.; Sherrill, C. D.; Crawford, T. D. Psi4: an open-source ab initio electronic structure program. *WIREs Comput. Mol. Sci.* **2012**, *2*, 556–565.

(18) Řezáč, J. Cuby 4 documentation. 2024; `http://cuby4.molecular.cz`.

(19) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An overview of the Amber biomolecular simulation package. *WIREs Computational Molecular Science* **2013**, *3*, 198–210.

(20) Stanton, J. F.; Gauss, J.; Harding, M. E.; Szalay, P. G.; et al. CFOUR, Coupled-Cluster techniques for Computational Chemistry. See also www.cfour.de. `www.cfour.de`.

(21) Brooks, B. R.; Brooks III, C. L.; Mackerell Jr., A. D.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry* **2009**, *30*, 1545–1614.

(22) Geudtner, G.; Calaminici, P.; Carmona-Espíndola, J.; del Campo, J. M.; Domínguez-Soria, V. D.; Moreno, R. F.; Gamboa, G. U.; Goursot, A.; Köster, A. M.; Reveles, J. U.; Mineva, T.; Vásquez-Pérez, J. M.; Vela, A.; Zúñinga-Gutierrez, B.; Salahub, D. R. deMon2k. *WIREs Comput. Mol. Sci.* **2012**, *2*, 548–555.

(23) Hourahine, B.; Aradi, B.; Blum, V.; Bonafé, F.; Buccheri, A.; Camacho, C.; Cevallos, C.; Deshaye, M. Y.; Dumitrică, T.; Dominguez, A.; Ehlert, S.; Elstner, M.; van der Heide, T.; Hermann, J.; Irle, S.; Kranz, J. J.; Köhler, C.; Kowalczyk, T.; Kubař, T.; Lee, I. S.; Lutsker, V.; Maurer, R. J.; Min, S. K.; Mitchell, I.; Negre, C.; Niehaus, T. A.; Niklasson, A. M. N.; Page, A. J.; Pecchia, A.; Penazzi, G.; Persson, M. P.; Řezáč, J.; Sánchez, C. G.; Sternberg, M.; Stöhr, M.; Stuckenberg, F.; Tkatchenko, A.; Yu, V. W.-z.; Frauenheim, T. DFTB+, a software package for efficient approximate density functional theory based atomistic simulations. *J. Chem. Phys.* **2020**, *152*, 124101.

(24) Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. General atomic and molecular electronic structure system. *J. Comput. Chem.* **1993**, *14*, 1347–1363.

(25) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; Li, X.; Caricato, M.; Marenich, A. V.; Bloino, J.; Janesko, B. G.; Gomperts, R.; Mennucci, B.; Hratchian, H. P.; Ortiz, J. V.; Izmaylov, A. F.; Sonnenberg, J. L.; Williams-Young, D.; Ding, F.; Lipparini, F.; Egidi, F.; Goings, J.; Peng, B.; Petrone, A.; Henderson, T.; Ranasinghe, D.; Zakrzewski, V. G.; Gao, J.; Rega, N.; Zheng, G.; Liang, W.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Throssell, K.; Montgomery, J. A., Jr.; Peralta, J. E.; Ogliaro, F.; Bearpark, M. J.; Heyd, J. J.; Brothers, E. N.; Kudin, K. N.; Staroverov, V. N.; Keith, T. A.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Ren-

dell, A. P.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Millam, J. M.; Klene, M.; Adamo, C.; Cammi, R.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Farkas, O.; Foresman, J. B.; Fox, D. J. Gaussian 16 Revision C.01. 2016.

(26) Werner, H.-J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M. Molpro: a general-purpose quantum chemistry program package. *WIREs Comput Mol Sci* **2012**, *2*, 242–253.

(27) Stewart, J. J. P. MOPAC 2016. 2016; `http://openmopac.net/`.

(28) Kállay, M.; Rolik, Z.; Csontos, J.; Ladjánszki, I.; Szegedy, L.; Ladóczki, B.; Samu, G. MRCC, a quantum chemical program suite, see also www.mrcc.hu. `www.mrcc.hu`.

(29) Neese, F. Software update: the ORCA program system, version 4.0. *WIREs Computational Molecular Science* **2018**, *8*, e1327.

(30) Franzke, Y. J.; Holzer, C.; Andersen, J. H.; Begušić, T.; Bruder, F.; Coriani, S.; Della Sala, F.; Fabiano, E.; Fedotov, D. A.; Fürst, S.; Gillhuber, S.; Grotjahn, R.; Kaupp, M.; Kehry, M.; Krstić, M.; Mack, F.; Majumdar, S.; Nguyen, B. D.; Parker, S. M.; Pauly, F.; Pausch, A.; Perlt, E.; Phun, G. S.; Rajabi, A.; Rappoport, D.; Samal, B.; Schrader, T.; Sharma, M.; Tapavicza, E.; Treß, R. S.; Voora, V.; Wodyński, A.; Yu, J. M.; Zerulla, B.; Furche, F.; Hättig, C.; Sierka, M.; Tew, D. P.; Weigend, F. TURBOMOLE: Today and Tomorrow. *J. Chem. Theory Comput.* **2023**, *19*, 6859–6890.

(31) XTB, Semiempirical Extended Tight-Binding Program Package. 2019; `https://github.com/grimme-lab/xtb`.

(32) Řezáč, J. Cuby 4 documentation: List of interfaces. 2024; `http://cuby4.molecular.cz/interfaces.html`.

(33) Stewart, J. J. P. Optimization of parameters for semiempirical methods V: Modification of NDDO approximations and application to 70 elements. *J Mol Model* **2007**, *13*, 1173–1213.

(34) Řezáč, J. Cuby 4 documentation: Geometry database. 2024; `http://cuby4.molecular.cz/geometry.html#database`.

(35) Řezáč, J. Cuby 4 documentation: Cuby interface. 2024; `http://cuby4.molecular.cz/interface_cuby.html`.

(36) Řezáč, J. Cuby 4 documentation: Cuby on clusters. 2024; `http://cuby4.molecular.cz/queue_systems.html`.

(37) Řezáč, J. Cuby 4 documentation: Dataset protocol. 2024; `http://cuby4.molecular.cz/protocol_dataset.html`.

(38) Goerigk, L.; Grimme, S. Efficient and Accurate Double-Hybrid-Meta-GGA Density Functionals—Evaluation with the Extended GMTKN30 Database for General Main Group Thermochemistry, Kinetics, and Noncovalent Interactions. *J. Chem. Theory Comput.* **2011**, *7*, 291–309.

(39) Goerigk, L.; Hansen, A.; Bauer, C.; Ehrlich, S.; Najibi, A.; Grimme, S. GMTKN55 - A database for general main group thermochemistry, kinetics, and non-covalent interactions. 2024; `https://www.chemie.uni-bonn.de/grimme/de/software/gmtkn/gmtkn55`.