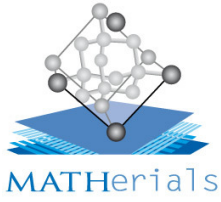
DFTK: The density-functional toolkit

Michael F. Herbst, Antoine Levitt, Eric Cancès

Matherials team, CERMICS, École des Ponts ParisTech

13th September 2019



https://michael-herbst.com/talks/2019.09.13_dftk_moansi.pdf

Why care about plane-wave DFT?

- 13 related papers in the WoS Top 100 list
- Applications:
  - Semiconductors and photovoltaics
  - Crystal structure prediction
  - Properties of alloys
  - ...

$\Rightarrow$ Challenge: Mass screening of materials

- Reliable, black-box codes
- Accuracy vs. computational effort.

Plane-wave DFT in one slide

- Kohn-Sham DFT: Seek minimiser $\Theta = (\psi_1, \ldots, \psi_N) \in (H^1(\mathbb{R}^3))^N$ of

$$\mathcal{E}^{\mathsf{KS}}(\Theta) = \frac{1}{2}\sum_{i=1}^{N} \int_{\mathbb{R}^3} \|\nabla\psi_i\|^2 \,\mathrm{d}\underline{\boldsymbol{r}} + \int_{\mathbb{R}^3} V^{\mathsf{loc}}(\underline{\boldsymbol{r}})\rho_\Theta(\underline{\boldsymbol{r}}) + \int_{\mathbb{R}^3} V^{\mathsf{nloc}}(\underline{\boldsymbol{r}})\rho_\Theta(\underline{\boldsymbol{r}}) + \frac{1}{2}\int_{\mathbb{R}^3}\int_{\mathbb{R}^3} \frac{\rho_\Theta(\underline{\boldsymbol{r}})\rho_\Theta(\underline{\boldsymbol{r}}')}{\|\underline{\boldsymbol{r}} - \underline{\boldsymbol{r}}'\|} \,\mathrm{d}\underline{\boldsymbol{r}}\,\mathrm{d}\underline{\boldsymbol{r}}' + E_{xc}(\rho_\Theta)$$

  with density $\rho_\Theta = \sum_{i=1}^{N} \|\psi_i\|^2$

- Plane-wave basis:

$$\forall \underline{\boldsymbol{k}} : \ \varphi_{\underline{\boldsymbol{G}}} = e^{\imath \underline{\boldsymbol{k}}\cdot\underline{\boldsymbol{x}}} e^{\imath \underline{\boldsymbol{G}}\cdot\underline{\boldsymbol{x}}} \qquad G^2 < 2E_{\mathsf{cut}}$$

- Pseudopotentials: V^{loc}, V^{nloc}
  - Remove core electrons
  - Smoothen valence orbitals (i.e. remove oscillations)

  $\Rightarrow$ Lower required E_{cut}

Some existing PW-DFT codes

- https://www.vasp.at
- https://www.abinit.org
- http://www.castep.org
- https://wiki.fysik.dtu.dk/gpaw
- https://www.quantum-espresso.org
- https://crd-legacy.lbl.gov/~chao/KSSOLV
- ...

⇒ Why another?

Questions related to reliability / high-thoughput

- What's the effect of the pseudopotential?
- Elevated / reduced precision?
- Error estimates / mixed grid methods?
- A fast *and* reliable SCF algorithm?
- Leverage GPU and other accelerators?
- Distributed computing?

$\Rightarrow$ Difficult to address in large codes

$\Rightarrow$ Interdisciplinary setting

Demands for interdisciplinary software

- **Mathematicians:** Toy models and unphysical edge cases
- **Scientist:** Wants to focus on science, not numerics
- **High-performance:** Exploit all hardware specialities
- **Practitioner:**
  - Reliable, black-box, high-level for setup and data analysis

- Everything in one project?
- Still keep a minimalistic code base?
- Need good compromise *and* suitable programming language

The approach of DFTK

- DFTK: density-functional toolkit:
  - Minimalistic code base (not a program package)
  - Use existing libraries and codes
  - Facilitate integration elsewhere
- Accessible to different communities:
  - Use custom Hamiltonians, potentials . . .
  - Use julia: Keep code as high-level as possible
  - Target modern HPC environments

⇒ Toy problems and full-scale applications

DFTK is written in julia

- High-level, dynamical language for HPC:
  - Parallelism, vectorisation, GPU, automatic differentiation, ...
- Key concept: Multiple dispatch
- At runtime: Function compiled *exactly* for argument types
  - ⇒ Easy parallelisation and vectorisation
  - ⇒ Type-specific and hardware-specific optimisations
  - ⇒ E.g. allows to switch computational back end
- Write code once, re-use for many back ends / machines ...
- Interoperability: `FORTRAN`, `C`, `C++`, `python`, R, ...

≈ `python` with deeply integrated numpy

A word about `julia` performance

		duration (s)
`python`	array operations (`numpy`)	11.8
`C`	`gcc`	8.1
`C`	`gcc -O3`	1.1
`C`	`gcc -O3 -march=native`	**0.5**
`FORTRAN`	`gfortran -O3 -march=native`	**0.5**
`julia`	array operations	3.3
`julia`	loops	1.9
`julia`	loops, no bounds check	**0.5**

- Best out of five on my laptop (C, Julia, python code: Antoine Levitt)
- Used software: gcc 8.3, gfortran 9.2.1, python 3.7, numpy 1.16.2, julia 1.0.3

DEMO

DEMO

Show-casing DFTK and julia

Status of DFTK

- Code: `https://github.com/mfherbst/DFTK.jl`

- $\approx$ 9 months of development time
- LDA, GGA functionals from `libxc`, analytic potentials
- Multiple SCF algorithms
- Insulators and metals (smearing)
- Laptop-level parallelism
- Single and double precision

Outlook

- Analysis of SCF convergence ($\rightarrow$ Gaspard Kemlin)
- GPU acceleration
- Automatic differentiation
- More mixed and elevated precision
- Response and properties
- Mixed basis or grid methods

Outlook

- Analysis of SCF convergence ($\rightarrow$ Gaspard Kemlin)
- GPU acceleration
- Automatic differentiation
- More mixed and elevated precision
- Response and properties
- Mixed basis or grid methods

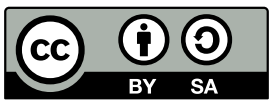
- $>$ Insert your feature here $<$
- `https://github.com/mfherbst/DFTK.jl`

Questions?

DFTK: `https://github.com/mfherbst/DFTK.jl`

Email: `michael.herbst@enpc.fr`

Blog: `https://michael-herbst.com/blog`



MATHerials Inria École des Ponts ParisTech