



University of
Zurich^{UZH}

CP2K: Automation, Scripting, Testing

Tiziano Müller

`tiziano.mueller@chem.uzh.ch`

CP2K Workshop @ UGent, 11.-13. March 2019

Dept. of Chemistry, UZH

Preparations

Installation

Verification

Reproducibility

Syntax-Checking & Input-Debugging

Archival

Input Generation

Structure-only: Supported formats

Full Input generation: GUIs

Full Input generation: Scripting

CP2K Preprocessor

“Run” Automation

Batch Processing

Workflows

Integration: Phonopy, PyRETIS, i-PI

Basis Set Verification

Preparations

```
$ git clone --recursive https://github.com/cp2k/cp2k.git
$ cd cp2k/tools/toolchain
$ ./install_cp2k_toolchain.sh
MPI is detected and it appears to be OpenMPI
nvcc not found, disabling CUDA by default
Compiling with 8 processes.
===== Finding binutils from system paths =====
[...]
===== generating arch files =====
arch files can be found in the /data/cp2k/tools/toolchain/install/arch subdirectory
Wrote /data/cp2k/tools/toolchain/install/arch/local.sopt
Wrote /data/cp2k/tools/toolchain/install/arch/local.sdbg
Wrote /data/cp2k/tools/toolchain/install/arch/local.ssmpt
[...]
===== usage =====
Done!
Now copy:
  cp /data/cp2k/tools/toolchain/install/arch/* to the cp2k/arch/ directory
To use the installed tools and libraries and cp2k version
compiled with it you will first need to execute at the prompt:
  source /data/cp2k/tools/toolchain/install/setup
To build CP2K you should change directory:
  cd cp2k/
  make -j 8 ARCH=local VERSION="sopt sdbg ssmpt popt pdbg psmp"
```

- Default: Uses system compiler, linker and MPI
- Builds and configures for: libxc, libint, libxsmm, ELPA, SIRIUS
- Support for Linux & macOS

- Build everything:

```
$ ./install_cp2k_toolchain.sh --install-all
```

- More options:

```
$ ./install_cp2k_toolchain.sh --help
```

→ Fortran requires `.mod` files and code built with same compiler!

- Manual clean (`build/`, `install/`) recommended after re-configuration
- Check https://www.cp2k.org/dev:compiler_support for supported compilers and libraries

Building CP2K with Spack



Spack

```
$ git clone https://github.com/spack/spack.git
$ . ./share/spack/setup-env.sh
$ spack install cp2k
$ spack load cp2k
```

- Package manager for scientific software
- Requires Python
- Automatically detects and reuses available compiler
- Recursively builds all CP2K prerequisites
- Installs CP2K and the arch-file used to build it

```
$ spack info cp2k
MakefilePackage:  cp2k
```

Description:

CP2K is a quantum chemistry and solid state physics software package that can perform atomistic simulations of solid state, liquid, molecular, periodic, material, crystal, and biological systems

Homepage: <https://www.cp2k.org>

Tags:

None

Preferred version:

6.1 <https://github.com/cp2k/cp2k/releases/download/v6.1.0/cp2k-6.1.tar.bz2>

Safe versions:

6.1 <https://github.com/cp2k/cp2k/releases/download/v6.1.0/cp2k-6.1.tar.bz2>
[...]

Variants:

Name [Default]	Allowed values	Description
blas [openblas]	openblas, mkl, accelerate	Enable the use of OpenBlas/MKL/Accelerate
elpa [off]	True, False	Enable optimised

```
$ spack find -p cp2k
==> 1 installed package
-- linux-opensuse_leap15-x86_64 / gcc@7.3.1 -----
   cp2k@6.1  [...] / linux-opensuse_leap15-x86_64 / gcc-7.3.1 / cp2k-6.1-byjtwyhrqqmzevpy3zwiccccmexshd

$ ls [...] / cp2k-6.1-byjtwyhrqqmzevpy3zwiccccmexshd / .spack / archived-files / arch /
linux-opensuse_leap15-x86_64-gcc.popt
```

- Use Spack arch-file for custom build of CP2K with Spack-installed libraries
- By default Spack builds all dependencies except compiler & linker.
Extra configuration needed to use system-MPI.

Alternative:



State of latest version

CP2K DASHBOARD					
Name	Host	Status	Commit	Summary	Last OK
Formatting	GCP	OK	821fb88 (0)	Checked 1115 files.	
Coding conventions	GCP	OK	821fb88 (0)	Found 0 issues (292 suppressed)	
Linux_s86-64-efortran.gdhe	PSI, MellinS	OK	821fb88 (0)	correct: 3106 / 3106; 42min	
Linux_s86-64-efortran.somp	PSI, MellinS	OK	821fb88 (0)	correct: 3031 / 3031; 43min	
Linux_s86-64-efortran.asmp	PSI, MellinS	OK	821fb88 (0)	correct: 3106 / 3106; 65min	
Linux_s86-64-intel.popt (v18.0.5.274)	PSI, MellinS	OK	821fb88 (0)	correct: 3106 / 3106; 99min	
Linux_s86-64-intel.somp (v18.0.5.274)	PSI, MellinS	FAILED	821fb88 (0)	correct: 3105 / 3106; failed: 1; 124min	h7143ef (-1)
Linux_s86-4.9.2_mkl-11.2.1_sopt	UZH, opt5	OUTDATED	h7143ef (-1)	correct: 3031 / 3031; 6min	h7143ef (-1)
Linux_s86-4.9.2_mkl-11.2.1_popt	UZH, opt5	OUTDATED	h7143ef (-1)	correct: 3090 / 3090; 8min	h7143ef (-1)
Linux_s86-4.9.2_mkl-11.2.1_somp	UZH, opt5	OUTDATED	h7143ef (-1)	correct: 3031 / 3031; 6min	h7143ef (-1)
Linux_s86-4.9.2_mkl-11.2.1_pomp	UZH, opt5	OUTDATED	h7143ef (-1)	correct: 3090 / 3090; 11min	h7143ef (-1)
CRAY_XC50-efortran_s86.somp	CSCS, PwDaint	OUTDATED	821fb88 (0)	correct: 3013 / 3015; wrong: 2; 120min	af6872 (-8)
CRAY_XC50-efortran_msc.asmp	CSCS, PwDaint	OUTDATED	821fb88 (0)	correct: 1257 / 3015; wrong: 1; failed: 1; 62min	af6872 (-8)
Current_Toolchain (pdhe)	GCP	OK	821fb88 (0)	correct: 3110 / 3110; 17min	
Current_Toolchain (popt)	GCP	OK	821fb88 (0)	correct: 3110 / 3110; 11min	
Current_Toolchain (somp)	GCP	OK	821fb88 (0)	correct: 3112 / 3112; 15min	
Current_Toolchain (pdhe)	GCP	OK	821fb88 (0)	correct: 3032 / 3032; 12min	

<https://dashboard.cp2k.org>

Multiple platforms/architectures available, including full logs and their arch-files.

Verify your build

```
$ make VERSION=sopt ARCH=local test
```

```
CP2K supports: cp2kflags: libint fftw3 libxc libderiv_max_am1=5 libint_
Skipping QS/regtest-cdft-hirshfeld-2 : missing required feature : parall
Skipping QS/regtest-cdft-hirshfeld-2 : missing required feature : mpiroa
[...]
```

```
----- Summary -----
```

```
Number of FAILED tests 0
Number of WRONG tests 0
Number of CORRECT tests 3031
Number of NEW tests 0
Total number of tests 3031
GREPME 0 0 3031 0 3031 X
Summary: correct: 3031 / 3031; 6min
Status: OK
```

```
-----
Regtest took 379.00 seconds.
-----
```

```
Thu Feb 28 15:33:59 CET 2019
```

```
***** testing ended *****
```

→ Automatically skips unavailable features

Reproducibility

```
$ cp2k -c your.inp
```

- Basic issues are found
 - Complex tests only at full runtime
- Use low cutoffs, limit SCF cycles to get a full check
(MAX_SCF, MAX_STEPS, ...)

```
.....  
*  ___  *  
*  /    *  
* [ABORT] *  
*  ___/   found an unknown keyword APACHE in section __ROOT__ *  
*    |    *  
*  0/|    *  
* /| |    *  
* /      *  
*                               input/input_parsing.F:246 *  
.....
```

Output capturing:

```
$ cp2k your.inp |& tee your.out
```

```
$ cp2k your.inp -o your.out (production run)
```

- Leave error output handling to batch-system if possible

```
$ cp2k -e your.inp
```

- Full-input: includes current default settings & resolved preprocessor variables
- Can also be used for debugging complex inputs and parsing errors
- Other artefacts:
 - **POTENTIAL**
 - **BASIS_SET**
 - Structure files:
 - **.xyz, .pdb, ...**
 - Force field, dispersion correction parameter, DFTB files
 - **proj-1.restart** (a full input file)
 - **proj-pos-1.xyz** (MD/GEO_OPT trajectories)
 - **proj-1.ener** (MD energies, temperature, ...)
 - **proj-1.cell** (cell parameters for CELL_OPT, NPT MD)
 - **proj-RESTART.wfn, proj-RESTART.kp** (orbitals for restart)

Input Generation

Use your favorite molecular structure editor.

Supported formats:¹

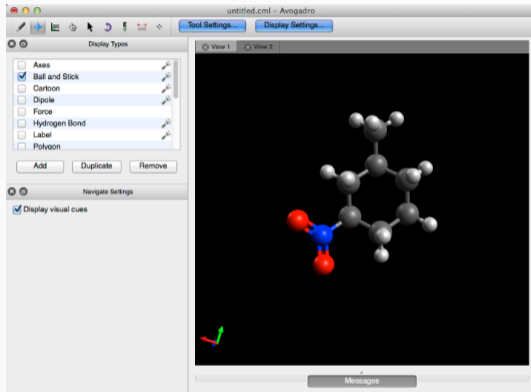
XYZ (coords only), PDB, CIF,
G96/G87 (GROMACS), PSF/UPSF (CHARMM), CRD (AMBER), XTL

¹*.restart files have coordinates integrated as &COORD section

Full Input generation: Avogadro

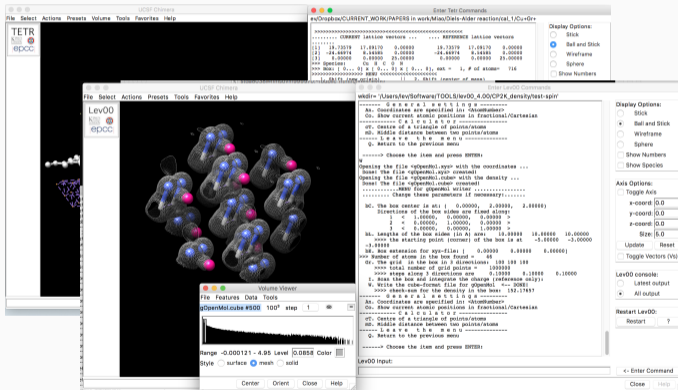


Avogadro



- CP2K Plugin for Avogadro 1.x:
<https://github.com/brhr-iwao/libavogadro1cp2k>
- CP2K Plugin for Avogadro 2.x:
<https://github.com/svedruziclab/avogadrolibs-cp2k>

- Menu-driven + visualisation
- TETR: pre-processing
 - geometry setup
 - supercell, surfaces, clusters calculation
- LEV00: analysis
 - Visualising charge & spin densities
 - DOS, Phonons, IR spectra



TETR+LEV00 Plugin for Chimera



- Domain Specific Language (DSL) with Python
- Keywords match CP2K input file syntax
- Integration with Python ASE
- Auto-completion based on Python auto-completion engines

```
from pycp2k import CP2K
from ase.lattice.cubic import Diamond

##### Create the structure with ASE #####
lattice = Diamond(directions=[[1, 0, 0], [0, 1, 0], [0, 0, 1]],
                  symbol='Si',
                  latticeconstant=5.430697500,
                  size=(1, 1, 1))

##### Define and setup the calculator object #####
calc = CP2K()
calc.working_directory = "./"
calc.project_name = "si_bulk"
calc.mpi_n_processes = 2

##### An existing input file can be parsed #####
calc.parse("template.in")

##### Define shortcuts for easy access #####
CP2K_INPUT = calc.CP2K_INPUT
GLOBAL = CP2K_INPUT.GLOBAL
FORCE_EVAL = CP2K_INPUT.FORCE_EVAL_add() # Repeatable items have to be first created
SUBSYS = FORCE_EVAL.SUBSYS
DFT = FORCE_EVAL.DFT
SCF = DFT.SCF

##### Write the simulation input #####
GLOBAL.Run_type = "ENERGY_FORCE"
FORCE_EVAL.Method = "Quickstep"

PYCP2K
```

- Powerful structure building tools
- Merging with pre-existing input file structure possible (templating)
- Uses `cp2k_shell` to run CP2K continuously → minimal overhead
- Can start CP2K on remote machine

```
import numpy as np

from ase.build import bulk
from ase.constraints import UnitCellFilter
from ase.optimize import MDMin
from ase.calculators.cp2k import CP2K

inp = """&FORCE_EVAL
      &MM
      &FORCEFIELD
      &SPLINE
          EMAX_ACCURACY 500.0
          EMAX_SPLINE 1000.0
          EPS_SPLINE 1.0E-9
      &END
      &NONBONDED
      &LENNARD-JONES
          atoms Ar Ar
          EPSILON [eV] 1.0
          SIGMA [angstrom] 1.0
          RCUT [angstrom] 10.0
      &END LENNARD-JONES
      [...]
&END FORCE_EVAL"""

calc = CP2K(label="test_stress", inp=inp, force_eval_method="Fist")

# Theoretical infinite-cutoff LJ FCC unit cell parameters
```

CP2K input may include extra directives which are evaluated before everything else:

@INCLUDE 'filename.inc'

The content of the specified file are included at this point. The path is assumed to be relative to the current working directory.

@SET VAR value

(re-)define a variable

\${VAR} or \$VAR

Replaced by the content of the previously set variable VAR

@IF .../@ENDIF

Conditionals. Supported operators: == and /= (lexical comparison). The value 0 or whitespaces evaluate to **FALSE**, everything else to **TRUE**.

@PRINT ...

Print the given text while pre-processing



→ `settings.inp` can contain `@SET` other `@INCLUDE` or full sections/keywords

“Run” Automation

"Run" automation

Batch Processing

Shell Scripts

Python Scripts

Scheduler assisted

CP2K Farming

Workflows

AiiDA

atomate

```
&GLOBAL
  PROJECT OldMacDonald
  PROGRAM FARMING
  RUN_TYPE NONE
&END GLOBAL
&FARMING
  NGROUPS 2 ! number of parallel jobs
  MASTER_SLAVE ! for load balancing
  GROUP_SIZE 42 ! number of processors per group, default: 8
  &JOB
    JOB_ID 1 ! optional, required for dependencies
    DIRECTORY dir-1
    INPUT_FILE_NAME water.inp
    OUTPUT_FILE_NAME water.out
  &END JOB
  &JOB
    DEPENDENCIES 1
    DIRECTORY dir-2
    INPUT_FILE_NAME water.inp
    OUTPUT_FILE_NAME more_water.out
  &END JOB
  [...]
&END FARMING
```

set to NONE

use FARMING section

insert original input here

- Jobs are run inside the same CP2K process
- MPI gets initialized once
→ reduced startup time
- Useful for many small jobs



Automated
Interactive Infrastructure
and Database

- Python-based
- Strong focus on Data Provenance
- Database backend (PostgreSQL) + File Repository
- Advanced workflow engine on top of Python
- Plugin architecture:
 - [CP2K Plugin](#)
 - [Gaussian Basis Set and Pseudopotential Plugin](#)
 - more in the [AiiDA Plugin Registry](#)
- Jupyter Notebook integration
- Integration with the [MaterialsCloud](#) Open Science platform

Workflows: AiiDA Example

```

calc = Code.get_from_string("cp2k").new_calc()

calc.label = "Awesome CP2K calculation"

atoms = ase.build.molecule('H2O') # build structure
atoms.center(vacuum=2.0)
structure = StructureData(ase=atoms)
calc.use_structure(structure) # ... or reuse existing

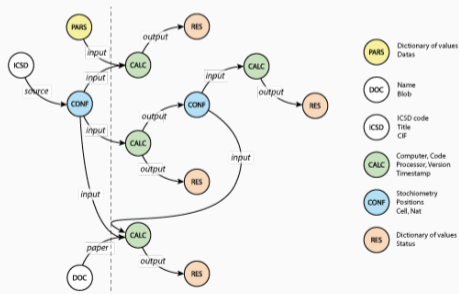
parameters = ParameterData(dict={
    'FORCE_EVAL': {
        'METHOD': 'Quickstep',
        'DFT': {
            'QS': {
                'EPS_DEFAULT': 1.0e-12,
            },
        },
        [...]
    },
})
calc.use_parameters(parameters)

calc.set_max_wallclock_seconds(3*60)
calc.set_resources({'num_machines': 4})
calc.set_computer(Computer.get("skitty"))

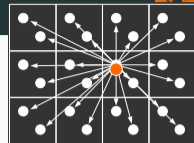
calc.store_all() # store in database
calc.submit() # submit for calculation

```

- Runs on your machine
- Manages job submission and retrieval (incl. scheduler support)
- Tracks structures & calculations

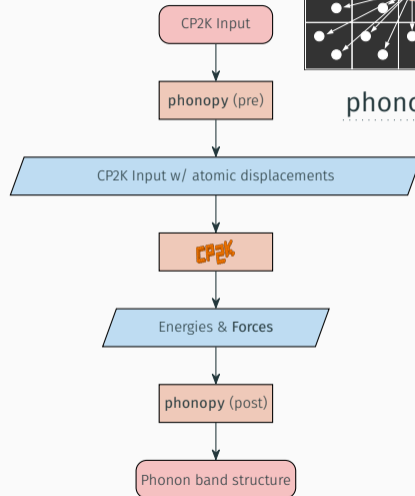


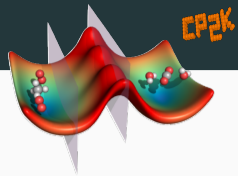
Integration: Phonopy, PyRETIS, i-PI



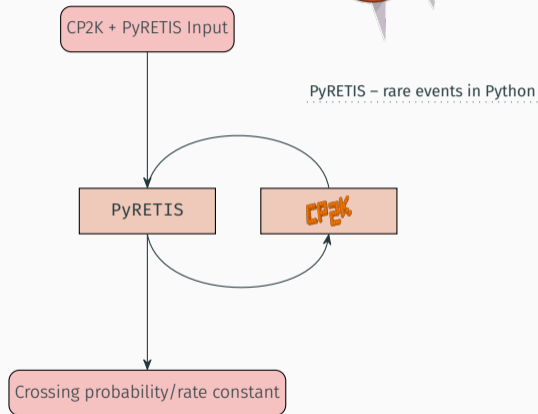
phonopy

- Python based
- File-based interface:
parses & generates code inputs
- Only needs equilibrated and
symmetrized crystal structure input
- Uses a supercell approach
- Can also generate:
DOS, pDOS, Thermal properties





- Python based
- Transition Interface Sampling (TIS) and Replica Exchange Transition Interface Sampling (RETIS)
- Can use CP2K as integrator for MD steps

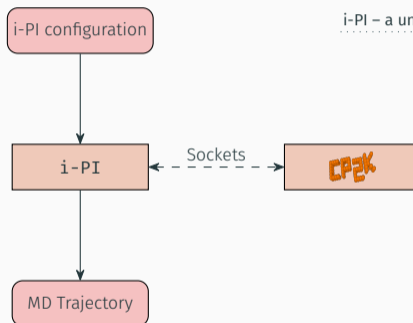


i-PI: a universal force engine



i-PI – a universal force engine

- Python based
- Focus on Path Integral Molecular Dynamics
- Communication with Force Engines via network sockets
- Many additional methods available



Basis Set Verification

The Elephant in the Room of Density Functional Theory Calculations

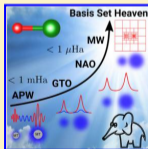
Stig Rune Jensen,[†] Santanu Saha,[‡] José A. Flores-Livas,[‡] William Huhn,[§] Volker Blum,[†]
Stefan Goedecker,[‡] and Luca Frediani[†]

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

[‡]Department of Physics, Universität Basel, Klingelbergstrasse 82, 4056 Basel, Switzerland

[§]Department of Mechanical Engineering and Materials Science, Duke University, Durham, North Carolina 27708, United States

ABSTRACT: Using multiwavelets, we have obtained total energies and corresponding atomization energies for the GGA-PBE and hybrid-PBE0 density functionals for a test set of 211 molecules with an unprecedented and guaranteed μ Hartree accuracy. These quasi-exact references allow us to quantify the accuracy of standard all-electron basis sets that are believed to be highly accurate for molecules, such as Gaussian-type orbitals (GTOs), all-electron numeric atom-centered orbitals (NAOs), and full-potential augmented plane wave (APW) methods. We show that NAOs are able to achieve the so-called chemical accuracy (1 kcal/mol) for the typical basis set sizes used in applications, for both total and atomization energies. For GTOs, a triple- ζ quality basis has mean errors of ~ 10 kcal/mol in total energies, while chemical accuracy is almost reached for a quintuple- ζ basis. Due to systematic error cancellations, atomization energy errors are reduced by almost an order of magnitude, placing chemical accuracy within reach also for medium to large GTO bases, albeit with significant outliers. In order to check the accuracy of the computed densities, we have also investigated the dipole moments, where in general only the largest NAO and GTO bases are able to yield errors below 0.01 D. The observed errors are similar across the different functionals considered here.



“For GTOs, a triple- ζ quality basis has mean errors of ~ 10 kcal/mol in total energies, while chemical accuracy is almost reached for a quintuple- ζ basis...”

→ Do we really need larger basis sets?

This is an open access article published under an ACS AuthorChoice License, which permits copying and redistribution of the article or any adaptations for non-commercial purposes.

THE JOURNAL OF PHYSICAL CHEMISTRY A

Article
pubs.acs.org/JPCA

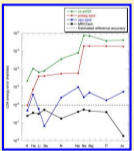
How Large is the Elephant in the Density Functional Theory Room?

Frank Jensen[✉]

Department of Chemistry, Aarhus University, Langelandsgade 140, DK-8000 Aarhus, Denmark

[Supporting Information](#)

ABSTRACT: A recent paper reported highly accurate density functional theory results for atomization energies and dipole moments using a multiwavelet-based method and compared the results with those obtained by standard Gaussian basis sets of the aug-cc-pVXZ type. Typical errors with the large aug-cc-pV5Z basis set were in the 0.2 kcal/mol range with outliers displaying errors of ~2 kcal/mol, and these results could be taken as an indication that Gaussian basis sets in general are unsuitable for achieving high accuracy. We show that by choosing Gaussian basis sets optimized for density functional theory, basis set methods are capable of achieving accuracy comparable to that from the multiwavelet approach.



“We show that by choosing Gaussian basis sets optimized for density functional theory, basis set methods are capable of achieving accuracy comparable to that from the multiwavelet approach...”

→ Not necessarily, just use the right one.

Frank Jensen. In: *J. Phys. Chem. A* 121.32 (Aug. 17, 2017). 00002, pp. 6104–6107.

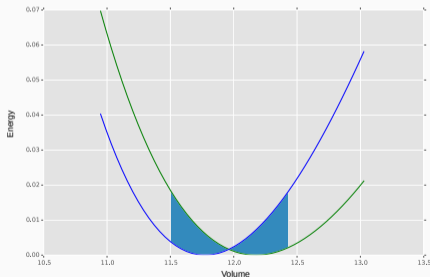
DOI: [10.1021/acs.jpca.7b04760](https://doi.org/10.1021/acs.jpca.7b04760)

Basis Set Verification: The Deltatest

- Solid-state benchmark²
- Measure: Difference between two Volume/Energy-curves
- 40+ “Methods”, 71 Elements (H-Rn, elemental crystals)
- DFT, PBE Functional
- All-Electron calculations as reference

Δ -Gauge³

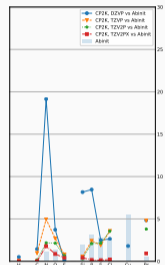
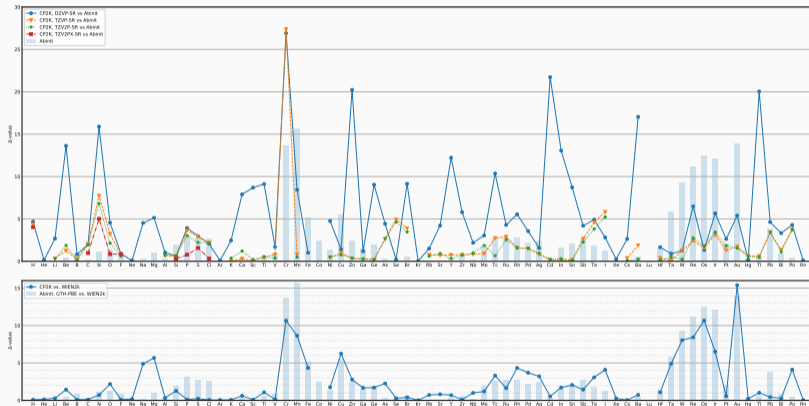
$$\Delta_i(a, b) = \sqrt{\frac{\int_{0.94V_{0,i}}^{1.06V_{0,i}} (E_{b,i}(V) - E_{a,i}(V))^2 dV}{0.12V_{0,i}}}$$



² Kurt Lejaeghere et al. In: *Science* 351.6280 (Mar. 25, 2016). 00079, aad3000. DOI: [10.1126/science.aad3000](https://doi.org/10.1126/science.aad3000)

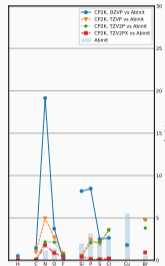
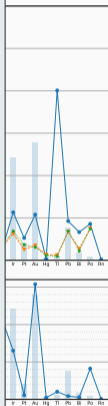
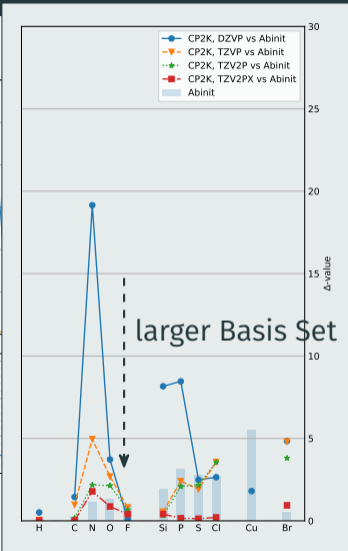
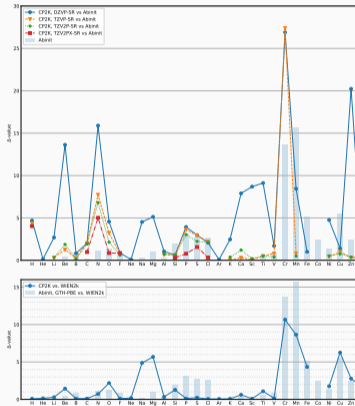
³ Kurt Lejaeghere et al. In: *Crit. Rev. Solid State* 39.1 (Jan. 1, 2014). 00112, pp. 1–24. DOI: [10.1080/10408436.2013.772503](https://doi.org/10.1080/10408436.2013.772503)

Basis Set Verification: Deltatest results for CP2K's MOLOPT Basis Set



↑ non-SR MOLOPT basis sets

Basis Set Verification: Deltatest results for CP2K's MOLOPT Basis Set



↑ non-SR MOLOPT basis sets

- MOLOPT basis set suitable for solid state calculations
- Larger- ζ MOLOPT basis sets systematically improve results
- Basis set related errors in same order as pseudization error
- For some elements: basis set inadvertently compensates pseudopotential error
- Publication of complete data and workflow in *Discovery* section of <http://www.materialscloud.org>
- Testing of All-Electron Peintinger⁴ basis set in progress
- More benchmarks coming

⁴Michael F. Peintinger et al. In: *J. Comput. Chem.* 34.6 (2013). 00455, pp. 451–459. DOI: [10.1002/jcc.23153](https://doi.org/10.1002/jcc.23153)

- <https://www.cp2k.org> Exercises, Lecture Slides
- <https://manual.cp2k.org> Input File reference
- cp2k-x.y/tests Minimal Working Examples
- <https://groups.google.com/group/cp2k> Google Group/Forum
- <https://github.com/cp2k/cp2k/issues> Issue Tracker

Thank you!