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

Outline

Preparations

Building CP2K with the toolchain scripts

```
$ 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.ssmp
[...]
========================== 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 ssmp popt pdbg psmp"
```

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

Building CP2K with the toolchain scripts: Configuration

- 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

Building CP2K with Spack: Configuration

```
$ 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,          Enable the use of
                       accelerate              OpenBlas/MKL/Accelerate
    elpa [off]         True, False             Enable optimised
```

Building CP2K with Spack: Locating the arch-file

```
$ 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-byjtwnyhrqqmzezvpy3zwiccccmexshd

$ ls [...]/cp2k-6.1-byjtwnyhrqqmzezvpy3zwiccccmexshd/.spack/archived-files/arch/
linux-opensuse_leap15-x86_64-gcc.popt
```

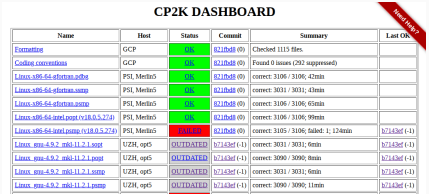
→ Use Spack arch-file for custom build of CP2K with Spack-installed libraries

- By default Spack builds all dependencies except compiler & linker.
  Extra configuration needed to use system-MPI.

Alternative:

Testing CP2K

State of latest version



CP2K DASHBOARD

Most Hits!!!

Name	Host	Status	Commit	Summary	Last OK
Formatting	GCP	OK	#21fbd8 (0)	Checked 1115 files.	
Coding conventions	GCP	OK	#21fbd8 (0)	Found 0 issues (292 suppressed)	
Linux-x86-64-gfortran-pdbg	PSI, Merlin5	OK	#21fbd8 (0)	correct: 3106 / 3106; 42min	
Linux-x86-64-gfortran-ssmp	PSI, Merlin5	OK	#21fbd8 (0)	correct: 3031 / 3031; 43min	
Linux-x86-64-gfortran-psmp	PSI, Merlin5	OK	#21fbd8 (0)	correct: 3106 / 3106; 65min	
Linux-x86-64-intel-popt (v18.0.5.274)	PSI, Merlin5	OK	#21fbd8 (0)	correct: 3106 / 3106; 99min	
Linux-x86-64-intel-psmp (v18.0.5.274)	PSI, Merlin5	FAILED	#21fbd8 (0)	correct: 3105 / 3106; failed: 1; 124min	b7143ef (-1)
Linux_gnu-4.9.2_mkl-11.2.1_sopt	UZH, opt5	OUTDATED	b7143ef (-1)	correct: 3031 / 3031; 6min	b7143ef (-1)
Linux_gnu-4.9.2_mkl-11.2.1_popt	UZH, opt5	OUTDATED	b7143ef (-1)	correct: 3090 / 3090; 8min	b7143ef (-1)
Linux_gnu-4.9.2_mkl-11.2.1_ssmp	UZH, opt5	OUTDATED	b7143ef (-1)	correct: 3031 / 3031; 6min	b7143ef (-1)
Linux_gnu-4.9.2_mkl-11.2.1_psmp	UZH, opt5	OUTDATED	b7143ef (-1)	correct: 3090 / 3090; 11min	b7143ef (-1)
CRAY_XC50-gfortran_ssmp.ssmp	CSCS, PizDaint	FAILED	#21fbd8 (0)	correct: 3013 / 3015; wrong: 2; 120min	ae6977f (-8)
CRAY_XC50-gfortran_mc.psmp	CSCS, PizDaint	FAILED	#21fbd8 (0)	correct: 1257 / 3015; wrong: 1; failed: 1; 62min	ae6977f (-8)
Current Toolchain (pdbg)	GCP	OK	#21fbd8 (0)	correct: 3110 / 3110; 17min	
Current Toolchain (popt)	GCP	OK	#21fbd8 (0)	correct: 3110 / 3110; 11min	
Current Toolchain (psmp)	GCP	OK	#21fbd8 (0)	correct: 3112 / 3112; 15min	
Current Toolchain (sdbg)	GCP	OK	#21fbd8 (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 : mpiran
[...]
------------------------------ 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

Input-Debugging & Output-Capturing

```
$ 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`, ...)

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

Structure-only: Supported formats

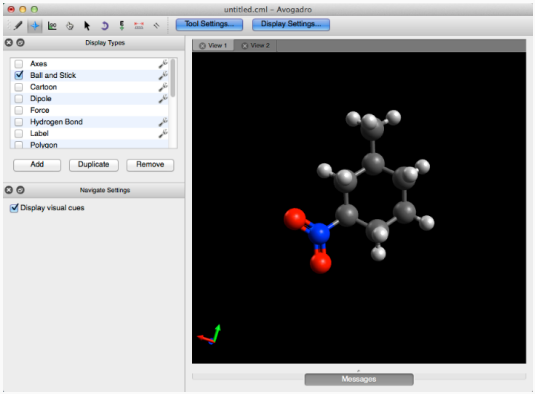
Use your favorite molecular structure editor.

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

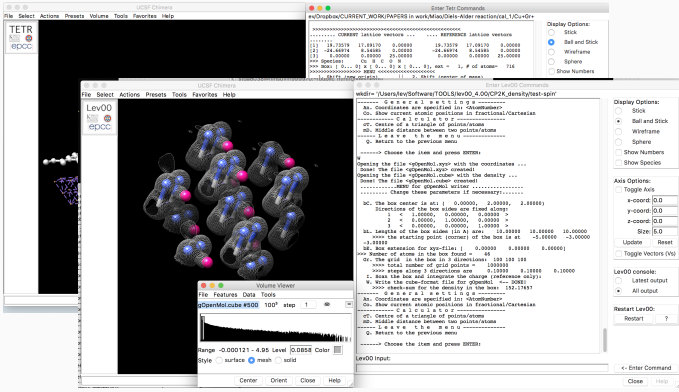
- CP2K Plugin for Avogadro 1.x: `https://github.com/brhr-iwao/libavogadro1cp2k`
- CP2K Plugin for Avogadro 2.x: `https://github.com/svedruziclab/avogadrolibs-cp2k`

Full Input generation: Chimera

UCSF CHIMERA
an Extensible Molecular Modeling System

Chimera

- 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

Full Input generation: PYCP2K

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

PYCP2K

```python
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"
```

Full Input generation: Python ASE

- 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

The Atomic Simulation Environment

```python
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.

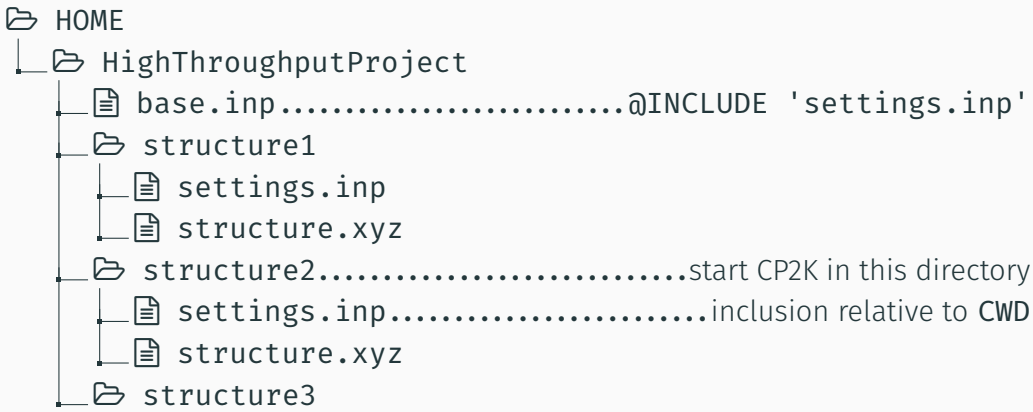
`@SET VAR value`
: (re-)define a variable

`${VAR}` or **`$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`.

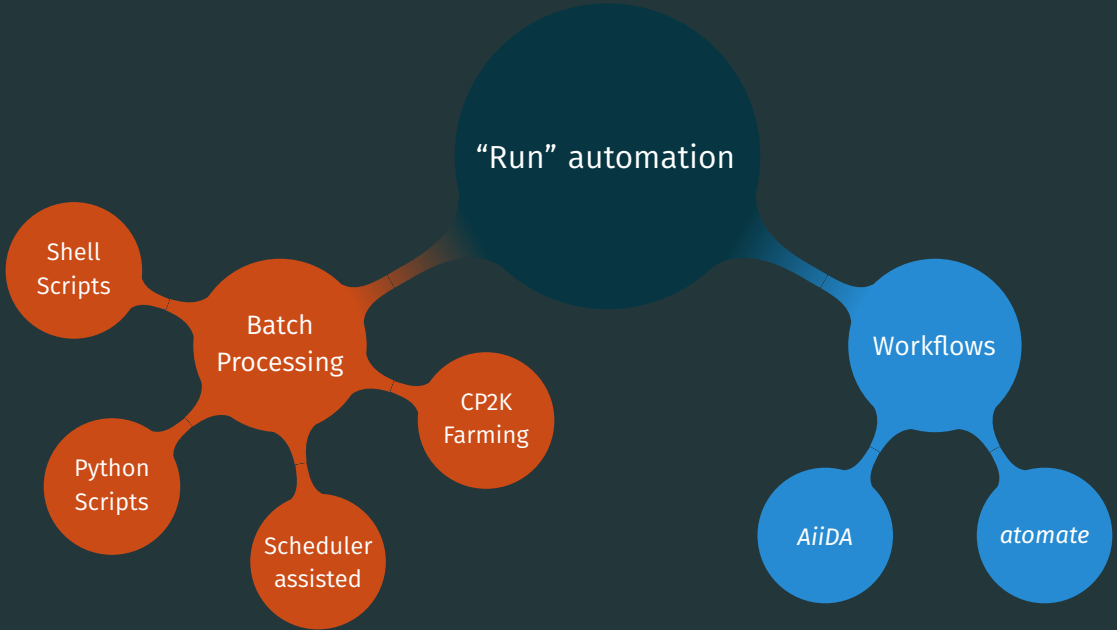
`@PRINT …`
: Print the given text while pre-processing

CP2K Internal Input Preprocessor: Example



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

“Run” Automation


“Run” automation
Shell Scripts
Batch Processing
CP2K Farming
Python Scripts
Scheduler assisted
Workflows
AiiDA
atomate

Batch Processing: CP2K Farming

```
&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

Workflows: AiiDA

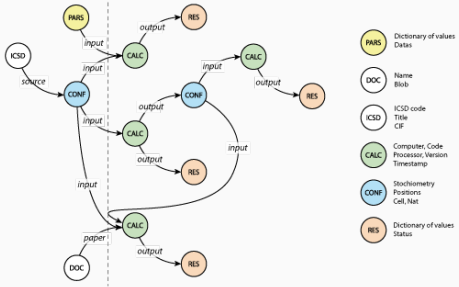
Automated
Interactive Infrastructure
and Database

- 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

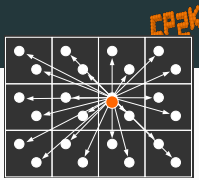


AiiDA Data Provenance Graph

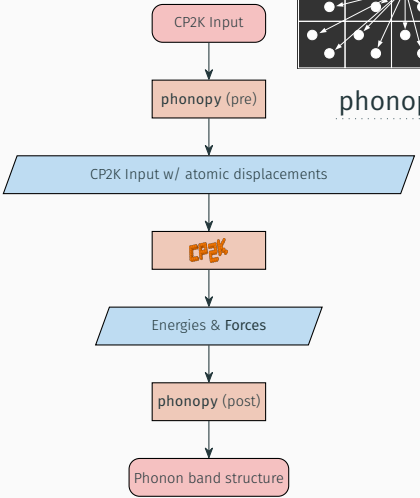
Integration: Phonopy, PyRETIS, i-PI

Phonopy: Phonon calculation with CP2K

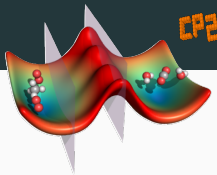
- Python based
- File-based interface:
  parses & generates code inputs
- Only needs equilibrated and
  symmetrized crystal structure input
- Uses a supercell approach
- Can also generate:
  DOS, pDOS, Thermal properties



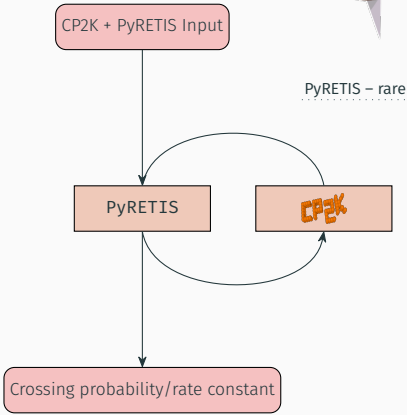
phonopy



PyRETIS: Transition Interface Sampling



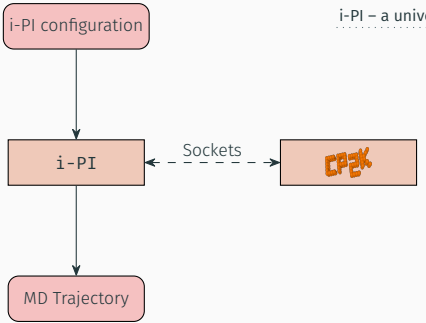
- Python based
- Transition Interface Sampling (TIS) and Replica Exchange Transition Interface Sampling (RETIS)
- Can use CP2K as integrator for MD steps



i-PI: a universal force engine

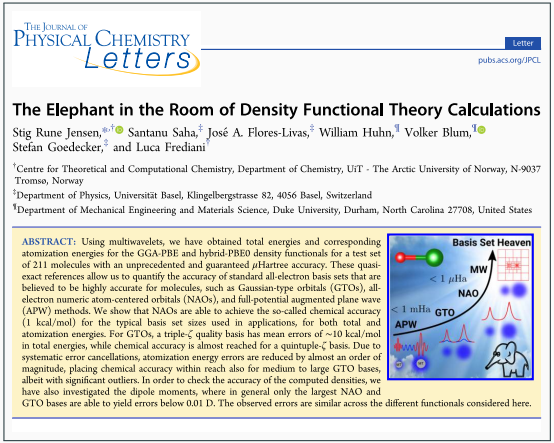
i-PI – a universal force engine

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



Basis Set Verification

Basis Set Verification: The challenge



THE JOURNAL OF PHYSICAL CHEMISTRY Letters

Letter

pubs.acs.org/JPCL

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

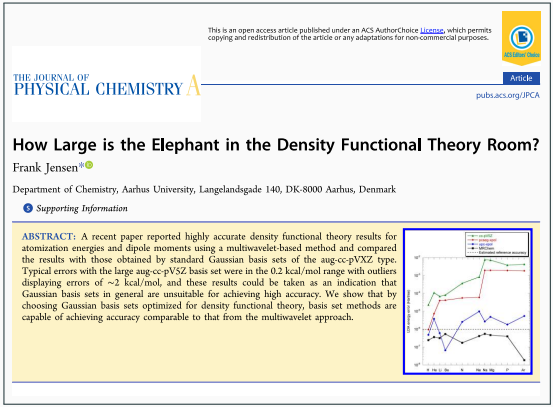
ABSTRACT: Using multiwavelets, we have obtained total energies and corresponding atomization energies for the GGA-PBE and hybrid-PBE0 density functionals for a test set of 211 molecules with an unprecedented and guaranteed μHartree accuracy. These quasi-exact references allow us to quantify the accuracy 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?

Stig Rune Jensen et al. In: *J. Phys. Chem. Lett.* 8.7 (Apr. 6, 2017). 00003, pp. 1449–1457. DOI: 10.1021/acs.jpclett.7b00255

Basis Set Verification: The challenge



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-pVSZ 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.

Frank Jensen. In: *J. Phys. Chem. A* 121.32 (Aug. 17, 2017). 00002, pp. 6104–6107.
DOI: 10.1021/acs.jpca.7b04760

"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..."

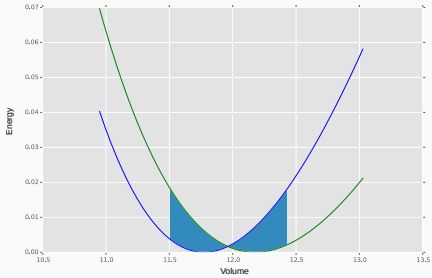
→Not necessarily, just use the right one.

Basis Set Verification: The Deltatest

- Solid-state benchmark[2]
- Measure: Difference between two Volume/Energy-curves
- 40+ “Methods”, 71 Elements (H-Rn, elemental crystals)
- DFT, PBE Functional
- All-Electron calculations as reference

Δ-Gauge[3]

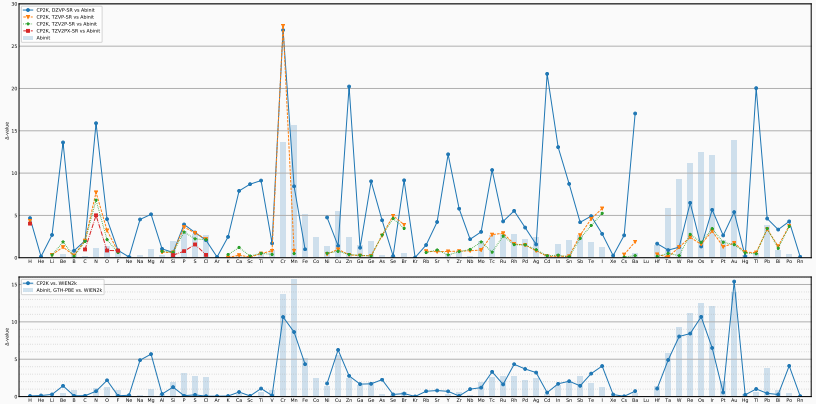
$$\Delta_i(a,b) = \sqrt{\frac{\int_{0.94V_{0,i}}^{1.06V_{0,i}} \left(E_{b,i}(V) - E_{a,i}(V)\right)^2 \,\mathrm{d}V}{0.12V_{0,i}}}$$

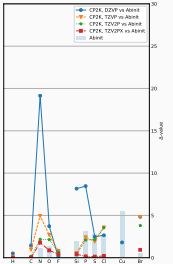


[2] Kurt Lejaeghere et al. In: *Science* 351.6280 (Mar. 25, 2016). 00079, aad3000. DOI: 10.1126/science.aad3000

[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

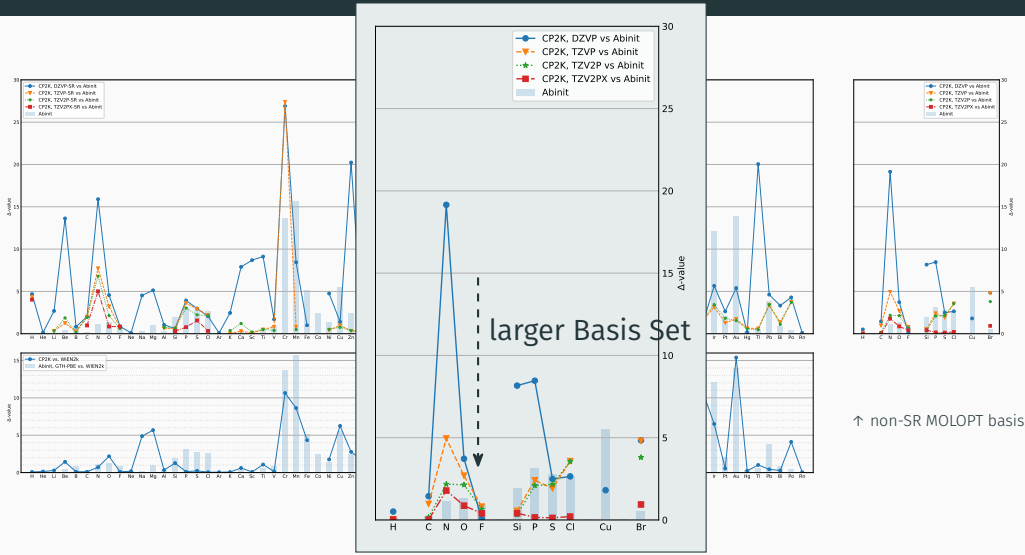
Basis Set Verification: Deltatest results for CP2K's MOLOPT Basis Set





↑ non-SR MOLOPT basis sets

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



↑ non-SR MOLOPT basis sets

Basis Set Verification: CP2K MOLOPT Deltatest Summary & Outlook

- 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 Peintinger[4] basis set in progress
- More benchmarks coming

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

Performance Optimisation

CP2K Timing Example

SUBROUTINE name contains method and step descriptors:

- **pw** Planewave
- **fft** Fast Fourier Transformation
- **mp** Message Passing (MPI)
- **qs** Quickstep
- **scf** Self-consistent field

ASD measure for how deeply nested a function is

SELF TIME time spent in routine and non seperately timed subroutines

```
-------------------------------------------------------------------------------
-                                                                             -
-                                T I M I N G                                  -
-                                                                             -
-------------------------------------------------------------------------------
SUBROUTINE                       CALLS  ASD         SELF TIME        TOTAL TIME
                               MAXIMUM       AVERAGE  MAXIMUM  AVERAGE  MAXIMUM
CP2K                                 1  1.0    0.847    0.890 2709.628 2709.629
qs_energies                          1  2.0    0.000    0.000 2708.215 2708.215
scf_env_do_scf                       1  3.0    0.001    0.002 2705.607 2705.607
scf_env_do_scf_inner_loop          630  4.0    0.038    0.656 2607.232 2607.239
qs_ks_update_qs_env                651  5.0    0.005    0.006 1789.360 1789.383
rebuild_ks_matrix                  630  6.0    0.002    0.002 1788.729 1788.736
qs_ks_build_kohn_sham_matrix       630  7.0    0.080    0.088 1788.728 1788.734
pw_transfer                      18285  9.3    1.136    1.356 1410.183 1433.258
fft_wrap_pw1pw2                  18285 10.3    0.171    0.207 1409.047 1432.022
fft_wrap_pw1pw2_400               9875 11.7  148.093  160.013 1355.429 1377.554
qs_vxc_create                      630  8.0    0.010    0.012 1048.606 1049.428
fft3d_ps                         18285 12.3  615.774  642.063 1002.533 1028.016
qs_rho_update_rho                  631  5.0    0.005    0.005  896.809  896.810
calculate_rho_elec                1262  6.0   19.360   65.700  896.804  896.806
density_rs2pw                     1262  7.0    0.128    0.140  815.787  829.917
xc_rho_set_and_dset_create         630 10.0   19.980   21.160  721.769  777.824
rs_pw_transfer                   10096  8.7    0.142    0.176  558.560  573.389
xc_vxc_pw_create                   210  9.0    9.400   10.387  531.224  532.044
xc_exc_calc                        420  9.0    1.469    1.582  517.372  517.373
pw_poisson_solve                   630  8.0   19.270   21.048  450.545  450.547
rs_pw_transfer_RS2PW_400          1264  9.0  185.950  197.914  263.644  280.272
[...]
mp_alltoall_z22v                 18285 14.3  173.031  215.341  173.031  215.341
mp_waitany                      141448 10.7  140.483  179.715  140.483  179.715
pw_scatter_p                      8402 13.3  168.017  174.224  168.017  174.224
```

CP2K Timing Guidelines

- Check that I/O routines are < 50% of total runtime

→ Remember Amdahl's law: scaling flattens eventually

→ Do you really need to write so much/often?

→ Are you running in the right directory?

- Compare Average and Maximum values

→ large difference could mean that nodes are waiting for single rank to finish

- Check settings for respective sections

→ Are you using the right algorithms?

General Guidelines

- Optimization starts with you: Biggest gains by proper setup
  - Cell size
  - SCF settings, preconditioner
  - Choice of basis set
  - ADMM
- No universal recipe, check scaling of *your* system
  - Run a small number of `MD` or `GEO_OPT` steps
  - Turn off outer-SCF, keep inner-SCF fixed
- When compiling yourself:
  - Use vendor-supplied BLAS, LAPACK, FFTW3 libraries
  - Build and use libxsmm, ELPA
  - CUDA support available, improvements are coming

Getting Help

`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!