CP2K: Automation, Scripting, Testing

Tiziano Müller
tiziano.mueller@chem.uzh.ch
CP2K Workshop @ UGent, 11.-13. March 2019
Dept. of Chemistry, UZH
Outline

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
Performance Optimisation
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 pssmp"
```

- 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

$ 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Allowed values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blas</td>
<td>[openblas] openblas, mkl, accelerate</td>
<td>Enable the use of OpenBlas/MKL/Accelerate</td>
</tr>
<tr>
<td>elpa</td>
<td>[off] True, False</td>
<td>Enable optimised</td>
</tr>
<tr>
<td>libxc</td>
<td>[on] True, False</td>
<td>Support additional functionals via libxc</td>
</tr>
<tr>
<td>mpi</td>
<td>[on] True, False</td>
<td>Enable MPI support</td>
</tr>
<tr>
<td>openmp</td>
<td>[off] True, False</td>
<td>Enable OpenMP support</td>
</tr>
<tr>
<td>pexsi</td>
<td>[off] True, False</td>
<td>Enable the alternative PEXSI method for density matrix evaluation</td>
</tr>
<tr>
<td>plumed</td>
<td>[off] True, False</td>
<td>Enable PLUMED support</td>
</tr>
<tr>
<td>smm</td>
<td>[libxsmm] libxsmm, libsmm</td>
<td>Library for small matrix blas multiplications</td>
</tr>
</tbody>
</table>
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-byjtwnyhrqqmzezvpy3zwiccccexshd

$ ls .../cp2k-6.1-byjtwnyhrqqmzezvpy3zwiccccexshd/.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

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 : parallel
Skipping QS/regtest-cdft-hirshfeld-2 : missing required feature : mpiranks>1
[...]
--------------------------------- 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
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

- 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

- 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

```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

#==================== 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'
FORCE_EVAL.PRINT.FORCES.Section_parameters = 'ON'
DFT.Basis_set_file_name = 'BASIS_SET'
DFT.Potential_file_name = 'GTH_POTENTIALS'
DFT.QS.Eps_default = 1.0E-10
DFT.MGRID.Ngrids = 4
DFT.MGRID.Cutoff = 300
DFT.MGRID.Rel_cutoff = 60
DFT.XC.XC_FUNCTIONAL.Section_parameters = 'PADE'
SCF.Scf_guess = 'ATOMIC'
SCF.Eps_scf = 1.0E-7
SCF.Max_scf = 300
SCF.DIAGONALIZATION.Section_parameters = 'ON'
SCF.DIAGONALIZATION.Algorithm = 'STANDARD'
SCF.MIXING.Section_parameters = 'T'
SCF.MIXING.Method = 'BROYDEN_MIXING'
SCF.MIXING.Alpha = 0.4
SCF.MIXING.Nbroyden = 8
KIND = SUBSYS.KIND_add('Si')  # Section_parameters can be provided as argument.
KIND.Basis_set = 'DZVP-GTH-PADE'
KIND.Potential = 'GTH-PADE-q4'
calc.create_cell(SUBSYS, lattice)
calc.create_coord(SUBSYS, lattice)

#============ Run the simulation or just write the input file ================
calc.write_input_file()
calc.run()
```
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

```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 = r""

&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
vol0 = 4 * 0.91615977036 # theoretical minimum
a0 = vol0 ** (1 / 3)
a = bulk('Ar', 'fcc', a = a0)
cell0 = a.get_cell()
a.calc = calc

a.set_cell(np.dot(a.cell,
    [[1.02, 0, 0.03], [0, 0.99, -0.02], [0.1, -0.01, 1.03]],
scale_atoms=True)
a *= (1, 2, 3)
cell0 *= np.array([1, 2, 3])[:, np.newaxis]
a.rattle()

# Verify analytical stress tensor against numerical value
s_analytical = a.get_stress()
s_numerical = a.calc.calculate_numerical_stress(a, 1e-5)
s_p_err = 100 * (s_numerical - s_analytical) / s_numerical
print("Analytical stress:")
print(s_analytical)
print("Numerical stress:")
print(s_numerical)
print("Percent error in stress:")
print(s_p_err)

# Minimize unit cell
opt = MDMin(UnitCellFilter(a), dt =0.01)
opt.run(fmax=1e-3)

# Verify minimized unit cell using Niggli tensors
g_minimized = np.dot(a.cell, a.cell.T)
g_theory = np.dot(cell0, cell0.T)
g_p_err = 100 * (g_minimized - g_theory) / g_theory
print("Minimized Niggli tensor:")
print(g_minimized)
print("Theoretical Niggli tensor:")
print(g_theory)
print("Percent error in Niggli tensor:")
print(g_p_err)"
```
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

Batch Processing

Shell Scripts
Python Scripts
Scheduler assisted
CP2K Farming

Workflows
AiiDA
atomate
Batch Processing: CP2K Farming

```plaintext
&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
```

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

- 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
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: 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
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

- 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

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?
Basis Set Verification: The challenge

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

\[ \Delta_{i}(a,b) = \sqrt{\int_{0.94V_{0,i}}^{1.06V_{0,i}} (E_{b,i}(V) - E_{a,i}(V))^2 \, dV} \]

\[ \Delta_{i}(a,b) = \frac{\int_{0.94V_{0,i}}^{1.06V_{0,i}} (E_{b,i}(V) - E_{a,i}(V))^2 \, dV}{0.12V_{0,i}} \]

---


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

CP2K, DZVP-SR vs Abinit
CP2K, TZVP-SR vs Abinit
CP2K, TZV2P-SR vs Abinit
CP2K, TZV2PX-SR vs Abinit

Abinit

H C N O F Si P S Cl Cu Br

↑ 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](http://www.materialscloud.org)
Testing of All-Electron Peintinger\(^4\) basis set in progress
More benchmarks coming

---

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

<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>CALLS</th>
<th>ASD MAXIMUM</th>
<th>ASD AVERAGE</th>
<th>SELF TIME MAXIMUM</th>
<th>SELF TIME AVERAGE</th>
<th>TOTAL TIME MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP2K</td>
<td>1</td>
<td>1.0</td>
<td>0.847</td>
<td>0.890</td>
<td>2709.628</td>
<td>2709.629</td>
</tr>
<tr>
<td>qs_energies</td>
<td>1</td>
<td>2.0</td>
<td>0.000</td>
<td>0.000</td>
<td>2708.215</td>
<td>2708.215</td>
</tr>
<tr>
<td>scf_env_do_scf</td>
<td>1</td>
<td>3.0</td>
<td>0.001</td>
<td>0.002</td>
<td>2705.607</td>
<td>2705.607</td>
</tr>
<tr>
<td>scf_env_do_scf_inner_loop</td>
<td>630</td>
<td>4.0</td>
<td>0.038</td>
<td>0.056</td>
<td>2607.232</td>
<td>2607.239</td>
</tr>
<tr>
<td>qs_update_qenv</td>
<td>651</td>
<td>5.0</td>
<td>0.005</td>
<td>0.006</td>
<td>1789.360</td>
<td>1789.383</td>
</tr>
<tr>
<td>rebuild_ks_matrix</td>
<td>630</td>
<td>6.0</td>
<td>0.002</td>
<td>0.002</td>
<td>1788.729</td>
<td>1788.736</td>
</tr>
<tr>
<td>qs_build_kohn_sham_matrix</td>
<td>630</td>
<td>7.0</td>
<td>0.000</td>
<td>0.000</td>
<td>1788.728</td>
<td>1788.734</td>
</tr>
<tr>
<td>pw_transfer</td>
<td>18285</td>
<td>9.3</td>
<td>0.038</td>
<td>0.038</td>
<td>1410.183</td>
<td>1433.258</td>
</tr>
<tr>
<td>fft_wrap_pw1pw2</td>
<td>18285</td>
<td>10.3</td>
<td>0.000</td>
<td>0.000</td>
<td>1409.047</td>
<td>1432.022</td>
</tr>
<tr>
<td>fft_wrap_pw1pw2_400</td>
<td>9875</td>
<td>11.7</td>
<td>148.093</td>
<td>160.013</td>
<td>1355.429</td>
<td>1377.554</td>
</tr>
<tr>
<td>qs_vxc_create</td>
<td>630</td>
<td>8.0</td>
<td>0.010</td>
<td>0.012</td>
<td>1048.606</td>
<td>1049.428</td>
</tr>
<tr>
<td>fft3d_ps</td>
<td>18285</td>
<td>12.3</td>
<td>615.774</td>
<td>642.063</td>
<td>1002.533</td>
<td>1028.016</td>
</tr>
<tr>
<td>qs_rho_update_rho</td>
<td>631</td>
<td>5.0</td>
<td>0.005</td>
<td>0.005</td>
<td>896.809</td>
<td>896.810</td>
</tr>
<tr>
<td>calculate_rho_elec</td>
<td>1266</td>
<td>6.0</td>
<td>59.300</td>
<td>65.700</td>
<td>896.804</td>
<td>896.806</td>
</tr>
<tr>
<td>density_rs2pw</td>
<td>1266</td>
<td>7.0</td>
<td>0.128</td>
<td>0.140</td>
<td>815.787</td>
<td>829.917</td>
</tr>
<tr>
<td>xc_rho_set_and_dset_create</td>
<td>630</td>
<td>10.0</td>
<td>19.980</td>
<td>21.160</td>
<td>721.769</td>
<td>777.824</td>
</tr>
<tr>
<td>rs_pw_transfer</td>
<td>10096</td>
<td>8.7</td>
<td>0.142</td>
<td>0.176</td>
<td>558.560</td>
<td>573.389</td>
</tr>
<tr>
<td>xc_vxc_pw_create</td>
<td>230</td>
<td>9.0</td>
<td>9.400</td>
<td>10.387</td>
<td>531.224</td>
<td>532.044</td>
</tr>
<tr>
<td>xc_exc_calc</td>
<td>429</td>
<td>9.0</td>
<td>1.469</td>
<td>1.582</td>
<td>517.372</td>
<td>517.373</td>
</tr>
<tr>
<td>pw_poisson_solve</td>
<td>630</td>
<td>8.0</td>
<td>19.270</td>
<td>21.048</td>
<td>450.545</td>
<td>450.547</td>
</tr>
<tr>
<td>rs_pw_transfer_RS2PW_400</td>
<td>1264</td>
<td>9.0</td>
<td>185.950</td>
<td>197.914</td>
<td>263.644</td>
<td>280.272</td>
</tr>
<tr>
<td>mp_alltoall_z22v</td>
<td>18285</td>
<td>14.3</td>
<td>173.031</td>
<td>215.341</td>
<td>173.031</td>
<td>173.031</td>
</tr>
<tr>
<td>mp_waitany</td>
<td>141448</td>
<td>10.7</td>
<td>140.483</td>
<td>179.715</td>
<td>140.483</td>
<td>179.715</td>
</tr>
<tr>
<td>pw_scatter_p</td>
<td>8402</td>
<td>13.3</td>
<td>168.017</td>
<td>174.224</td>
<td>168.017</td>
<td>174.224</td>
</tr>
</tbody>
</table>
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-SOURCE>/tests ................................. Minimal Working Examples

https://groups.google.com/group/cp2k ........ Google Group/Forum

https://github.com/cp2k/cp2k/issues ........... Issue Tracker
Thank you!