CP2K Developers Meeting

2025/10/27

Topics

CP2K Developers Meeting

- New and Ongoing Developments
- Current Issues with CP2K
- Next CP2K Release
- Planned Events in the Context of CP2K

CP2K Developers Meeting

– announcement routes sufficient?

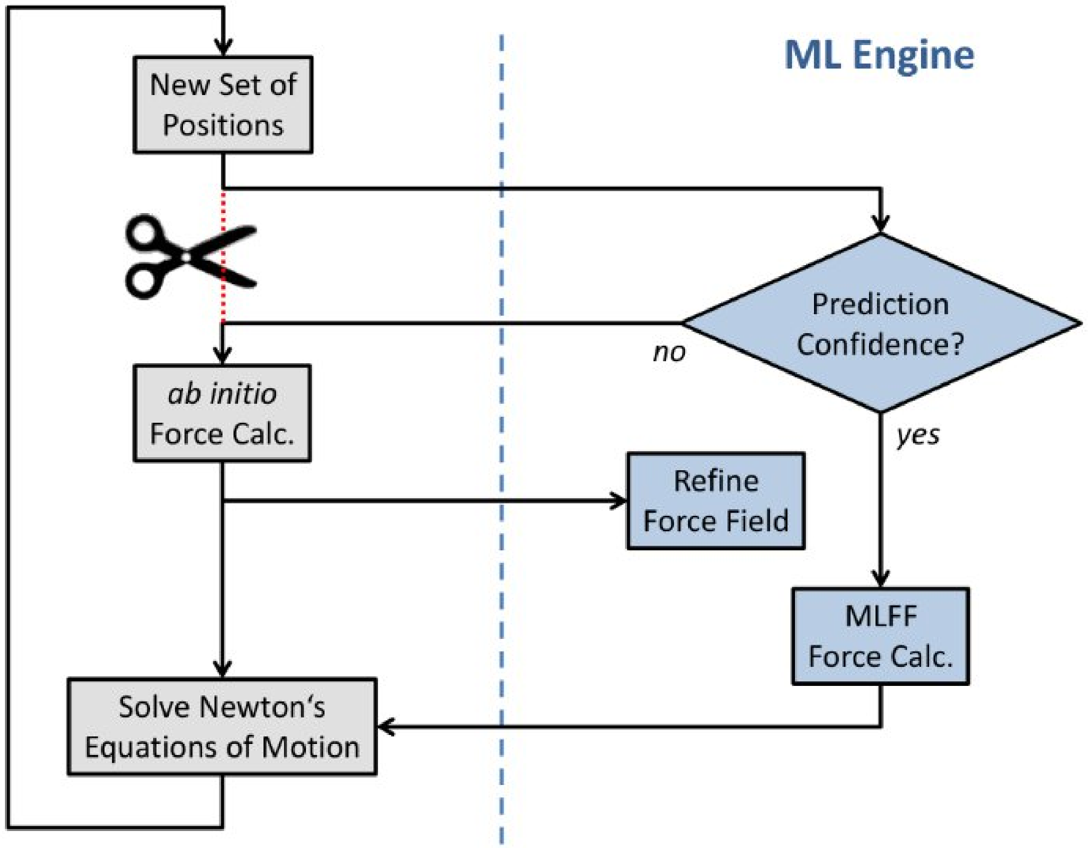
New and Ongoing Developments in CP2K

- Small-cell periodic GW code (R. Pasquier, J. Wilhelm)
- All-electron basis sets for excited states in nanostructures (M. Graml, R. Pasquier, J. Wilhelm)
- Spatial descriptors for excited states (M. Graml, J. Wilhelm)
- Heterogeneous acceleration of the ERI computation in HFX (J. Menzel, R. Schade, X. Wu, C. Plessl)
- MiMiC Interface: How to approach testing? (A. Antalik)
- ....

On-the-Fly Machine Learning

„Is machine learning helpful for me if I just want to run one long trajectory, and there is no pre-trained force field available for my system?"

Basic idea *(as presented in CP2k Dev Meeting March 2024)*:



On-the-Fly Machine Learning

- We need to predict reliable **uncertainties** for the energies / forces
  (neural networks are not optimal for such a task)
- VASP: Bayesian Inference / Gaussian processes → very successfull *(but commercial)*
- We found the FLARE C++ library for sparse Gaussian processes:

> Developed in Boris Kozinsky's group at Harvard university
>
> FLARE
>
> J. Vandermause, Y. Xie, J. S. Lim, C. J. Owen, B. Kozinsky, *Nat. Commun.* **2022**, *13.1*, 5183.
>
> `https://github.com/mir-group/flare`

- **Problem:** FLARE was typically used for 10-50 atoms...
  Very slow and prohibitive memory usage *(1 TiB)* for ≈ 1000 atoms!
- No longer actively developed...
- We revised the algorithms to scale better for larger systems
  (both CPU and memory)

Introducing Prokyon

- We are developing **Prokyon**, a free software C++ library for on-the-fly ML
- License probably GNU L-GPL
- Can be plugged into any QM code *(ORCA, Quantum Espresso, ...)*, but **CP2k is the prime target**
- Will contain several on-the-fly ML methods, the first one will be sparse Gaussian processes based on FLARE
- Ability to save and load ML force fields, so that these can be shared in the community

https://prokyon-lib.org
(not yet online...)

CP2k Integration

Keep it as simple as possible for the user – „black box“:

```
&FORCE_EVAL

   METHOD Quickstep

   &DFT
      ...
   &END DFT

&END FORCE_EVAL
```

```
&FORCE_EVAL

   &OTF_ML
      ...
   &END OTF_ML

   METHOD Quickstep

   &DFT
      ...
   &END DFT

&END FORCE_EVAL
```

- Can be used with any **&FORCE_EVAL** *(Hybrid DFT, MP2, GW, Force Field, ...)*
- Compatible with **&MULTIPLE_FORCE_EVALS**
- Main aim is AIMD, but will also work with **GEO_OPT**, etc.

State of the Project

UNIVERSITÄT PADERBORN

Paderborn Center for Parallel Computing

TECHNISCHE UNIVERSITÄT ILMENAU

- We already have a running alpha version
- First CP2k public commit planned for 12/2025

  → maybe already in CP2k 2026.1?
- Prokyon library published at the same time
- Final parallelization scheme still open question...
- Intermediate solution: Using OpenMP (one node)

Contributing Persons

Paderborn University:

Martin Brehm
Hossam Elgabarty
Omid Shayestehpour
Christian Plessl *(PC2)*
Robert Schade *(PC2)*

TU Ilmenau:

Christian Dreßler
Jonas Hänseroth

Small-cell periodic GW code: Introduction

- The DFT eigenvalues are obtained from the Kohn-Sham equation:

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + v_{\mathrm{ext}}(\mathbf{r}) + v_{\mathrm{H}}(\mathbf{r}) + v_{\mathrm{xc}}(\mathbf{r})\right]\psi_n(\mathbf{r}) = \varepsilon_n\,\psi_n(\mathbf{r})$$

- However the DFT eigenvalues are not the real energies (they are Lagrange multipliers), so that we have: $E_{\mathrm{gap}} = \varepsilon^{\mathrm{KS}}_{n=\mathrm{LUMO}} - \varepsilon^{\mathrm{KS}}_{n=\mathrm{HOMO}} + \Delta_{\mathrm{xc}}$
- Issue: derivative discontinuity wrong in many cases in DFT (in fact, vanishes for LDA/GGA)
- Solution: GW method: $\hat{G}(\varepsilon) = \left(\hat{G}_0^{-1}(\varepsilon) - \hat{\Sigma}(\varepsilon)\right)^{-1} = \left(\varepsilon - \left(\hat{h}_0 + \hat{\Sigma}(\varepsilon)\right) + i\eta\right)^{-1}$
- The quantity $\hat{\Sigma}(\varepsilon)$ is the self-energy, it carries all electron interaction effects.
- GW approximation: $\Sigma_{\mathrm{GW}}(\mathbf{r},\mathbf{r}';\varepsilon) = i\int \mathrm{d}\varepsilon'\; G(\mathbf{r},\mathbf{r}';\varepsilon-\varepsilon')W(\mathbf{r},\mathbf{r}';\varepsilon')$
- Self-consistent equation. In practice, SCF rarely done: G0W0 approximation
- Our code: fully periodic G0W0 calculations in small cells using SOC correction

Our periodic GW small-cell code

$$\psi_{n\mathbf{k}}(\mathbf{r}) = \sum_{\mu} C_{\mu n}(\mathbf{k}) \sum_{\mathbf{R}} e^{i\mathbf{k}\cdot\mathbf{R}} \phi_{\mu}^{\mathbf{R}}(\mathbf{r})$$

$$G_{\mu\nu}(i\tau,\mathbf{k}) = \theta(\tau) \sum_{a}^{\text{empty}} C_{\mu a}(\mathbf{k}) C_{\nu a}^{*}(\mathbf{k}) e^{-(\varepsilon_{a\mathbf{k}}-\varepsilon_{\mathrm{F}})\tau} - \theta(-\tau) \sum_{i}^{\text{occ}} C_{\mu i}(\mathbf{k}) C_{\nu i}^{*}(\mathbf{k}) e^{-(\varepsilon_{i\mathbf{k}}-\varepsilon_{\mathrm{F}})\tau}$$

$$G_{\mu\nu}^{\mathbf{R}}(i\tau) = \int_{\mathrm{BZ}} \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} e^{-i\mathbf{k}\cdot\mathbf{R}} G_{\mu\nu}(i\tau,\mathbf{k})$$

$$\chi_{PQ}^{\mathbf{R}}(i\tau) = \sum_{\lambda\mathbf{R}_1 \nu\mathbf{R}_2} \left[\sum_{\mu} \sum_{\mathbf{S}_1}^{\mathrm{SC}} (\mu\mathbf{R}_1 - \mathbf{S}_1\, \nu\mathbf{R}_2 | P\mathbf{0})\, G_{\lambda\mu}^{\mathbf{S}_1}(-i\tau) \right] \times \left[\sum_{\sigma} \sum_{\mathbf{S}_2}^{\mathrm{SC}} (\lambda\mathbf{R}_1\, \sigma\mathbf{R}_2 - \mathbf{S}_2 | Q\mathbf{R})\, G_{\nu\sigma}^{\mathbf{S}_2}(i\tau) \right]$$

$$\chi_{PQ}(\mathbf{k},i\omega) = \sum_{\mathbf{R}}^{\mathrm{SC}} \int d\tau \cos(\omega\tau) e^{i\mathbf{k}\cdot\mathbf{R}} \chi_{PQ}^{\mathbf{R}}(i\tau)$$

$$\epsilon(\mathbf{k},i\omega) = \mathbf{Id} - \mathbf{V}^{0.5}(\mathbf{k}) \mathbf{M}^{-1}(\mathbf{k}) \chi(\mathbf{k},i\omega) \mathbf{M}^{-1}(\mathbf{k}) \mathbf{V}^{0.5}(\mathbf{k})$$

$$\mathbf{W}^{\mathrm{c}}(\mathbf{k},i\omega) = \mathbf{V}^{0.5}(\mathbf{k}) (\epsilon^{-1}(\mathbf{k},i\omega) - \mathbf{Id}) \mathbf{V}^{0.5}(\mathbf{k})$$

$$W_{PQ}^{\mathrm{c},\mathbf{R}}(i\omega) = \int_{\mathrm{BZ}} \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} e^{-i\mathbf{k}\cdot\mathbf{R}} W_{PQ}^{\mathrm{c}}(\mathbf{k},i\omega)$$

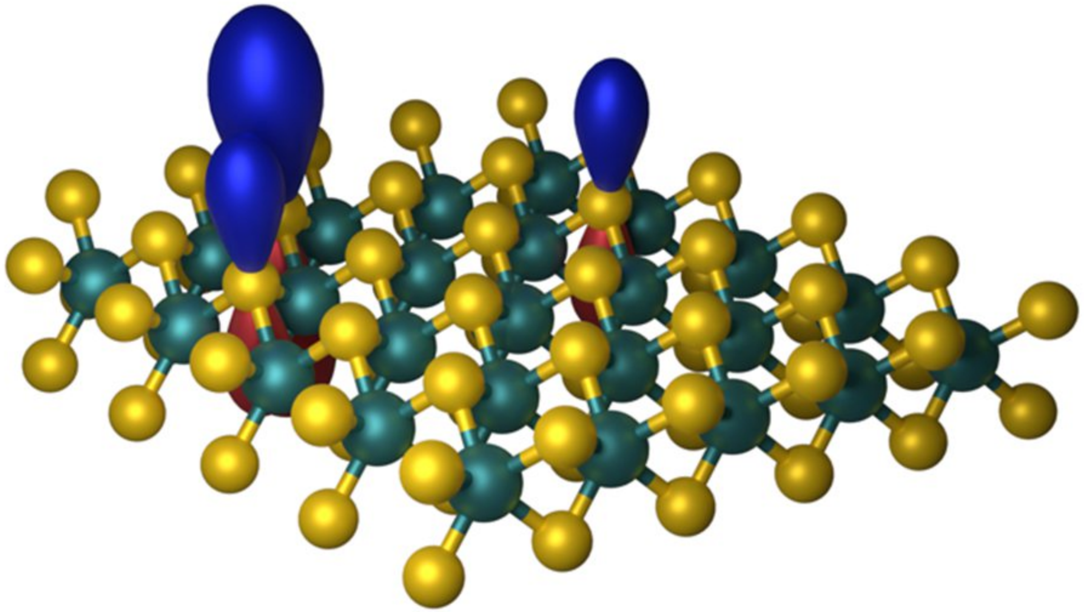
$$\Sigma_{\lambda\sigma}^{\mathrm{c},\mathbf{R}}(i\tau) = i \sum_{P\nu} \sum_{\mathbf{R}_1}^{\mathrm{SC}} \sum_{\mathbf{S}_1}^{\mathrm{SC}} \left[\sum_{\mu} \sum_{\mathbf{S}_2}^{\mathrm{SC}} (\lambda\mathbf{0}\, \mu\mathbf{S}_1 - \mathbf{S}_2 | P\mathbf{R}_1)\, G_{\mu\nu}^{\mathbf{S}_2}(i\tau) \right] \times \left[\sum_{Q} \sum_{\mathbf{R}_2}^{\mathrm{SC}} (\sigma\mathbf{R}\, \nu\mathbf{S}_1 | Q\mathbf{R}_1 - \mathbf{R}_2)\, W_{QP}^{\mathrm{c},\mathbf{R}_2}(i\tau) \right]$$

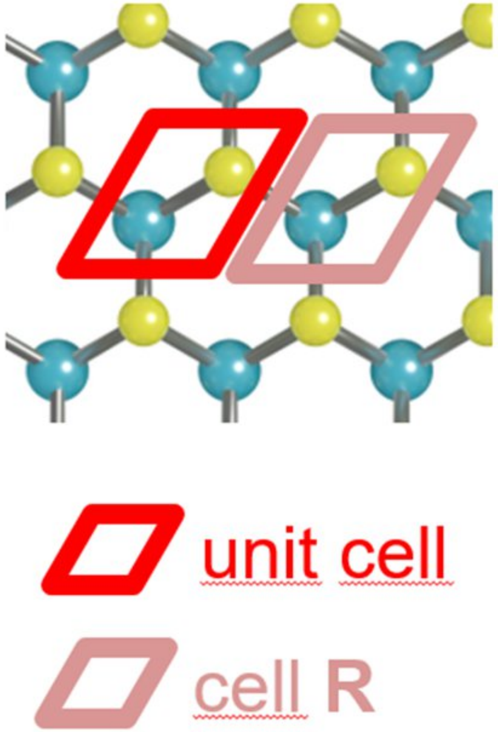
$$\Sigma_{\lambda\sigma}^{\mathrm{c}}(\mathbf{k},i\tau) = \sum_{\mathbf{R}}^{\mathrm{SC}} e^{i\mathbf{k}\cdot\mathbf{R}} \Sigma_{\lambda\sigma}^{\mathrm{c},\mathbf{R}}(i\tau)$$

$$\varepsilon_{n\mathbf{k}}^{G_0W_0} = \varepsilon_{n\mathbf{k}}^{\mathrm{DFT}} + \Sigma_{n\mathbf{k}}^{\mathrm{x}} + \mathrm{Re}\,\Sigma_{n\mathbf{k}}^{\mathrm{c}}(\varepsilon_{n\mathbf{k}}^{G_0W_0}) - v_{n\mathbf{k}}^{\mathrm{xc}}$$

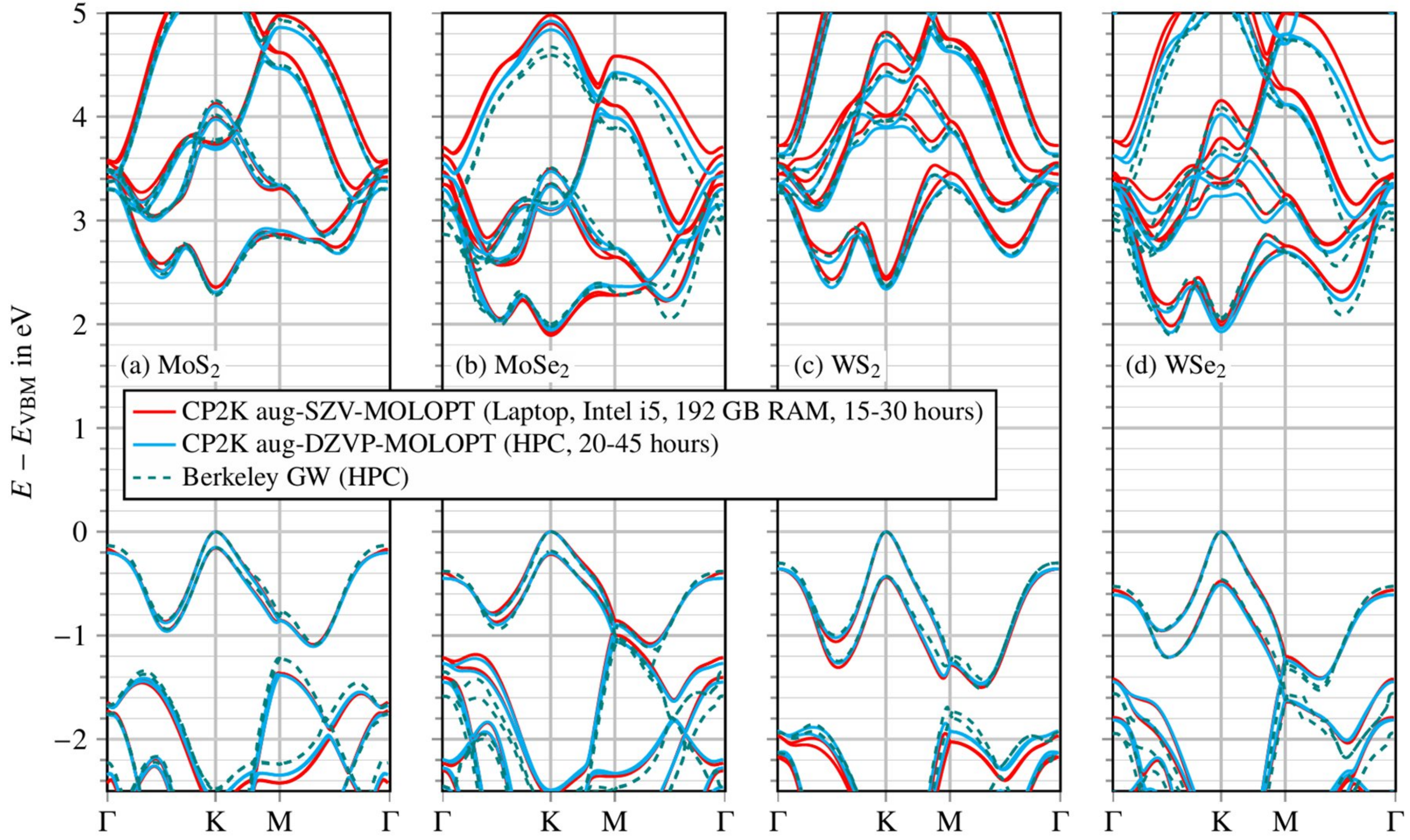
R. Pasquier, M. Camarasa, A. Hehn, D. Hernangómez, J. WIlhelm, arxiv 2507.18411 (2025)

Example: TMD bandstructures

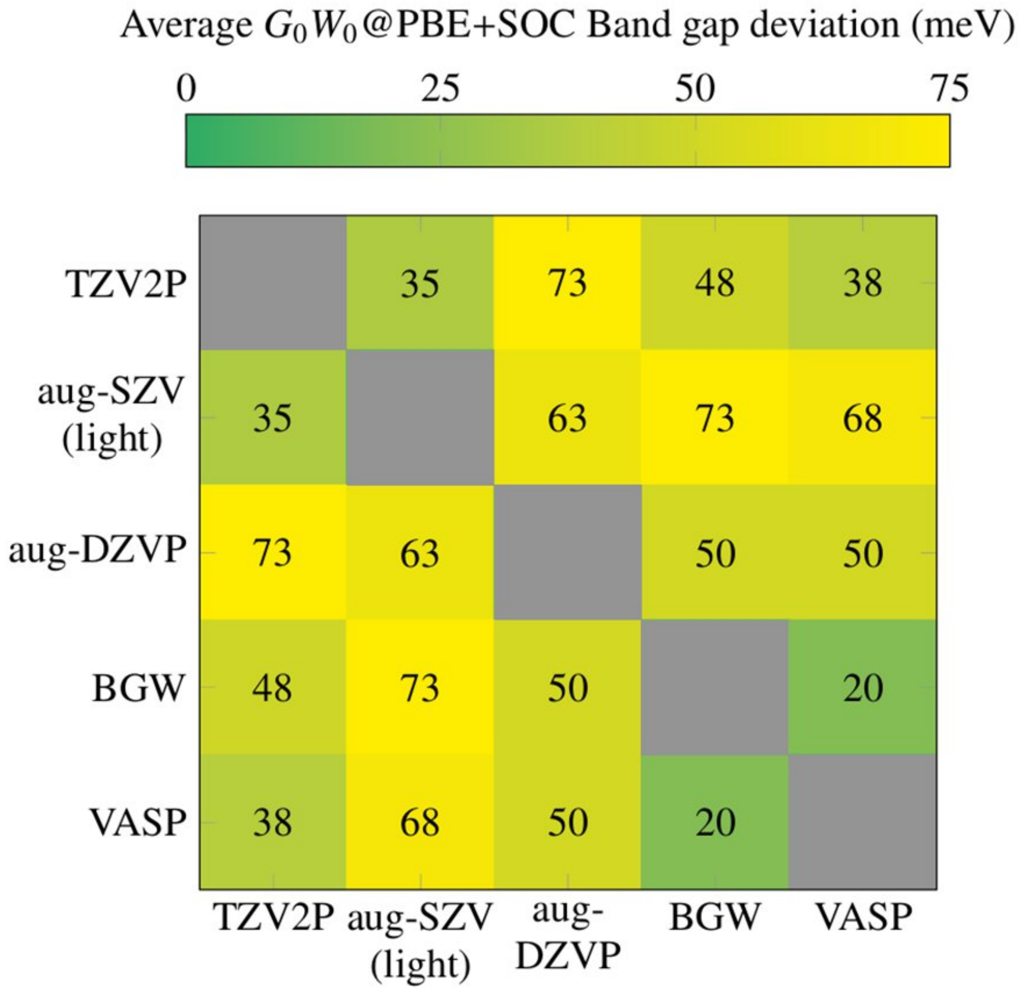


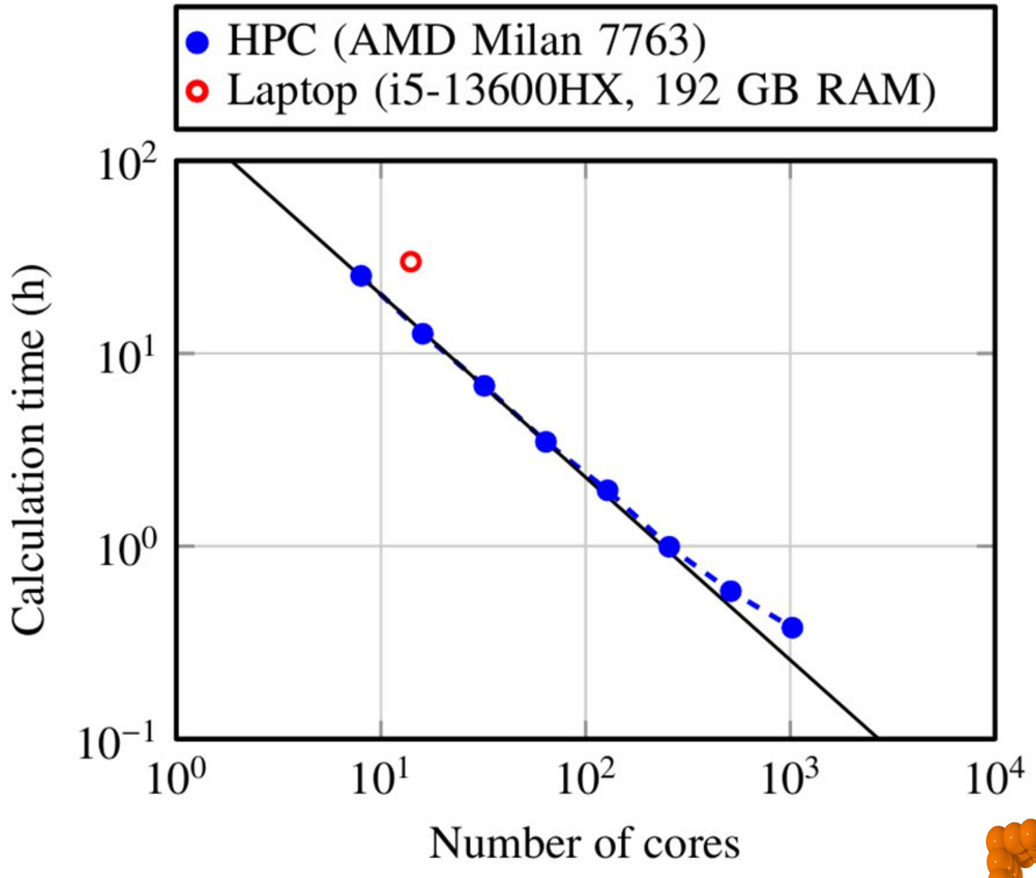


Example: TMD bandstructures

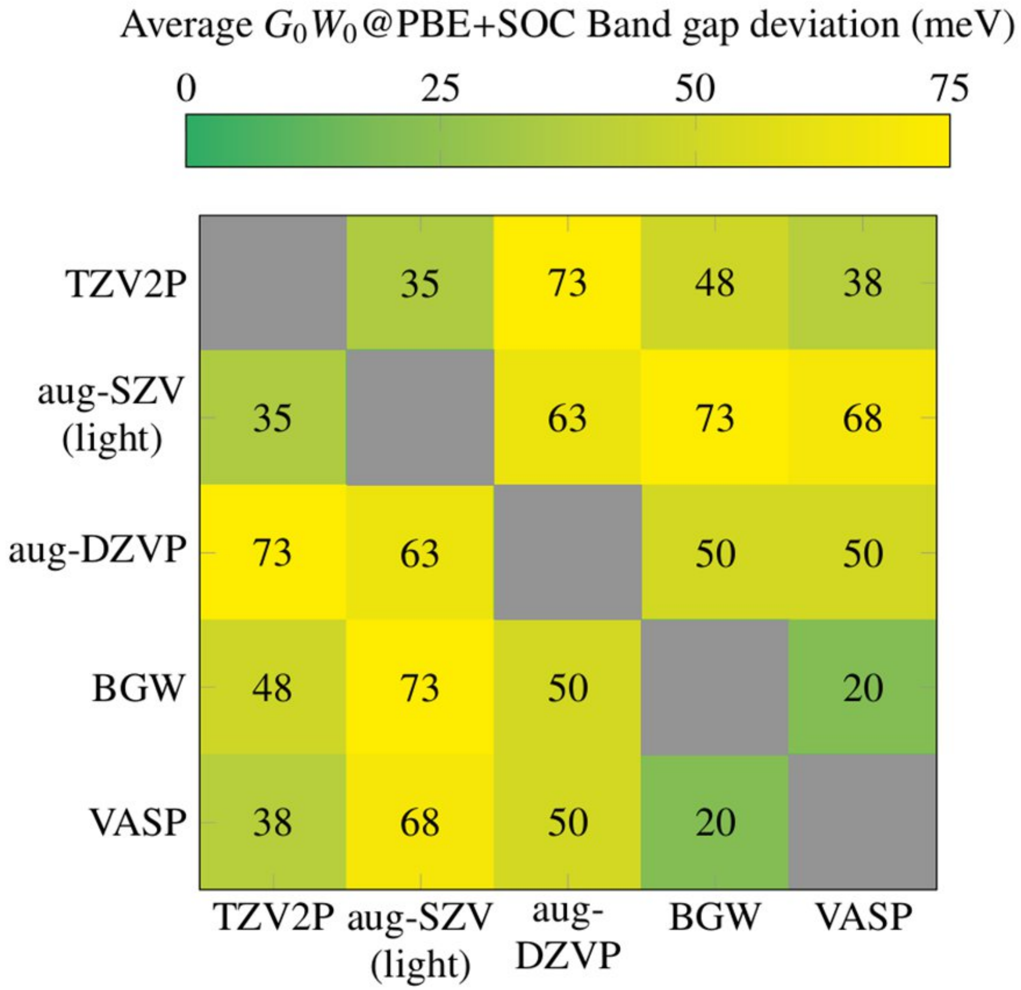


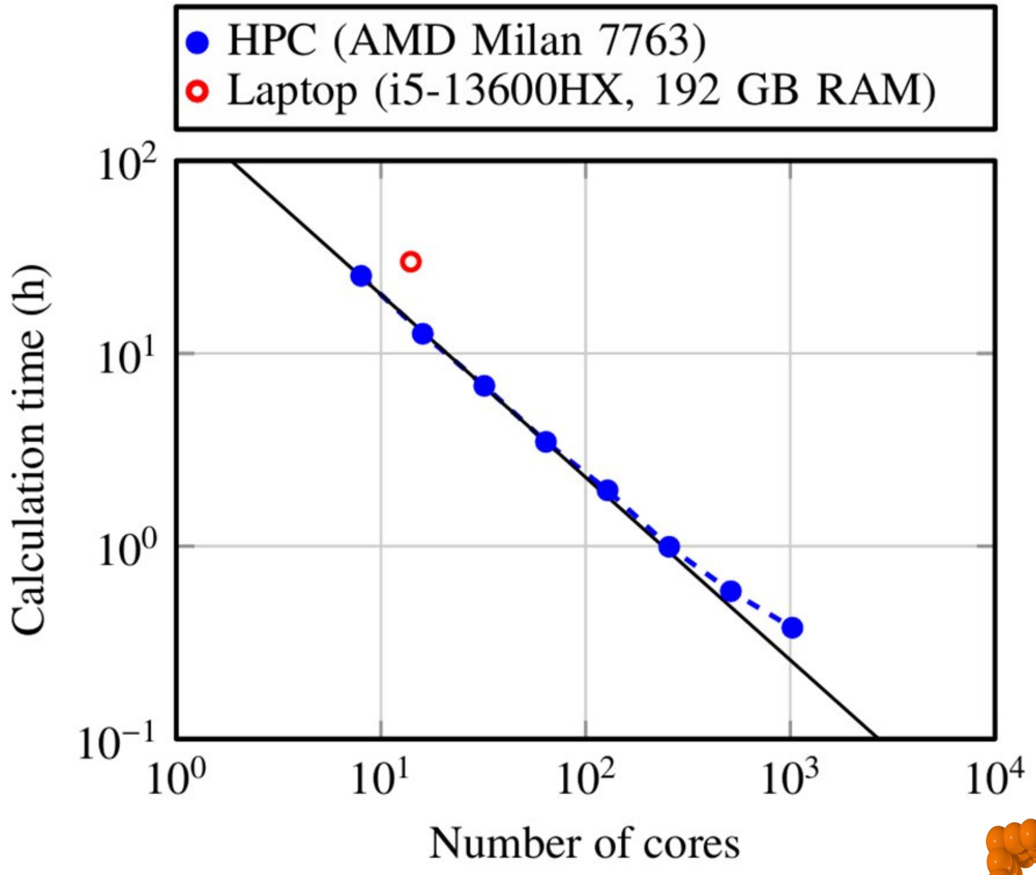
Accuracy and speed of the algorithm





Accuracy and speed of the algorithm



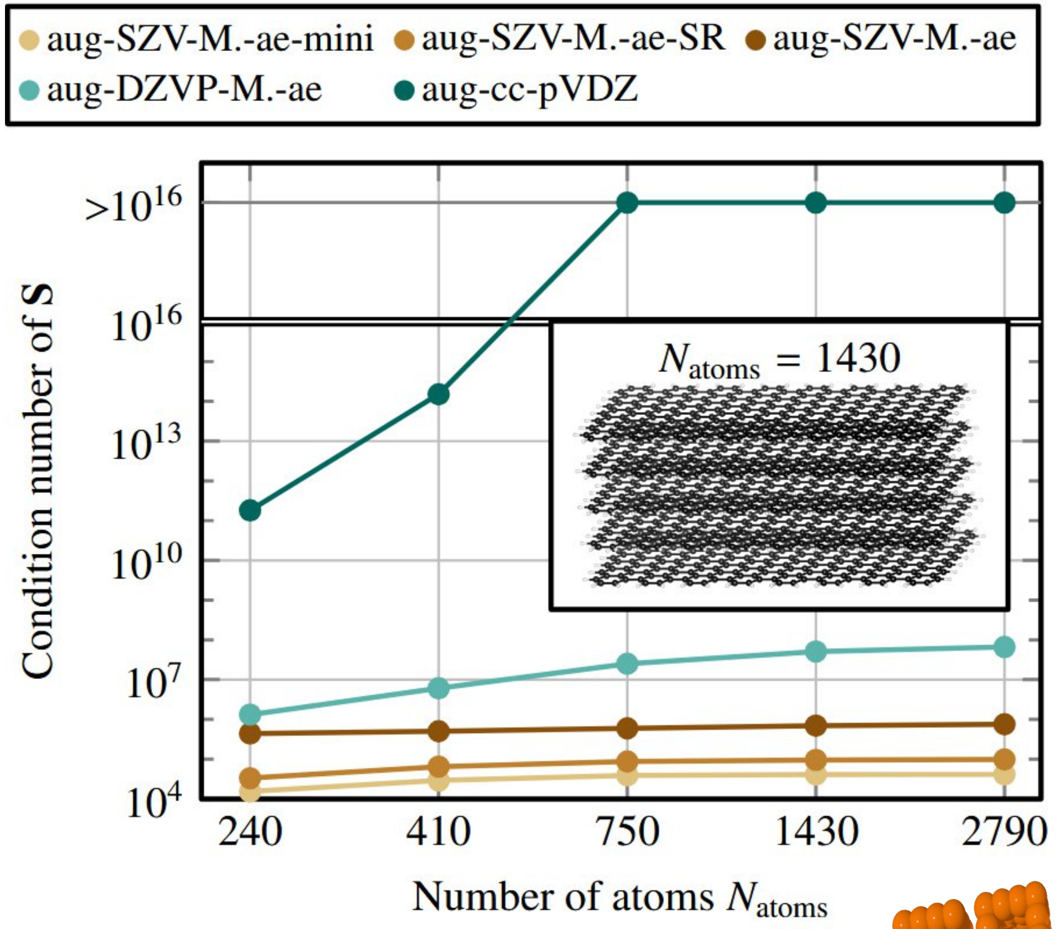


All-electron basis sets for excited states in nanostructures

Roothaan-Hall equations in KS-DFT

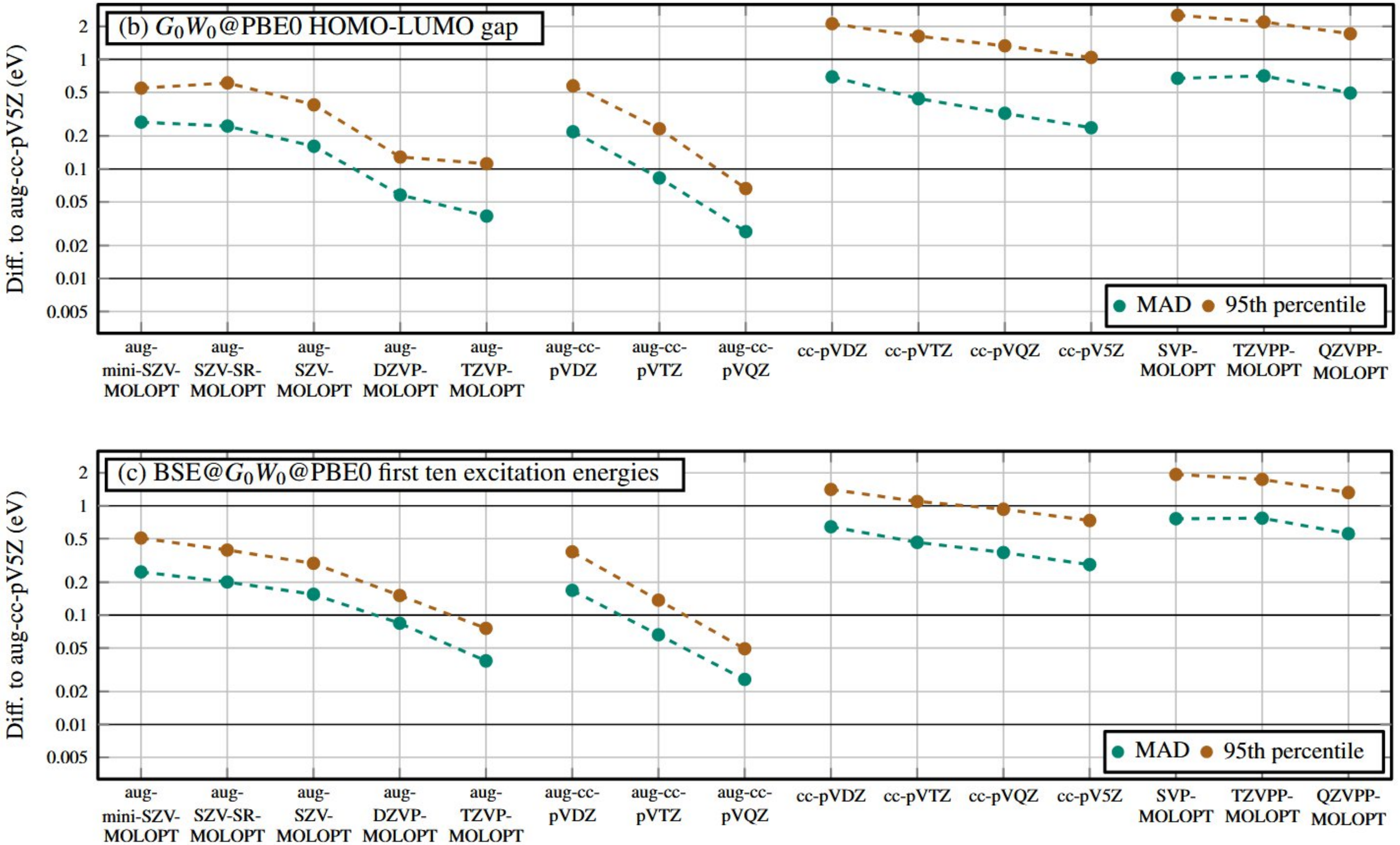
$$\sum_{\nu} h_{\mu\nu} C_{\nu n} = \sum_{\nu} S_{\mu\nu} C_{\nu n} \varepsilon_n \,,$$

$$S_{\mu\nu} = \int d\mathbf{r}\, \phi_\mu(\mathbf{r})\, \phi_\nu(\mathbf{r})\,.$$



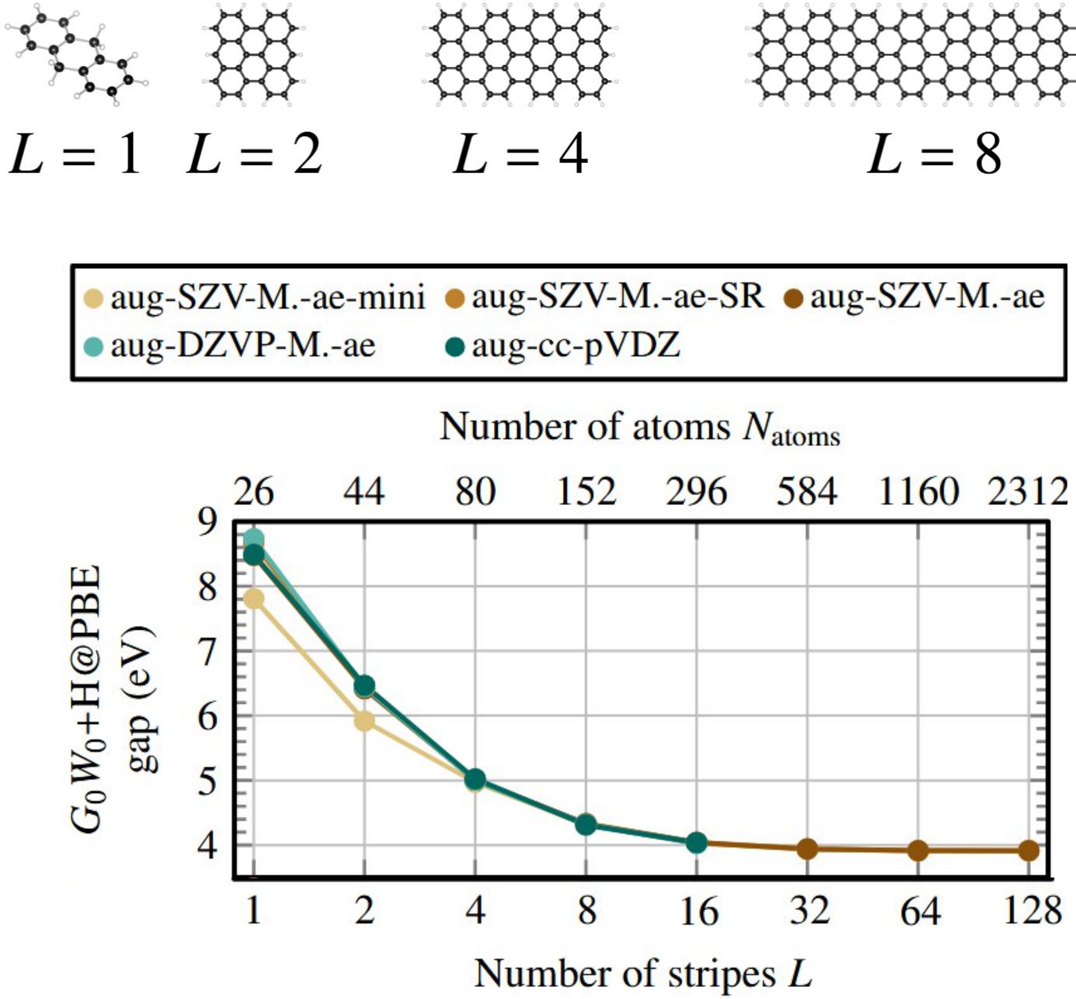
arXiv:2508.12884; Github for basis sets

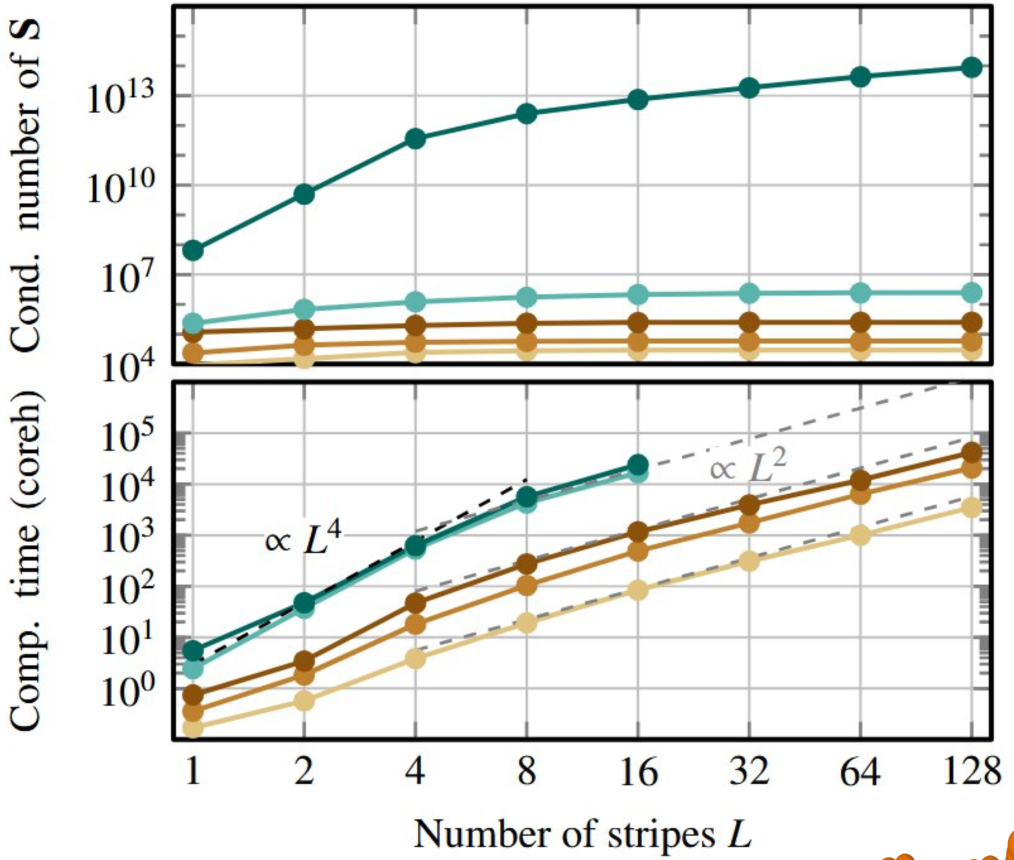
All-electron basis sets for excited states in nanostructures



arXiv:2508.12884; Github for basis sets

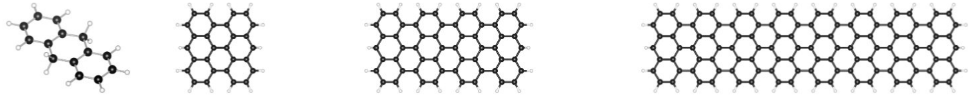
All-electron basis sets for excited states in nanostructures





[arXiv:2508.12884](); [Github for basis sets]()

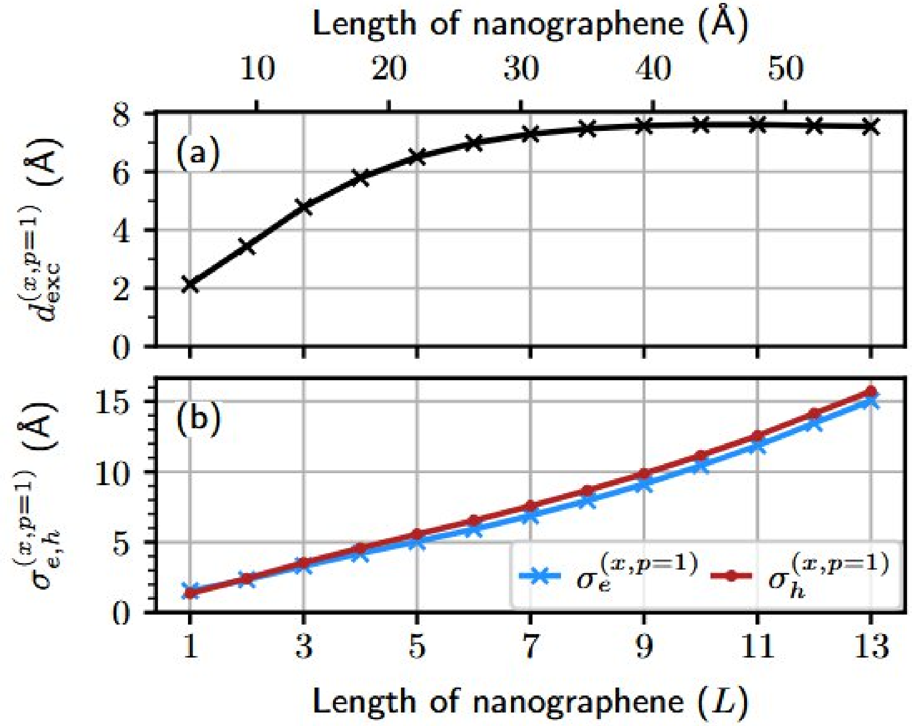
Spatial descriptors for excited states

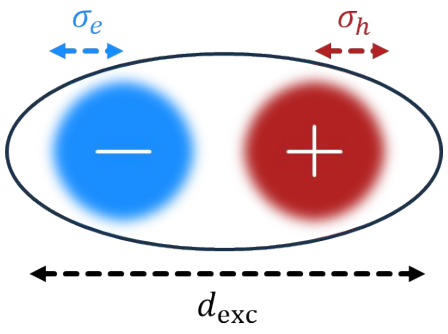


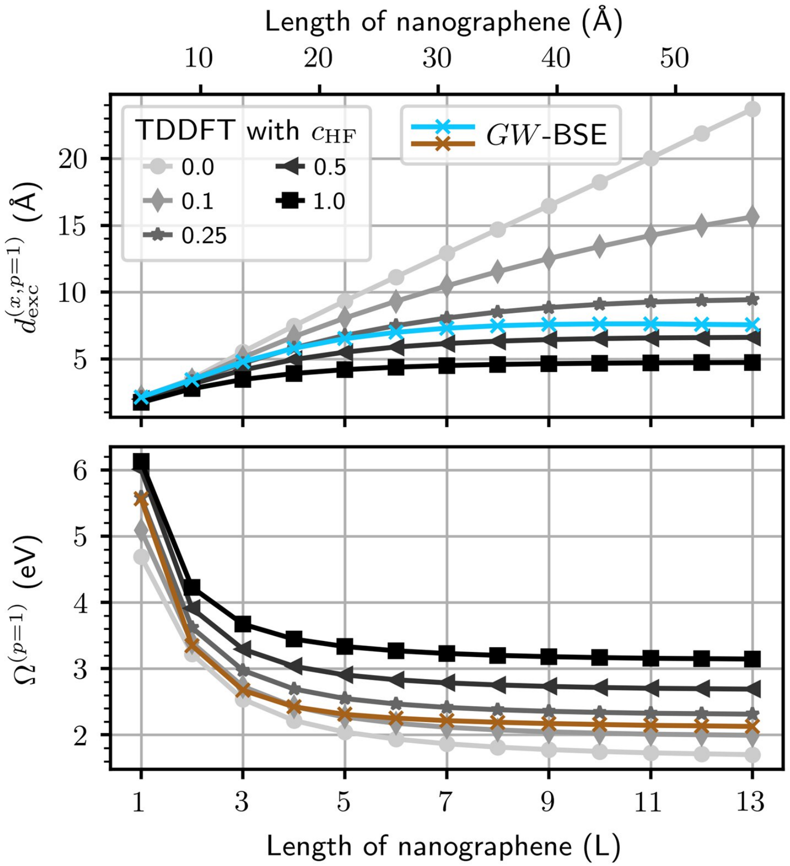
$L = 1$ $L = 2$ $L = 4$ $L = 8$

$$\begin{pmatrix} \mathbb{A} & \mathbb{B} \\ \mathbb{B} & \mathbb{A} \end{pmatrix} \begin{pmatrix} \mathbf{X}^{(n)} \\ \mathbf{Y}^{(n)} \end{pmatrix} = \Omega^{(n)} \begin{pmatrix} \mathbb{1} & 0 \\ 0 & -\mathbb{1} \end{pmatrix} \begin{pmatrix} \mathbf{X}^{(n)} \\ \mathbf{Y}^{(n)} \end{pmatrix}$$

$$\Psi^{(n)}_{\text{exc}}(\mathbf{r}_e, \mathbf{r}_h) = \sum_{i,a} X^{(n)}_{ia} \psi_a(\mathbf{r}_e) \psi_i(\mathbf{r}_h) + Y^{(n)}_{ia} \psi_i(\mathbf{r}_e) \psi_a(\mathbf{r}_h)$$







Idea from S. A. Mewes et al., J. Chem. Theory Comput. 14, 710 (2018).

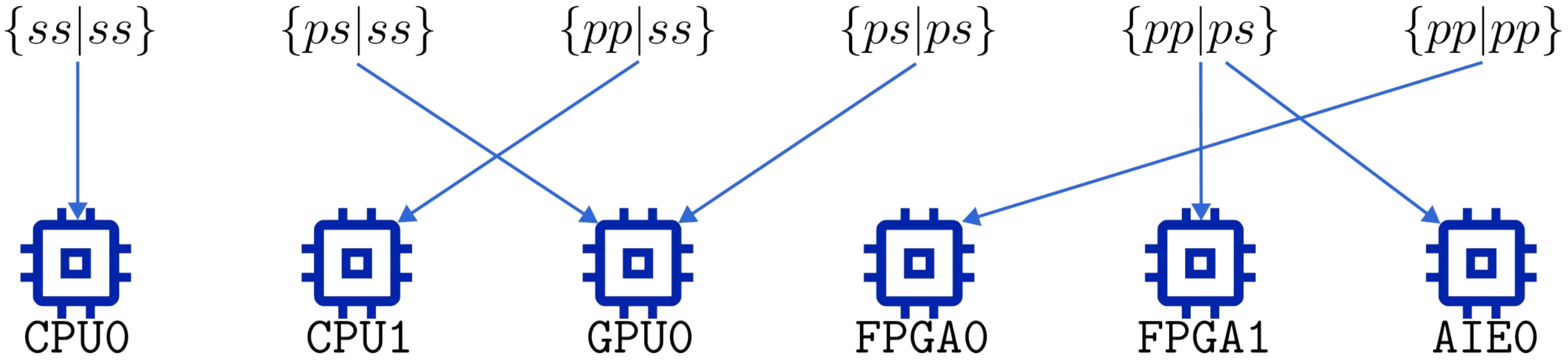
Heterogeneous acceleration of the ERI computation in HFX

- Current implementation (CPU-only):
  - Every CPU-rank is responsible for computing all ERIs between a unique product of atom-subsets

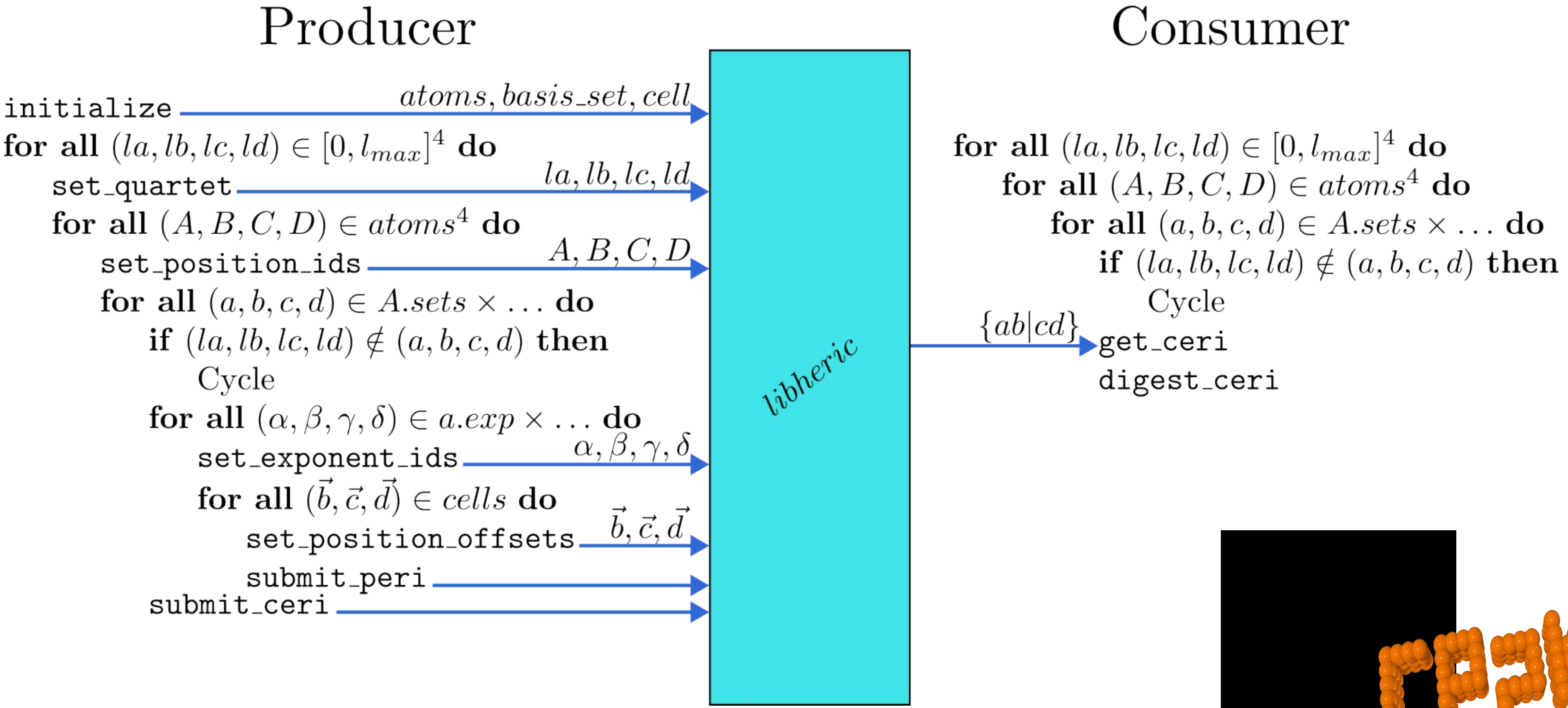
$[A_1, A_2] \times [A_1, A_2] \times [A_1, A_2] \times [A_1]$ $[A_1, A_2] \times [A_1, A_2] \times [A_1] \times [A_2]$ $[A_1, A_2] \times [A_1, A_2] \times [A_2] \times [A_2]$

Heterogeneous acceleration of the ERI computation in HFX

- New implementation:
  - Each accelerator-rank is responsible for subsets of quartet classes
  - Reason: accelerators only slowly switch between quartet classes



Heterogeneous acceleration of the ERI computation in HFX



Heterogeneous acceleration of the ERI computation in HFX

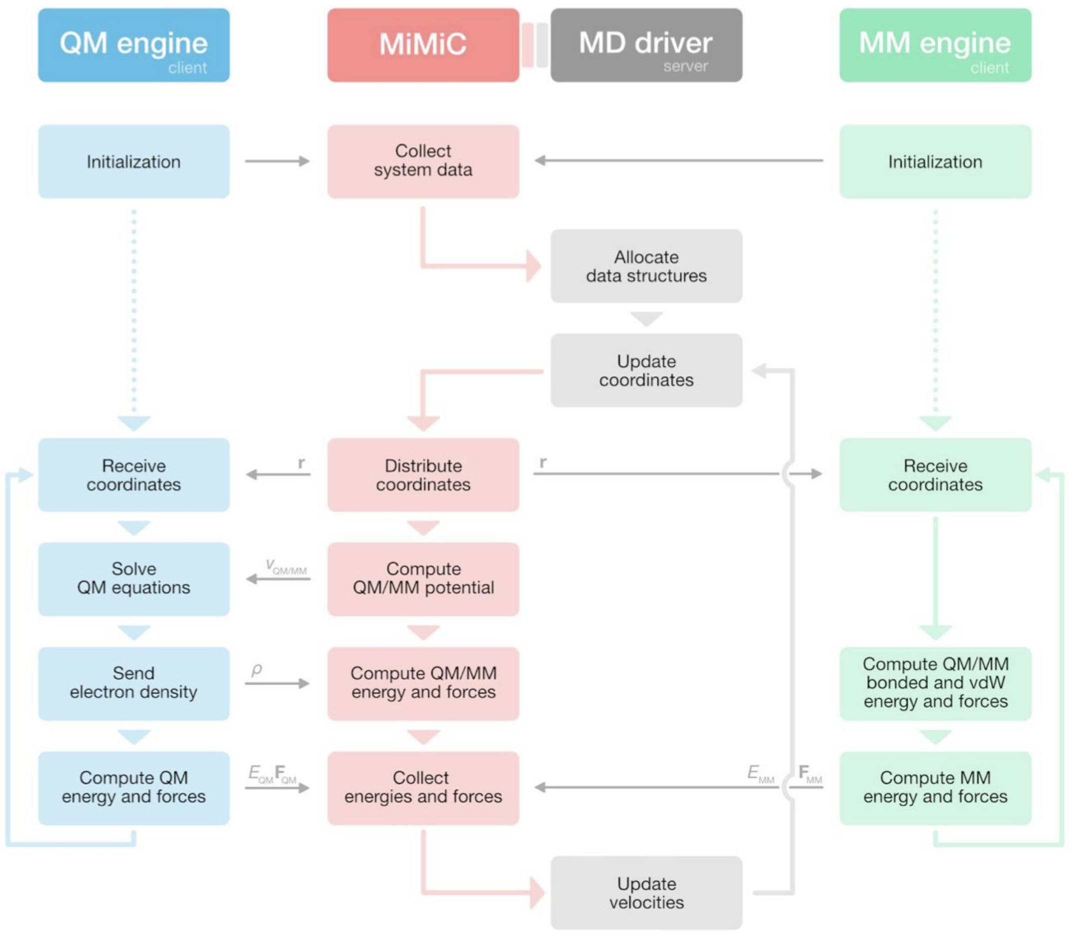
Accelerator Overview

	CPU	GPU	FPGA	AIE
Library	libint	libintx (ref)	SERI (ref), intel-eri (ref)	eri-aie (ref)
Supported Devices	-	Nvidia Tesla GPUs AMD Instinct GPUs	Intel Stratix 10 AMD Alveo U280	AMD VCK5000
Algorithm	OSHGP	MD	Rys	Rys
Contraction	x(early)	x		? (in-dev)
Sph. Transf.		x		? (in-dev)
Compression			x	? (in-dev)
Comment	AVX2 can be enabled AVX512 can be implemented	High-performance only for all-to-all {bra|, |ket} computations -> Difficult to apply screening properly	Unknown how to use Rys quadrature with range-truncated coulomb operator	Unknown how to use Rys quadrature with range-truncated coulomb operator

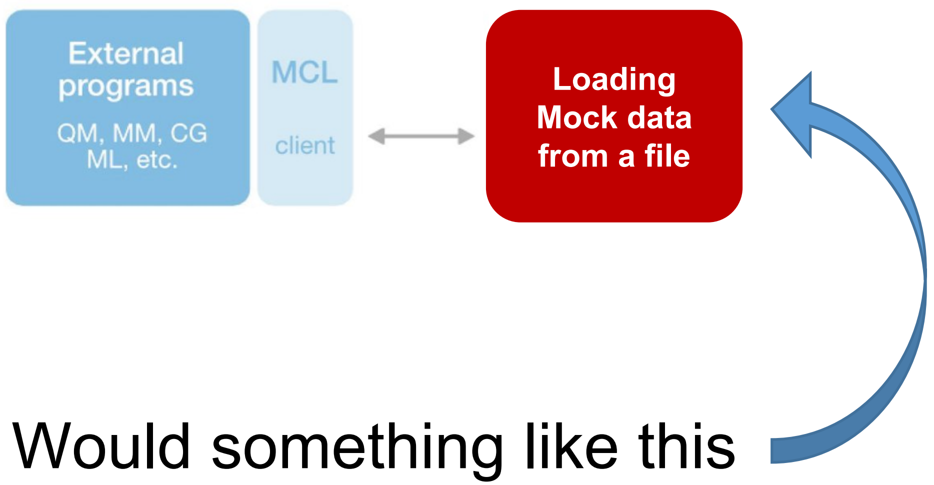
MiMiC Interface: How to tackle tests?

QM/MM with MiMiC

How does it work?



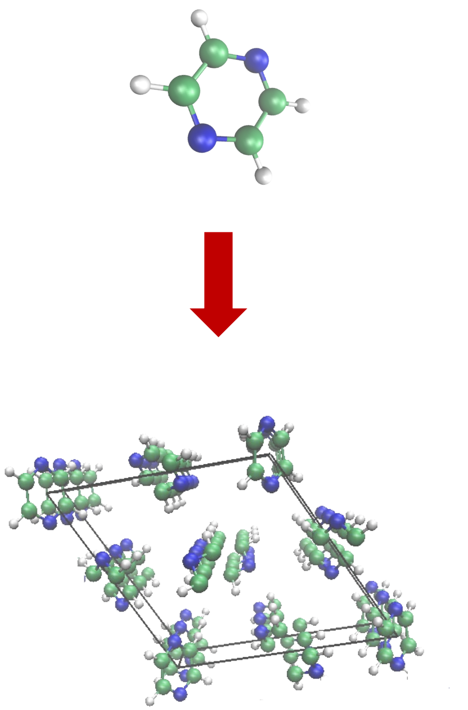
How is the interface implemented?

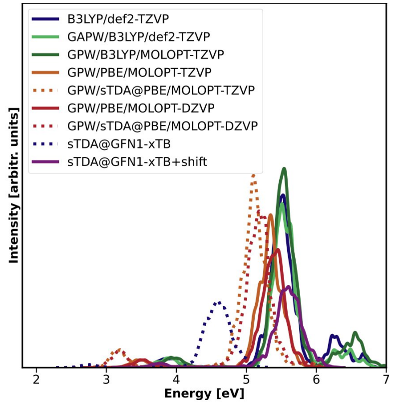


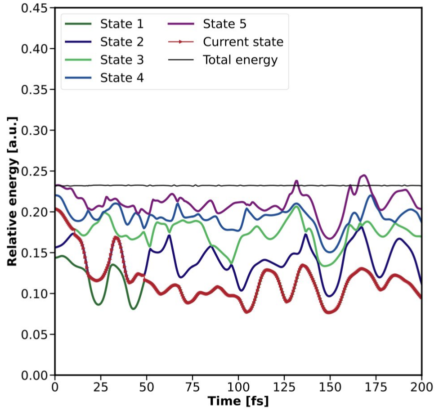
Would something like this
be viable for regression testing

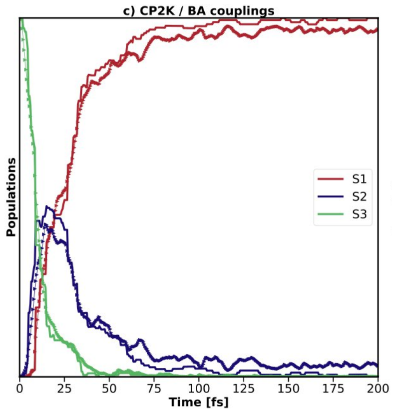
Ongoing developments in Kiel

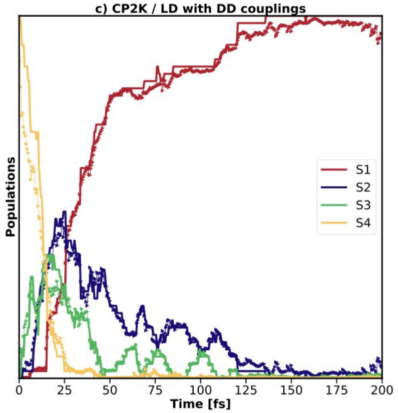
- **Non-adiabatic molecular dynamics** relying on semi-empirical or fast numerical time derivative couplings or local diabatization
- **Smeared occupation** for time-dependent density functional theory ansätze to capture static correlation (based on different distribution functions)
- **Simplified Bethe-Salpeter equation** and multipole expansions for GFN1-xTB (CRC proposal)

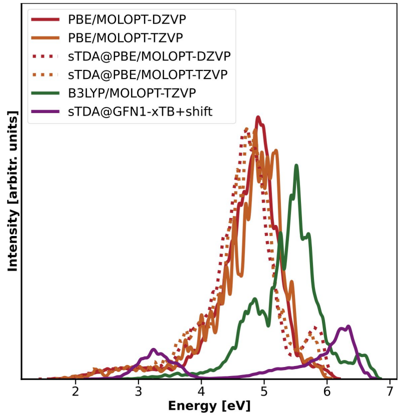


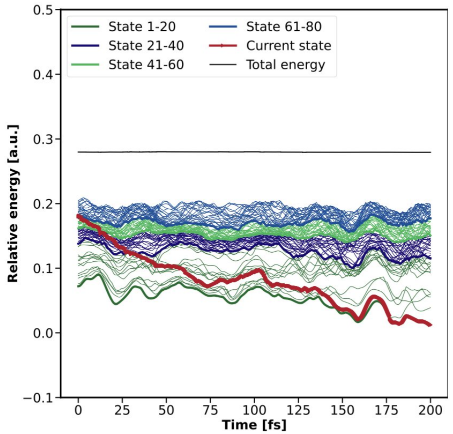


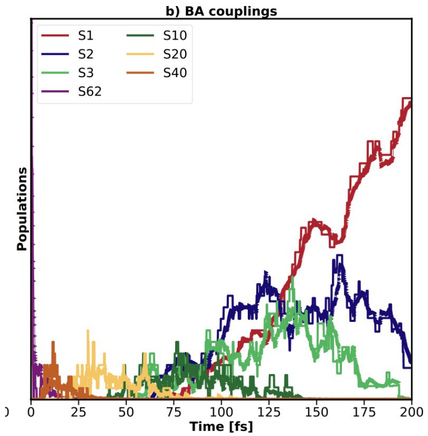


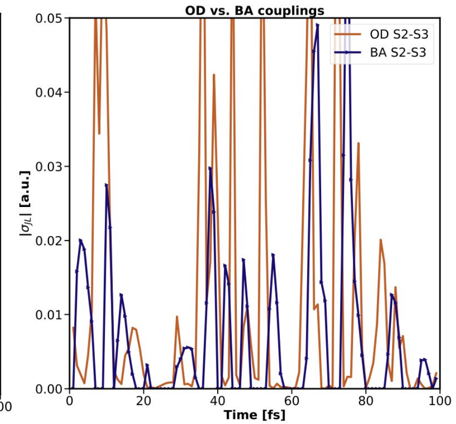












New FFT backend (Frederick Stein)

- Currently: Fortran, GPL license, insignificant acceleration on GPUs
- New backend: C, BSD3 license, exploits more capabilities of FFTW (including R2C/C2R, MPI)
- Core functionality: done
- Integration of local FFTs: done
- Currently only FFTW backend and a simple reference backend (issue: licenses)
- Missing: addition, integration etc. (pw_methods), global FFTs

- Shall we make FFTW a hard dependency?
- Shall we drop the Stefan-Goedecker implementation?

Finite Temperature RPA (Frederick Stein)

- Extension of the zero-temperature RPA formalism to finite temperatures
- Comparable accuracy to VASP
- more expensive than Zero-temperature RPA, especially at high temperatures
- Grids: Matsubara, Minimax (plus tools to optimize them to arbitrary precision)
- TODO: PR (soon), Gradients (not this year)

Skala/GauXC (Frederick Stein, Johann Pototschnig)

- Together with Sebastian Ehlert (Microsoft Research AI for Science)
- GauXC: calculates XC matrix elements from density matrix in GTO basis sets (https://github.com/wavefunction91/GauXC)
- Offers GPU acceleration for a selection of functionals
- Includes Skala (neural network-based XC functional for DFT)

Are there any comments for adding another dependency?

GFN2-xTB and tblite integration (Pototschnig)

- GFN2-xTB energies and gradients are available via the tblite interface for molecules

https://github.com/tblite/tblite

- For periodic boundary conditions there are problems concerning the multipolar Ewalds summation
    Cumulative atomic multipole moments

$$m_a^{klm} = Z_a x_a^k y_a^l z_a^m - \sum_{r\in a}\sum_s D_{rs} < r|x^k y^l z^m|s > - \sum_{k',l',m'>0;k',l',m'\neq k,l,m} \binom{k}{k'}\binom{l}{l'}\binom{m}{m'} x_a^{k-k'} y_a^{l-l'} z_a^{m-m'} m^{k'l'm'}$$

- GFN1-xTB is also available, but there are slight differences (e.g. H and He basis sets)

W. A. Sokalski, M. Shibata, R. Rein, and R. L. Ornstein, “Cumulative atomic multipole moments complement any atomic charge model to obtain more accurate electrostatic properties,” J. Comput. Chem., vol. 13, no. 7, pp. 883–887, Sep. 1992, doi: 10.1002/jcc.540130713.

- tight binding method for ionization potentials: IPEA-xTB

- unrelated: testing D4 correction

V. Ásgeirsson, C. A. Bauer, and S. Grimme, “Quantum chemical calculation of electron ionization mass spectra for general organic and inorganic molecules,” Chem. Sci., vol. 8, no. 7, pp. 4879–4895, 2017, doi: 10.1039/C7SC00601B.

PR on openPMD-api (Franz Poeschel)

- currently in code review https://github.com/cp2k/cp2k/pull/4058
- openPMD alternatively to Cube currently implemented for MO_CUBES, ELF_CUBE and E_DENSITY_CUBE
- file formats: HDF5, ADIOS2, JSON, TOML
- adding to Spack package at https://github.com/spack/spack-packages/pull/1996

Next CP2K Release

- 2026.1: only CMake!

Planned Events in the Context of CP2K

- Past:
  - Recordings from CP2K/GROMACS QM/MM Autumn School available https://www.youtube.com/playlist?list=PLKNFQamVSA_tVYaAwkUeGiJ_sk0lGW30I
- Planned:

Other interesting events:

– Chemical Compound Space Conference 2026, Munich, March, 10th to 12th https://ccsc2026.github.io/

–FortCon2025 https://events.fortrancon.org/event/1/contributions/