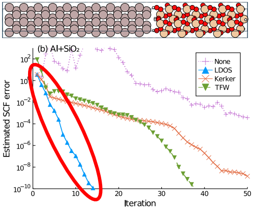
A mathematical viewpoint on robustness and efficiency in first-principle simulations

Michael F. Herbst

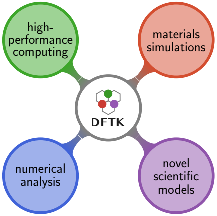
Mathematics for Materials Modelling (`matmat.org`), EPFL

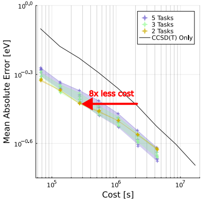
15 February 2024

Slides: `https://michael-herbst.com/talks/2024.02.15_unifr.pdf`



Developing novel DFT algorithms





Overcoming model deviations by multi-task learning

Energy consumption of materials discovery

- Current solutions limited by properties of available materials

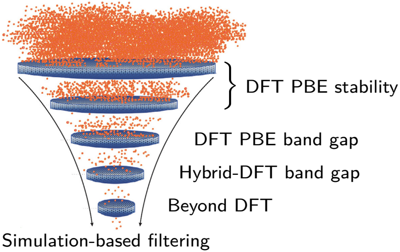
  ⇒ Innovation driven by discovering new materials

- Experimental research extremely energy intensive
  - 1 fume hood $\simeq$ 2-3 average households[1]

⇒ Complement experiment by computational materials discovery

[1] D. Wesolowski *et. al.* Int. J. Sustain. High. Edu. **11**, 217 (2010).

Computational materials discovery



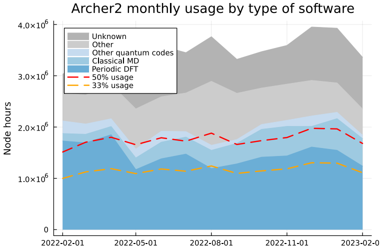
DFT PBE stability

DFT PBE band gap

Hybrid-DFT band gap

Beyond DFT

Simulation-based filtering



- Goal: Only promising candidates go to the lab
- Systematic simulations on $\simeq 10^4 - 10^6$ compounds
  - Complemented by data-driven approaches

- Noteworthy share of world's supercomputing resources
- Growing list of data / workflow management tools

FireWorks AFLOW Automatic-FLOW for Materials Discovery MATERIALSCLOUD AiiDA NOMAD NOVEL MATERIALS DISCOVERY pyiron

Energy consumption of computation

- Energy consumption of LUMI (one of the most efficient):
  - 60 million kWh / year $\simeq$ 1.5 EPFLs $\simeq$ 14000 households
- Challenge of high-throughput:
  - Many parameters to choose (algorithms, tolerances, models)
  - Despite elaborate heuristics: Failure rate $\simeq 1\%$
- Thousands of failed calculations

  $\Rightarrow$ Wasted resources

  $\Rightarrow$ Increased human attention (limits througput)

A focus on robust materials simulations

- **Goal** in MχtMat group:
  - Obtain reliable & efficient simulations
  - Develop and employ mathematical analysis of error
  - Transform empirical wisdom to built-in convergence guarantees

⇒ Understand where and how to spend efforts best

- Practical error indicators:
  - Automatic & robust verification
  - Multi-fidelity statistical surrogates
  - Active learning of missing physics
- Leverage inexactness:
  - Error balancing: Optimal adaptive parameter selection
  - Adaptive tolerances & selective precision

⇒ Multidisciplinary expertise required

A focus on robust materials simulations

- **Goal** in MXtMat group:
  - Obtain reliable & efficient simulations
  - Develop and employ mathematical analysis of error
  - Transform empirical wisdom to built-in convergence guarantees

⇒ Understand where and how to spend efforts best

- Practical error indicators:
  - Automatic & robust verification
  - Multi-fidelity statistical surrogates
  - Active learning of missing physics
- Leverage inexactness:
  - Error balancing: Optimal adaptive parameter selection
  - Adaptive tolerances & selective precision

⇒ Multidisciplinary expertise required

Opportunities for mathematical research

- Gap: Mathematical understanding & simulation practice
- Broad range of concerned mathematical fields:
  - Optimisation, numerical linear algebra, analysis of PDEs, uncertainty quantification, model order reduction, ...
- Application domain: Source for research problems
  - Large-scale eigenvalue problems
    (L. Lin, Y. Saad, C. Yang, ...)
  - Acceleration, fixed-point methods
    (T. Kelly, A. Miedlar, Y. Saad, R. Schneider, H. vd. Vorst, H. Walker, ...)
  - Non-linear PDEs
    (Z. Bai, E. Cancès, G. Friesecke, M. Lewin, I. Sigal, ...)
- Application domain: Source for new methods
  - Davidson diagonalisation (H. vd. Vorst, ...)
  - Thorough exploration of Anderson-type acceleration (see above)
- 17 minisymposia at SIAM in 2021/22 (-CSE, -LA, -MS, -PP, -UQ)
  with contributions related to electronic-structure theory

(Exaggerative) state of codes in this field

Mathematical research

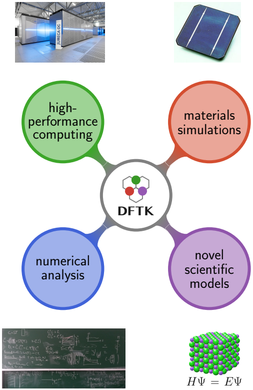
- **Goal:** Numerical experiments
- **Scope:** Reduced models
- High-level **language**:
  Matlab, python, ...
- **Lifetime:** 1 paper
- **Size:** $<$ 1k lines
- Does not care about performance

Application research

- **Goal:** Modelling physics
- **Scope:** All relevant systems
- Mix of **languages:**
  C, FORTRAN, python, ...
- **Lifetime:** 100 manyears
- **Size:** 100k – 1M lines
- Obliged to write performant code

- Working with these codes requires different skillsets
  ⇒ Orthogonal developer & user communities
- Obstacle for knowledge transfer:
  - Mathematical methods never tried in practical setting
    (and may well not work well in the real world)
  - Some issues cannot be studied with mathematical codes
    (and mathematicians may never get to know of them)
- What about emerging hardware, accelerators, performance?
  - Should be the regime of Computer Science (yet another community)

Difficulties of interdisciplinary research



- A social problem . . .
  - Community conventions (e.g. language, publication culture)
  - Speed of research (development of model vs. its analysis)
- . . . cemented in software:
  - Priorities differ ⇒ What is considered a “good code” differs
  - Substantial obstacle for integration

- **Hypothesis:** People compose if software composes

⇒ MatMat **goal**: Software to foster cross-community research

- **DFTK**, the Density-Functional ToolKit
  - Allows restriction to relevant model problems,
  - *and* scale-up to application regime (1000 electrons)
  - Integrated with high-throughput: MARVEL AiiDA

Contents

1. Model error & multi-task surrogates

2. Developing robust SCF algorithms

3. How does DFTK achieve this?

DFT model classes

- DFT energy minimisation problem:

$$\min_\rho \mathcal{E}(\rho) = \min_\rho \left[E_{\text{core}}(\rho) + E_H(\rho) + E_{\text{xc}}(\rho)\right]$$

- DFT model hierarchy for E_{xc}: Jacob's ladder
  - Each *rung* defines (parametrised) model class
- Higher rungs (think hybrid DFT):
  - Generally more expensive, but also more accurate
  - But: DFT is a non-variational approximation to exact physics

  ⇒ Should not impose accuracy order in statistical learning

- **Guiding idea:** Can we combine information from different functionals to balance accuracy / cost / deviating predictions?

⇒ **Goals:**

- Reduce data generation cost
- Dataset of opportunity

DFT model classes

- DFT energy minimisation problem:
$$\min_{\rho} \mathcal{E}(\rho) = \min_{\rho} \left[E_{\mathsf{core}}(\rho) + E_H(\rho) + E_{\mathsf{xc}}(\rho) \right]$$

- DFT model hierarchy for E_{xc}: Jacob's ladder
  - Each *rung* defines (parametrised) model class

- Higher rungs (think hybrid DFT):
  - Generally more expensive, but also more accurate
  - But: DFT is a non-variational approximation to exact physics

  ⇒ Should not impose accuracy order in statistical learning

- **Guiding idea:** Can we combine information from different functionals to balance accuracy / cost / deviating predictions?

⇒ **Goals:**

  - Reduce data generation cost
  - Dataset of opportunity

Test problem: Ionisation potentials of organic molecules

- Dataset: $\simeq 3000$ small organic molecules[1]
  - ANI-1 subset (2–5 heavy atoms, a few with 6 heavies)
- Targeted quantity: Ionisation potential
  - Note: A challenging quantity for DFT
- Considered models:

	density-functional theory (DFT)			coupled cluster
model	PBE	PBE0	PBE0_DH	CCSD(T)
scaling	$O(N^3)$	$O(N^3)$	$O(N^3)$	$O(N^7)$
advantage	cheap	cheapish	cheapish	accurate
rung	2nd (GGA)	4th (Hybrid)	6th (double Hybrid)	*Reference*

- **Goal:** Surrogate for CCSD(T) but mostly use DFT data

[1] C. Duan, F. Fang, A. Nandy, H. Kulik. J. Chem. Theo. Comput. **16**, 4373 (2020).

Delta learning: Learning to correct the error

- **Idea:** Surrogate for *difference* between high- & low-fidelity
- Gaussian Process (GP) ansatz:

$$\mathsf{IP}^{\mathsf{CCSD(T)}} - \mathsf{IP}^{\mathsf{DFT}} = f(\xi) + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0, \sigma^2 I) \qquad \text{(Gaussian noise)}$$

$$f \sim \mathcal{GP}\left(\mu, K_\theta\right) \qquad \text{(GP prior)}$$

ξ: vector of molecular descriptors, I^x: vector of simulated data,
K_θ: Kernel (e.g. polynomial, sq. exponential), σ, μ, θ: hyperparameters

- **Training:** Need DFT & CCSD(T) data
- **Prediction:** Add DFT simulation to predicted mean of GP
- Apply recursively: Multiple levels
- Disadvantages:
  - Ordering imposed
  - Data of all lower levels need to be available

Multitasking: All DFT models are equal

- Asymmetric multitasking[1]:
  - ρ^α: Correlation between CCSD(T) and low-fidelity
  - δ^α: Disparity of low-fidelity models
  - GP prior on f^α & δ^α of different kernel, mean, variance
  - ε^α is iid noise

$$\mathsf{IP}^\alpha = f^\alpha(\xi) + \varepsilon^\alpha \qquad \text{for model } \alpha \in \{\mathsf{CCSD(T)}, \mathsf{PBE}, \mathsf{PBE0}, \dots\}$$

$$f^{\mathsf{PBE}}(\xi) = \rho^{\mathsf{PBE}} f^{\mathsf{CCSD(T)}}(\xi) + \delta^{\mathsf{PBE}}(\xi)$$

$$f^{\mathsf{PBE0}}(\xi) = \rho^{\mathsf{PBE0}} f^{\mathsf{CCSD(T)}}(\xi) + \delta^{\mathsf{PBE0}}(\xi)$$

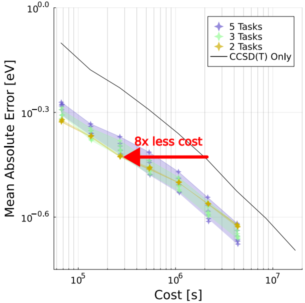
- Remarks:
  - Allows model discrepancies
  - Keeps analytical formula for posterior
  - Calibration set: Fix ρ^α by Pearson correlation, optimise hyperparameters

[1] G. Leen, J. Peltonen, S. Kaski. Mach. Learn. **89**, 157 (2012)

Multitasking: IP results[1]

	Core	Additional	Target
CCSD(T)	●●●●●●●●●●		●●●●●●●●●●
PBE0_DH	●●●●	●●●●	
PBE0	●● ●●	●● ●●	
PBE	●● ●●	●● ●●	
BLYP	●● ●●	●● ●●	●●●●●●●●●●

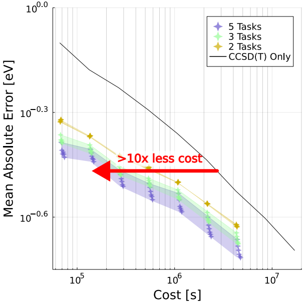
- Goal: Prediction of **T** systems at CCSD(T) level
  - Here: DFT predictions of **T** supplied (optional in our setup)



8x less cost

C and **A** data not shared between tasks

(worst case)



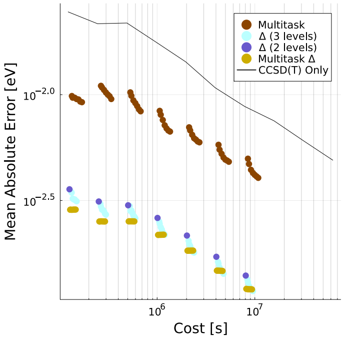
>10x less cost

C and **A** data fully shared between tasks

(best case)

[1] K. Fisher, MFH, Y. Marzouk *Multitask meth. to predict molec. prop. from heterogeneous data* arXiv 2401.17898

Multitasking: Comparison to Δ for water[1]



- Different test case: Water 3-body energy
- Model differences smoother (smaller mean and variances)

⇒ Bare multitask worse then Δ learning

- Combination of both ideas: Multitask-Δ
- Keeps flexibility improvements of multitask approaches

[1] K. Fisher, MFH, Y. Marzouk *Multitask meth. to predict molec. prop. from heterogeneous data* arXiv 2401.17898

Contents

1. Model error & multi-task surrogates

2. Developing robust SCF algorithms

3. How does DFTK achieve this?

Density-functional theory (insulators)

- DFT approximation: Non-linear eigenvalue problem

$$\begin{cases} \forall i \in 1 \dots N : \ \left(-\frac{1}{2}\Delta + V\left(\rho_\Phi\right)\right) \psi_i = \varepsilon_i \psi_i, \\ V(\rho) = V_{\text{nuc}} + v_C \rho + V_{\text{XC}}(\rho), \\ \rho_\Phi = \sum_{i=1}^{N} \left|\psi_i\right|^2, \\ \Phi = (\psi_1, \dots, \psi_N) \in \left(L^2(\mathbb{R}^3, \mathbb{C})\right)^N \text{orthogonal} \end{cases}$$

nuclear attraction V_{nuc}, exchange-correlation V_{XC}, Hartree potential $-\Delta\left(v_C \rho\right) = 4\pi\rho$

- Solved as self-consistent field (SCF) problem:

$$\rho\big(V(\rho)\big) = \rho$$

- Hits plenty of “non-“s: Non-convex, non-linear, non-local, non-smooth

Self-consistent field problem

- Density-mixing SCF procedure (preconditioner P, damping α)

$$\rho_{n+1} = \rho_n + \boldsymbol{\alpha}\, P^{-1} \left[\rho(V(\rho_n)) - \rho_n\right]$$

- Near a fixed-point the error goes as

$$e_{n+1} \simeq \left[1 - \boldsymbol{\alpha} P^{-1} \varepsilon^\dagger\right] e_n$$

with dielectric matrix $\varepsilon^\dagger = (1 - \chi_0 K)$, $K(\rho) = V'(\rho)$, $\chi_0(V) = \rho'(V)$

- Convergence iff $-1 < \left[1 - \boldsymbol{\alpha} P^{-1} \varepsilon^\dagger\right] < 1$
  - Dielectric matrix $\varepsilon^\dagger$: Depends on physics (conduction, screening)
  - By second-order conditions: $\varepsilon^\dagger \geq 0$ (near fixed point)

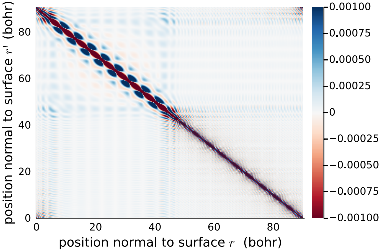
⇒ Crucial to design preconditioner such that $P^{-1} \varepsilon^\dagger \approx I$

- Note: P need to adapt to physics of unknown system!

Black-box P: Local density of states (LDOS) mixing[1]

- Bulk preconditioner (e.g. Kerker) neglect local structure of $\varepsilon^\dagger$

- We propose to employ $\varepsilon^\dagger = (1 - \chi_0 K)$
- $\chi_0(r, r')$ unit-cell internal fluctuations, diagonal dominant:



- Tackle charge sloshing: Consider large-scale variations of χ_0:

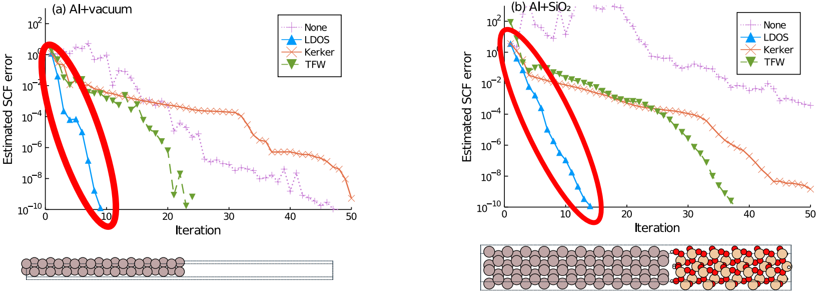
$$\chi_0(r, r') \simeq -\mathsf{LDOS}(r)\delta(r, r') \qquad \text{(homogenisation } \mathsf{LDOS}(r) \approx \int \chi_0(r, r')\, \mathrm{d}r')$$

- Apply preconditioner iteratively:

$$P^{-1}\rho_n = \left[1 - \widetilde{\chi_0} K)\right]^{-1} \rho_n, \qquad \widetilde{\chi_0}(r, r') = -\mathsf{LDOS}(r)\delta(r, r')$$

[1] MFH, A. Levitt. J. Phys. Condens. Matter **33**, 085503 (2021).

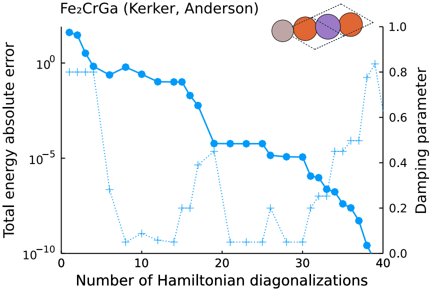
LDOS preconditioning (examples)[1]



- Inhomogeneous material: Aluminium metal + Insulator
- LDOS automatically interpolates between Kerker mixing (suitable for metals) and no mixing (suitable for insulators)
  - ⇒ Based on mathematical understanding of screening
  - ⇒ Parameter-free and black-box

[1] MFH, A. Levitt. J. Phys. Condens. Matter **33**, 085503 (2021).

Black-box α: Adaptive damping[1]



- **Theorem:** SCF convergence guaranteed if α small enough (see paper)
- α adapted *in each step* using line search & quadratic model
- Novelty: Reuse of expensive quantities in next SCF step
  $\Rightarrow$ No overhead if line search immediately successful
- For tricky systems: Adaptive damping has an overhead
  - But: Avoids trial and error
  - Mathematically motivated safeguard mechanism

[1] MFH, A. Levitt. J. Comput. Phys. **459**, 111127 (2022).

DEMO

DEMO

How did DFTK help us to get there?

→ `https://michael-herbst.com/talks/2024.02.15_2024.02.15_unifr.html`

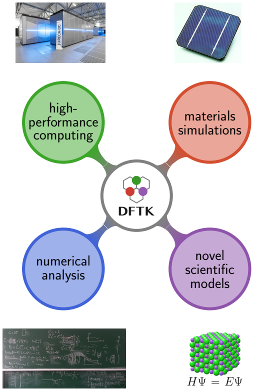
Contents

1. Model error & multi-task surrogates

2. Developing robust SCF algorithms

3. How does DFTK achieve this?

How does DFTK achieve this?



- The magic of julia:
  - Separating the what from the how
- Clear design, inspired by mathematical structure
  - ⇒ Self-explaining code (a clear what)
- Focus on keeping code accessible (7500 lines)
  - Started in 2018, already 30 contributors
  - Key features by undergrads & outsiders

⇒ High-productivity research framework

⇒ Supports joint research across disciplines

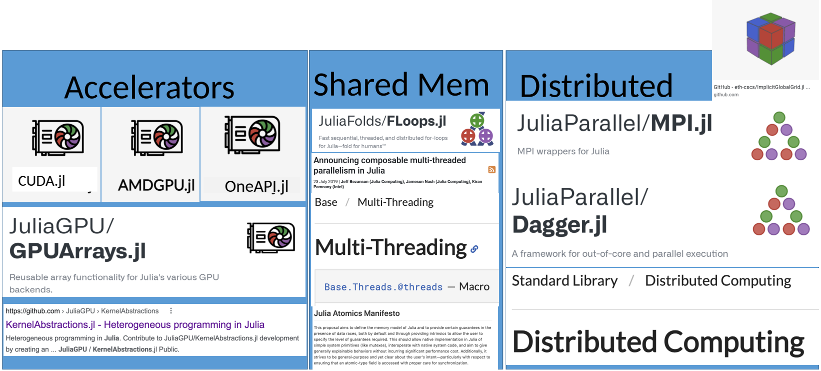
Separating the what from the how

- Why is this separation so important . . .
  - . . . for composable software?
  - . . . for multidisciplinary research?

- Consider the **goal**: Modelling a physical system
- Traditionally users code in detail how the computation should proceed (Imperative programming)
  - How = architecture
  - How = linear algebra primitive (e.g. orthogonalisation)
  - How = memory layout
  - . . .

- But all this has nothing to do with physics!
- Can the how be abstracted away?
  - such that CS / Math can deal with it *independently*
- Let's see julia's HPC developments . . .

julia HPC abstractions



Accelerators

CUDA.jl

AMDGPU.jl

OneAPI.jl

JuliaGPU/**GPUArrays.jl**

Reusable array functionality for Julia's various GPU backends.

https://github.com › JuliaGPU › KernelAbstractions

KernelAbstractions.jl - Heterogeneous programming in Julia

Heterogeneous programming in **Julia**. Contribute to JuliaGPU/KernelAbstractions.jl development by creating an ... **JuliaGPU** / **KernelAbstractions**.jl Public.

Shared Mem

JuliaFolds/**FLoops.jl**

Fast sequential, threaded, and distributed for-loops for Julia—fold for humans™

Announcing composable multi-threaded parallelism in Julia

23 July 2019 | **Jeff Bezanson (Julia Computing), Jameson Nash (Julia Computing), Kiran Pamnany (Intel)**

Base / Multi-Threading

Multi-Threading

`Base.Threads.@threads` — Macro

Julia Atomics Manifesto

This proposal aims to define the memory model of Julia and to provide certain guarantees in the presence of data races, both by default and through providing intrinsics to allow the user to specify the level of guarantees required. This should allow native implementation in Julia of simple system primitives (like mutexes), interoperate with native system code, and aim to give generally explainable behaviors without incurring significant performance cost. Additionally, it strives to be general-purpose and yet clear about the user's intent—particularly with respect to ensuring that an atomic-type field is accessed with proper care for synchronization.

Distributed

GitHub - eth-cscs/ImplicitGlobalGrid.jl ...
github.com

JuliaParallel/**MPI.jl**

MPI wrappers for Julia

JuliaParallel/**Dagger.jl**

A framework for out-of-core and parallel execution

Standard Library / Distributed Computing

Distributed Computing

```julia
function power_method(A, x; niter=100)
    for i = 1:niter
        x = A * x
        x ./= norm(x)
    end
    x
end
```

```julia
A = rand(10, 10); A = A + A' + 10I; x = rand(10)

using LinearMaps, IterativeSolvers
itinv(A) = LinearMap(x -> cg(A, x), size(A)...)

using CUDA
power_method(itinv(CuArray(A)), CuArray(x))

using AMDGPU
power_method(itinv(ROCArray(A)), ROCArray(x))
```

DFTK design: Keeping code concise & accessible

$$\text{Stress} = \frac{1}{\det(\mathbf{L})} \left. \frac{\partial E\left[P_*, (I + \mathbf{M})\,\mathbf{L}\right]}{\partial \mathbf{M}} \right|_{\mathbf{M}=0}$$

```
# Run SCF, get P*
scfres = self_consistent_field(basis)
     L = basis.model.lattice
stress = 1/det(L) * gradient(
    M -> recompute_energy(
                 scfres, (I + M) * L),
    zero(L)
)
```

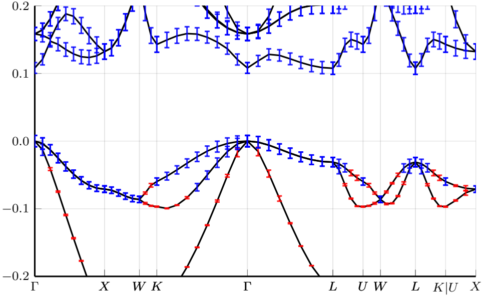
- Stress computation (Definition *vs.* julia code)[1]
- Post-processing step ⇒ Not performance critical
- Comparison of implementation complexity:
  - DFTK : 20 lines[1] (forward-mode algorithmic differentiation)
  - Quantum-Espresso: 1700 lines[2]
  - ≃ 10-week GSoC project

⇒ No performance impact & accessible code

[1] https://github.com/JuliaMolSim/DFTK.jl/blob/master/src/postprocess/stresses.jl

[2] https://github.com/QEF/q-e/blob/develop/PW/src

Support of *a posteriori* error analysis



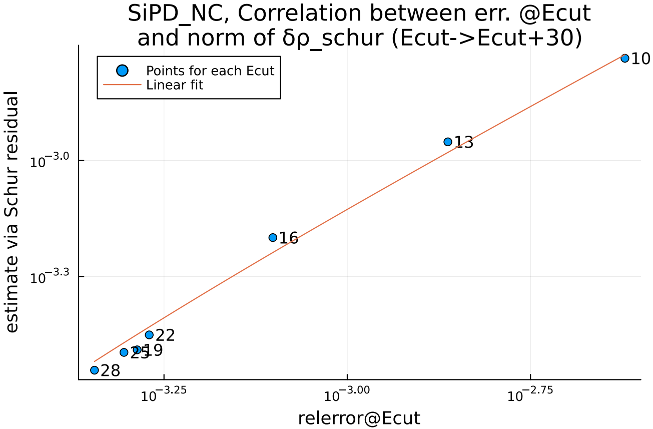
- Albeit the HPC capabilities: Numerical experiments are feasible

- E.g. fully guaranteed error bounds for band structures[1]
- Deals with a reduced Kohn-Sham model and requires interval arithmetic
- Captures basis set error, floating-point error, convergence error

[1] MFH, A. Levitt, E. Cancès. Faraday Discus. **223**, 227 (2020).

Outlook towards DFT properties

- Based on recent perturbative error estimates[1]
- Towards *a posteriori* error estimates for density and forces

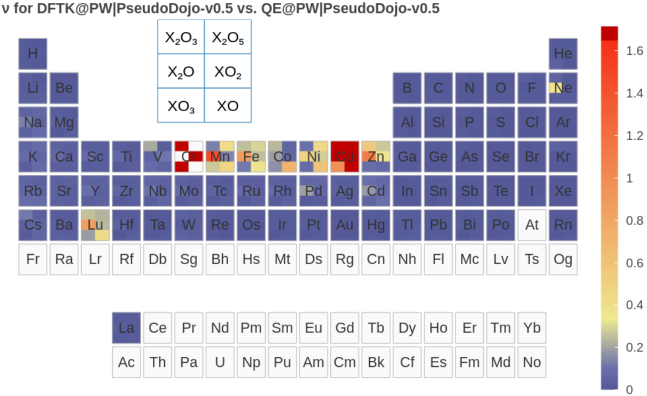


[1] E. Cancès, G. Dusson, G. Kemlin *et. al.* SIAM J. Sci. Comp., **44**, B1312 (2022).

Integration with AiiDA

- Integration with AiiDA high-throughput workflow manager
  - `https://github.com/aiidaplugins/aiida-dftk`

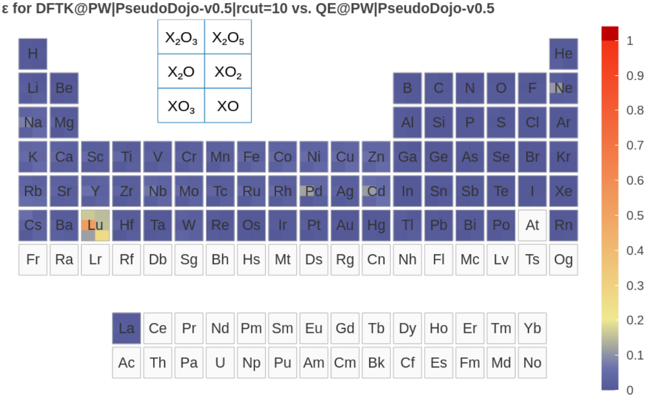
- Used in automated verification tests: $(r_{\text{cut}} = \infty)$



⇒ Excellent agreement QE vs. DFTK

Integration with AiiDA

- Integration with AiiDA high-throughput workflow manager
  - `https://github.com/aiidaplugins/aiida-dftk`
- Used in automated verification tests: $(r_{\text{cut}} = 10\text{a.u.})$



⇒ Excellent agreement QE vs. DFTK

Summary

- Research in the MatMat group
  - Motivated by high-throughput materials design
  - Understand simulation error (numerics, models)
  - Facilitate cross-community interaction
- Black-box strategies for SCF damping & preconditioning
  - Build on combining mathematical and physical insight
  - Safeguard mechanism: Increase robustness for hard cases
- DFTK: Multidisciplinary software development
  - julia-based framework for new DFT algorithms
  - In one code: Reduced problems and high-throughput problems
  - High-productivity research framework
  - Overcome disciplinary barriers: People compose if software composes

Acknowledgements

- Niklas Schmitz (MatMat)
- Cédric Travelletti (MatMat)
- Yihan Wu (EPFL)
- Austin Zadoks (EPFL)
- All DFTK contributors

- Eric Cancès (École des Ponts)
- Katharine Fisher (MIT)
- Antoine Levitt (Université Paris-Saclay)
- Youssef Marzouk (MIT)
- Giovanni Pizzi (PSI)

EPFL MatMat Summer of code

Fonds national suisse

MARVEL
NATIONAL CENTRE OF COMPETENCE IN RESEARCH

MIT

CESMIX

Questions?

MatMat https://matmat.org

mfherbst

michael.herbst@epfl.ch

https://michael-herbst.com/talks/2024.02.15_unifr.pdf

DFTK https://dftk.org

julia https://michael-herbst.com/learn-julia
https://michael-herbst.com/julia-for-materials

EPFL MatMat