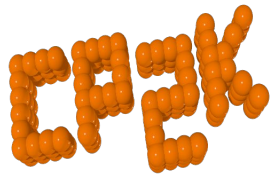
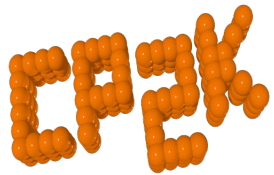
CP2K Developers Meeting

May 5th 2022, 14:00-16:00 CET

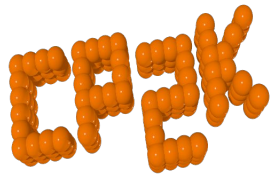


CP2K Developers Meeting

1. Introduction
2. Gradients of RI-MP2-based double-hybrid functionals (Frederick Stein)
3. DBCSR with OpenCL backend (Hans Pabst)
4. Submatrix+xTB on Perlmutter (Thomas Kühne)
5. ERI-Acceleration with FPGAs (Robert Schade)
6. UK based update ERI and ML / python (Matt Watkins (if available))
7. CP2K Verification Project (Thomas Kühne/Matthias Krack)
8. CP2K Release (all)
9. CP2K-related Events (all)



Introduction



Gradients of RI-MP2-based double-hybrid functionals (1/2)

Latest developments:

- Extension to metaGGA functionals (also for forces with Harris functionals)
- Implementation of more options to the CPKS solver (*)
  - Does not help with convergence
  - much simpler than old solver, more easily extendable
- Specialization of the grid REF backend routines to single Gaussians (*)
  - No significant improvement
- Removal of some unnecessary FFTs within the collocation routines (*)
  - No significant improvement
- Removal of tau_g in qs_rho_type to save memory (*)

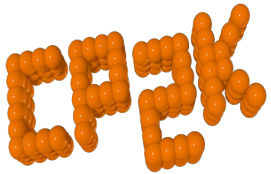
(*) Not pushed to trunk, shall I?

Gradients of RI-MP2-based double-hybrid functionals (2/2)

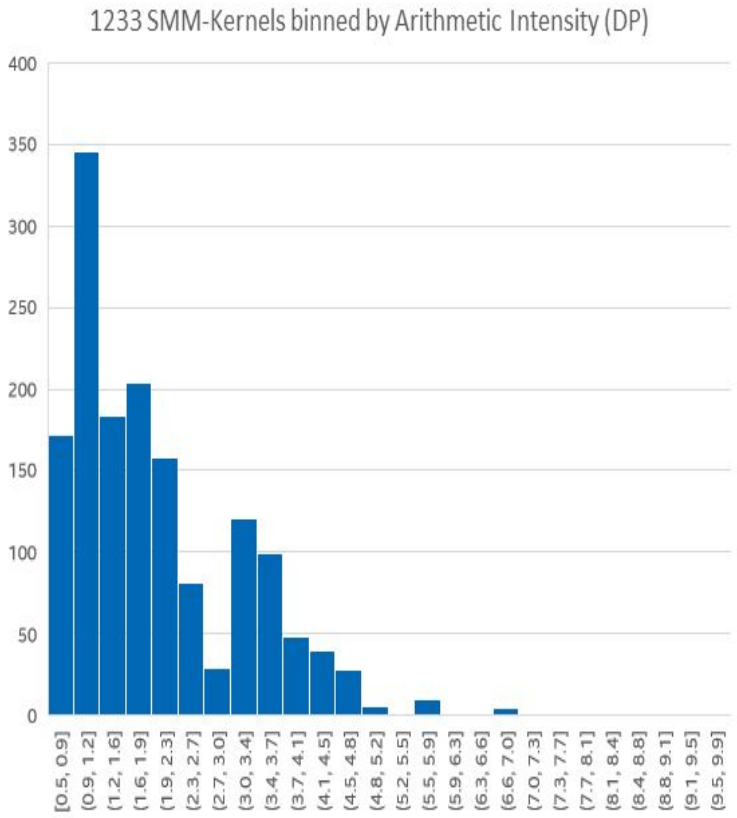
- Communication in MP2 energy calculations
  - Communication within MP2 very costly
  - Implementation uses partial replication of matrices and blocking to reduce communication
  - Replication not applied to open-shell systems
  - Memory management does not include requirements of the gradient code
  - Automatically determined block size is 1
  - Block size > 1 leads to wrong gradients (fixed)
  -
  - Ideas:
    - Extend to open-shell systems
    - Revise memory management
    - Reasonably automatically determined block sizes

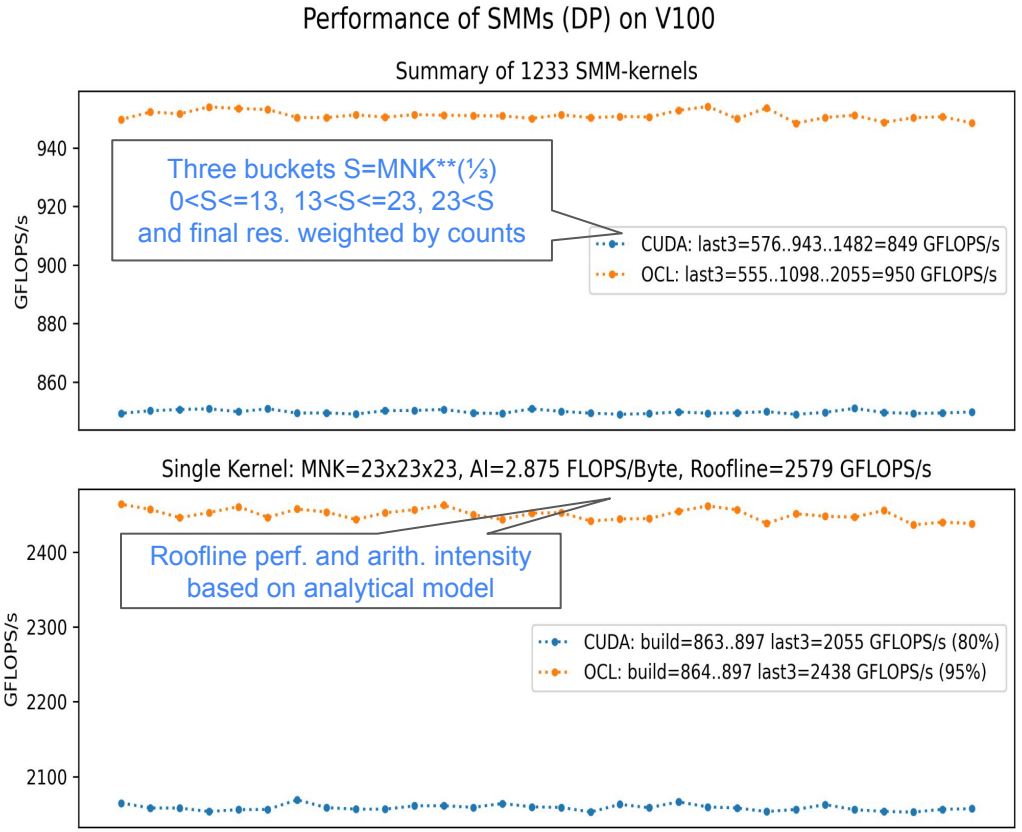
DBCSR with OpenCL backend

- **OpenCL**: industry standard (Khronos group) consisting of runtime API and C99/C11 based kernel language. Relatively well supported with several vendors tightly/timely following specification, e.g., Intel and Nvidia (both at OpenCL 3.0 aka latest).
  - In contrast to CUDA (or SYCL/DPC++), OpenCL has separate source code between host and device, i.e., two-phase compilation (or JIT) is exposed. OpenCL supports C++/templates since v3.0 (extension).
  - Blends well with SYCL/DPC++ (to get rid of verbose OpenCL C runtime aka “glue code”).
- **DBCSR**: lightweight “ACC interface” is tightly resembling CUDA (incl. misconceptions like “thread-local active device” which is de-facto a global variable)
  - OpenCL backend is modeling thread-local active device, and has additional code to synchronize a whole device (need to know all streams pointing to a device).
- **Proofpoint**: match CUDA and OpenCL performance on Nvidia GPUs.
  - Auto-tuned params for P100+V100, and hand-tuned defaults (“untuned”) for other GPUs. (Ideal case: auto-tuning has no effect, i.e., good perf. due to other measures)



DBCSR with OpenCL backend

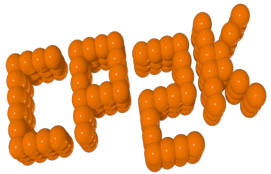




DBCSR with OpenCL backend

- **OpenCL performance: positive surprise across all tested vendors.**
  - Nvidia: only few hitches: PTX inline assembly to access atomic instructions (straight-forward), MPS is CUDA-only (workaround: disable process-exclusive mode per nvidia-smi).
  - AMD Mi100: FP-atomics seem not available (in contrast to HIP), some instability.
  - Intel: dGPU exercise makes Intel's OpenCL even better, FP-atomics available.
- **Status**: CI-tested on Nvidia GPUs ("production quality")
  - OpenCL backend (BE) hosted in cp2k/dbcsr repository (should move to cp2k/cp2k/offload?).
  - LIBXSMM code registry used for: (1) caching kernels, and (2) loading tuned parameters.
  - Developers can write OpenCL kernels and leverage "glue" code from OpenCL backend.
  - Documented and enabled for CP2K toolchain (can be of course improved).
  - Remember: CUDA/HIP BE bails out for kernels w/o known parameters.

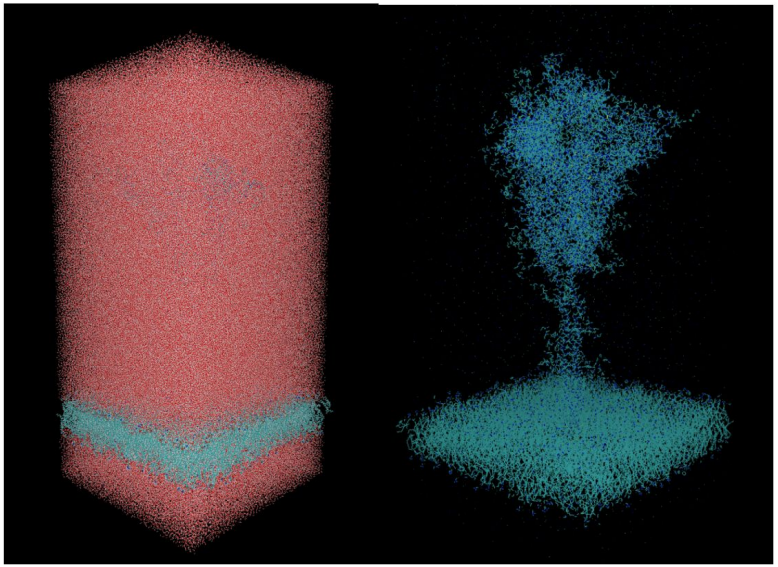
- **Call to Action**: decide about Docker recipes and CI-testing of CUDA (GRID/DBM/DBT) and OpenCL (DBCSR).
  - Consider OpenCL instead of CUDA going forward with GPUs



Submatrix+xTB on Perlmutter

- previous results on JUWELS Booster: https://www.sciencedirect.com/science/article/pii/S0167819122000242
- New results on Perlmutter (NERSC)
  - SARS-CoV-2 Spike protein in aqueous solution
  - approx. 1.7 mio. atoms
  - Wrapp et al., Science, 367(6483), 2020; Casalino et al., Int J High Perform Comput Appl, 35(5), 2021
  - replicated in x- and y-direction
    i.e. 7x7 gives ≈83 mio. atoms

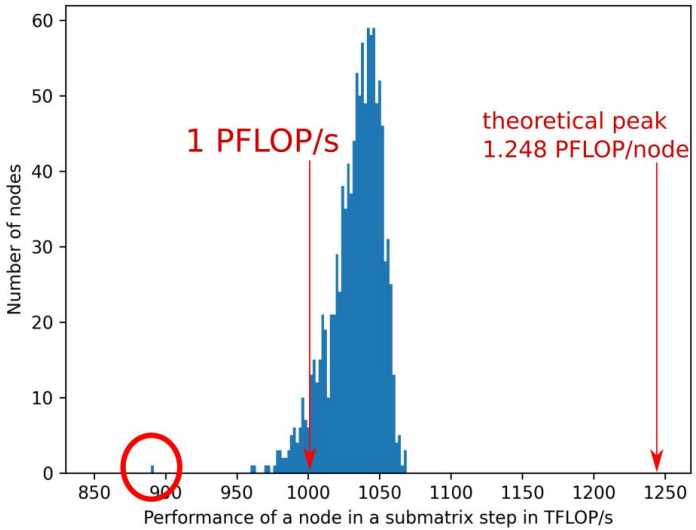
⇒ scalable benchmark system



Spike protein in aqueous solution (right without H and O)

Submatrix+xTB on Perlmutter

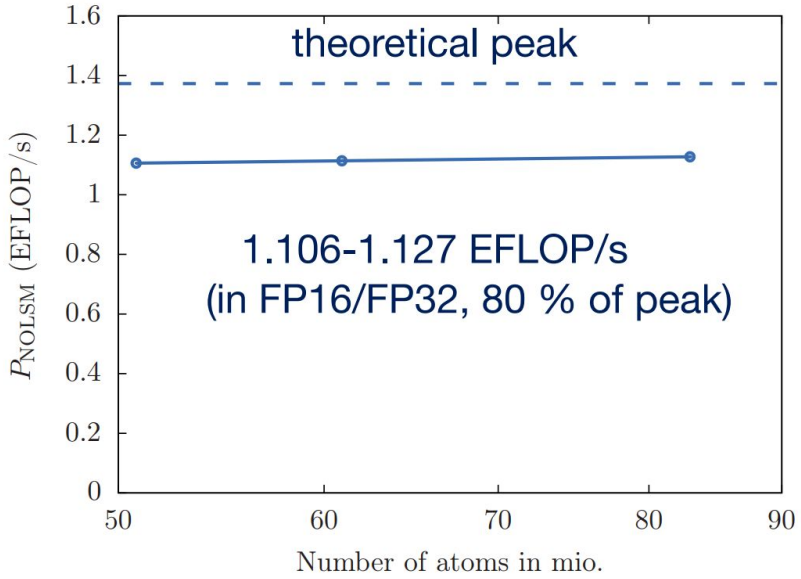
- run with 1100 nodes
- 4400 NVIDIA A100 GPUs
- Three system sizes:
  - 6x5: 51 mio. atoms
  - 6x6: 61 mio. atoms
  - 7x7: 83 mio. atoms
- 0.312 PFLOP/s per NVIDIA A100 theoretical peak (with boost)
- 1.248 PFLOP/s per node theoretical peak



⇒ about 1 PFLOP/s per node

Submatrix+xTB on Perlmutter

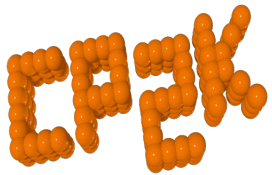
- run with 1100 nodes
- 4400 NVIDIA A100 GPUs
- Three system sizes:
  - 6x5: 51 mio. atoms
  - 6x6: 61 mio. atoms
  - 7x7: 83 mio. atoms



⇒ maximal performance of ≈ 1.1 EFLOP/s in FP16/FP32 reached

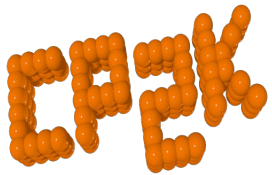
ERI-Acceleration with FPGAs

- Computation of uncontracted electron repulsion integrals on FPGAs
- Current focus on FPGA kernel on Intel Stratix 10
  - DRK approach (Rys-quadrature-based)
  - present state:
    - [fp|dd]: 4.9 GERI/s (in FP32, 2x Xeon 6148 ~0.6 GERI/s with libint)
    - [fd|ff]: 4.7 GERI/s (in FP32, 2x Xeon 6148 ~0.3 GERI/s with libint)
- Next Steps:
  - compression to mediate PCIe-interface
  - libint/libcint-like interface
  - offloading-performance-model for optimal choice of angular momenta to be calculated on FPGAs



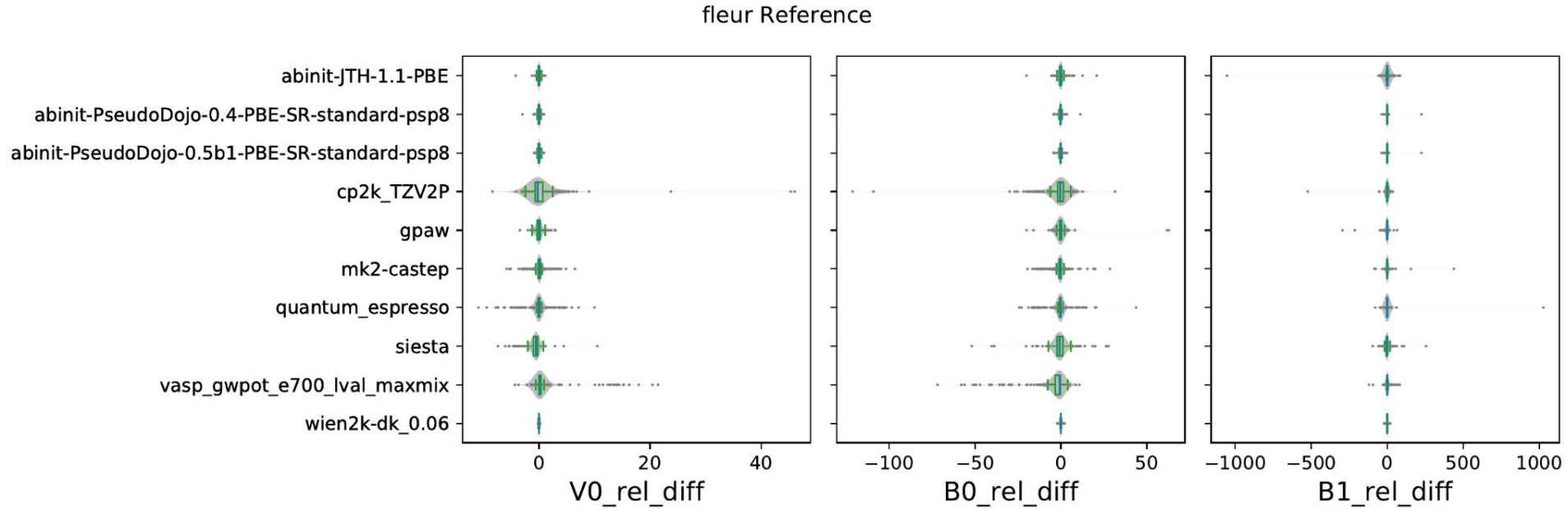
UK based update

- 2 x 3 year postdoc positions:
- GPU ERI calcs - joint position with Rutherford-Appleton Lab (Ian Bush) national lab and Lincoln (Matt Watkins) just filled
  - part of UK exascale project (https://excalibur.ac.uk/)
  - aiming for flexible ERI package for periodic systems
  - CRYSTAL and CP2K targets
  - scoping exercise - what is the best way to complement FPGA work?
- ML / deeper python interface and general (job advert out this week)
  - deeper python interface - f90wrap? https://github.com/jameskermode/f90wrap/blob/master/README.md
  - general usability and connection to ML packages
  - support for periodic k-point code



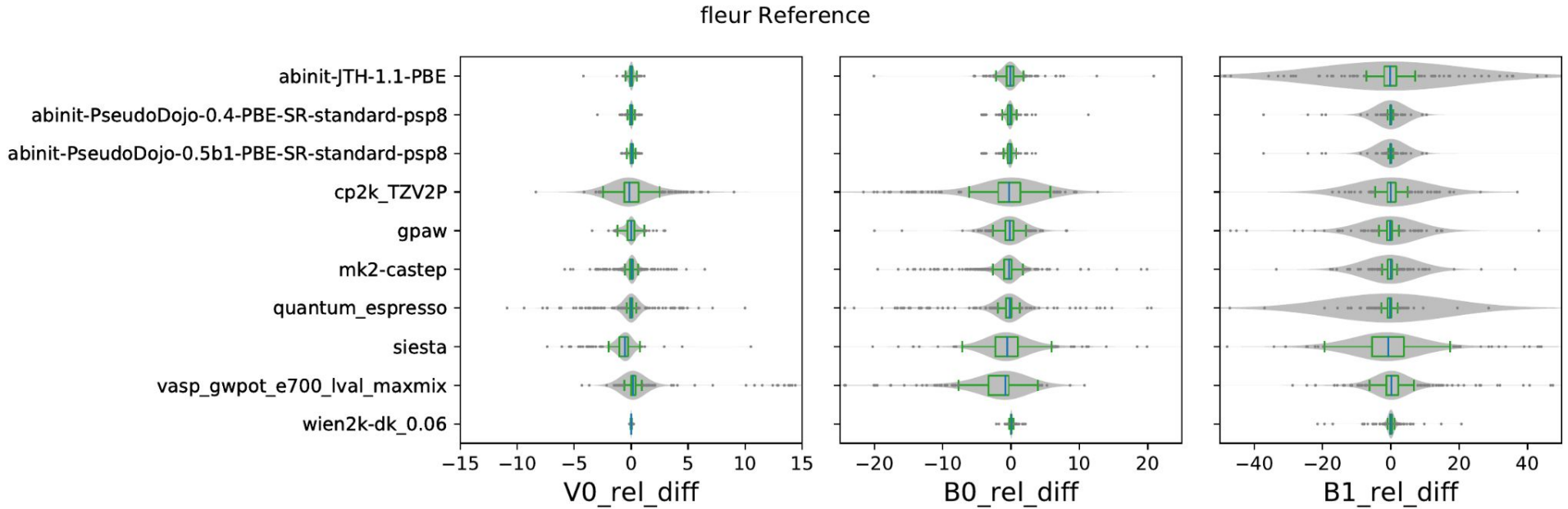
CP2K Verification Project

- Unaries



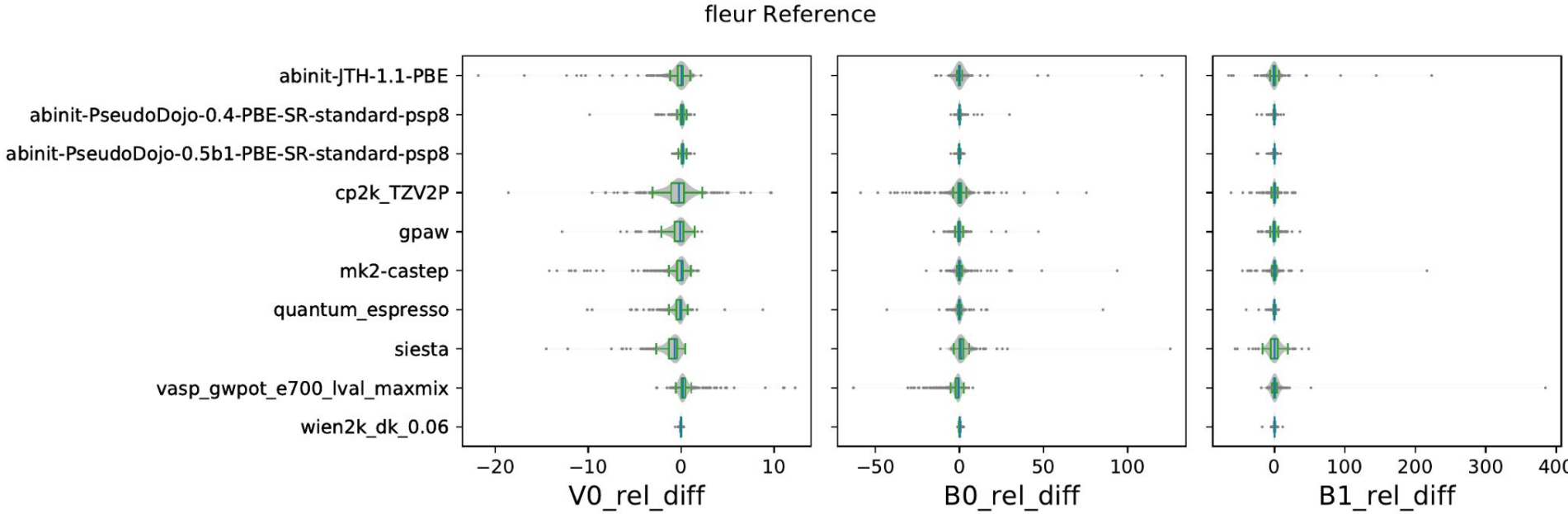
CP2K Verification Project

- Unaries



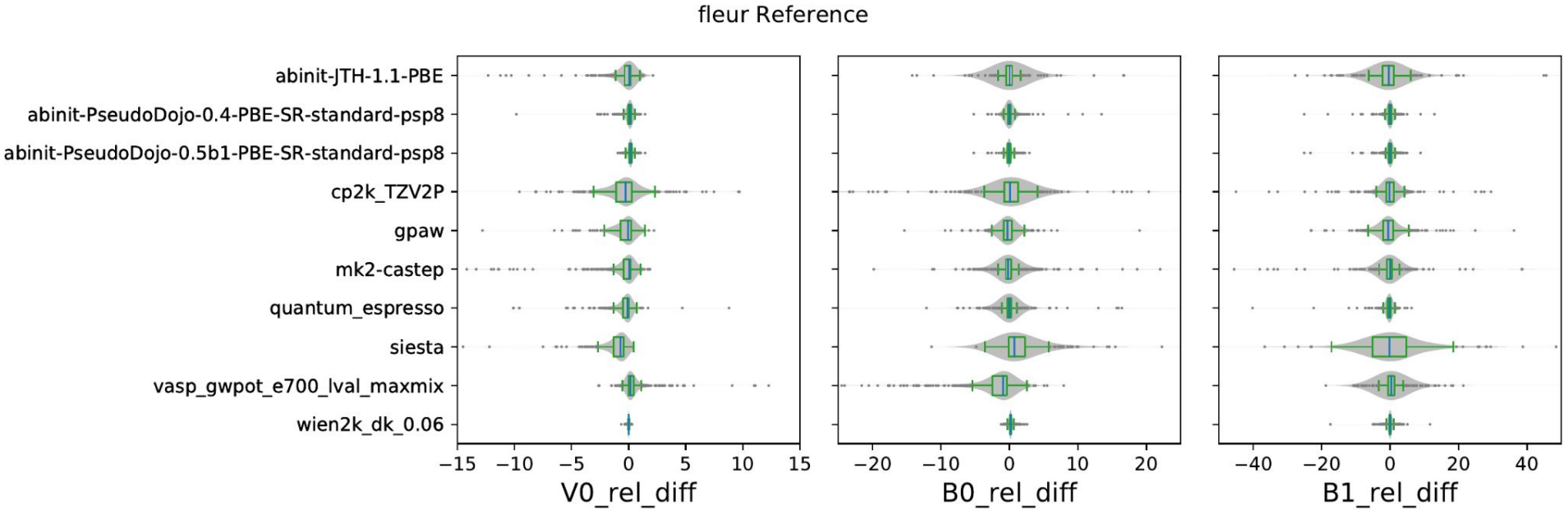
CP2K Verification Project

- Oxides



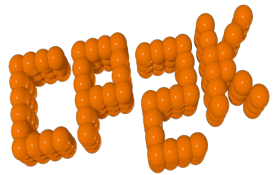
CP2K Verification Project

- Unaries



CP2K Release

- timeline 2-3 months
- feature freeze: end of june
- switching to year.month or year.number naming schema?
- yes, (year).(release number in year)



CP2K-related Events

- Past: Advanced Research in Quantum Chemistry and Solid State Physics with ORCA, CP2K, TRAVIS, and CP-PAW *7 Apr - 8 Apr* (https://events.uni-paderborn.de/event/208/)

  annual UK-CP2K user meeting (towards autumn, Matt Watkins)

- Aspects:
  - events for beginners (Winterschool 2020)
  - events for advanced users (spectroscopy,...)
- workshop on QM/MM every two years (Marcella does talk on CP2K in Lausanne in two weeks)
- possibility to announce as CECAM events
- Suggestion: Classical MD and QM/MM with GROMACS and CP2K
- more information and announcements
- more recordings!