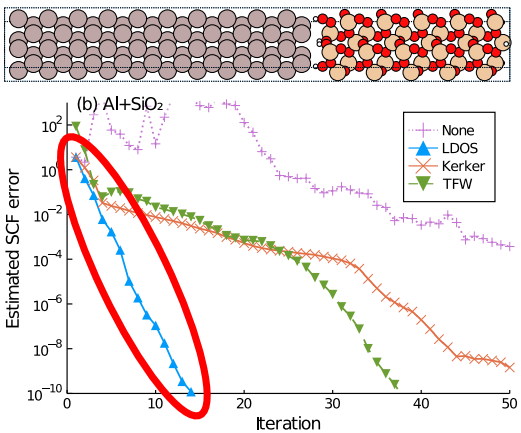
Black-box algorithms and robust error control for density-functional theory

Michael F. Herbst

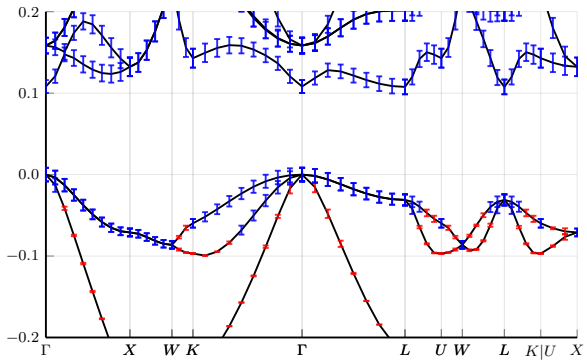
Applied and Computational Mathematics, RWTH Aachen University

10 June 2022

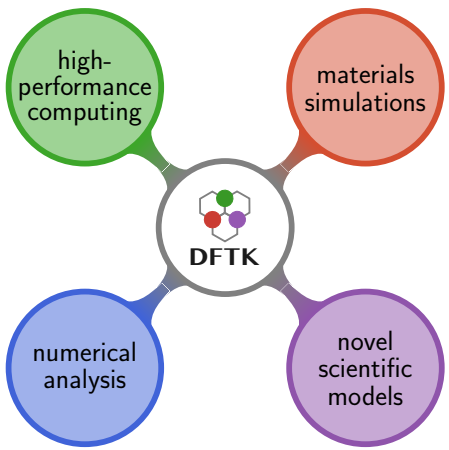
Slides: `https://michael-herbst.com/talks/2022.06.10_bbox_algos_ad.pdf`



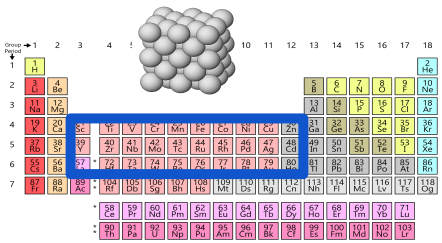
Self-adapting black-box DFT algorithms



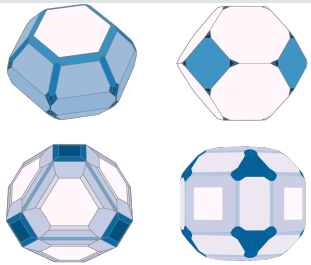
Error control of DFT quantities



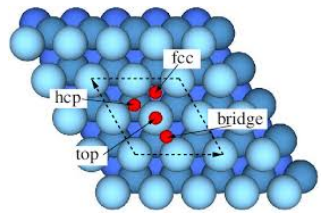
Task: Develop a new metallic catalyst for a surface reaction[1]



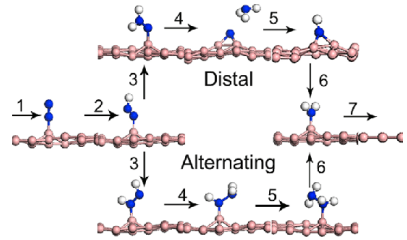
Host metal + dopant
$\simeq 30 \times 30 = 900$



Host surface
$\simeq 3 - 5$



Dopant adsorpotion site
$\simeq 30$



Reaction intermediates
$\simeq 10$

- Combinatorial design space: $\simeq 10^5 - 10^6$ possibilities
- Systematic experiments: Time and cost intensive

$\Rightarrow$ Computational screening to complement and accelerate

- Harvest curated data bases
- Data-driven methods and statistical learning

$\Rightarrow$ Regular need of millions of first-principle calculations

- Growing list of tools to manage workflows and curate data

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

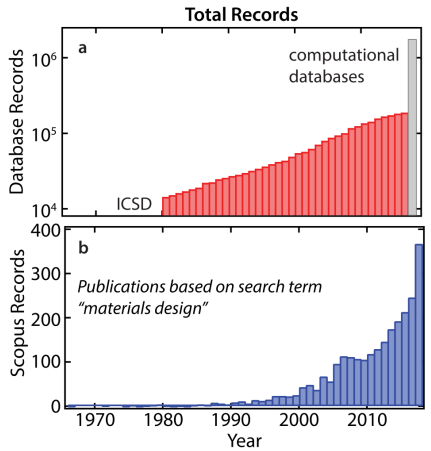
[1] ACED project: `https://www.cmu.edu/aced/index.html`

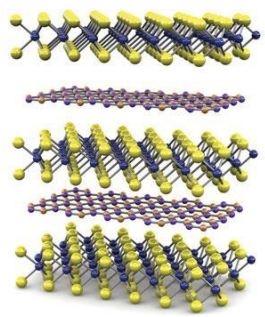
Status of high-throughput screening

- Exponentially growing impact

- Broad span of successful discoveries:[a]
  - Semiconductors[b]
  - Lithium-ion-based batteries[c]
  - Magnetic compounds[d]
  - 2D materials: Batteries[e], electronics[f]

⇒ Crucial tool to tackle 21st century challenges
  - Energy storage
  - Renewable energies
  - Quantum computing
  - ...





[a] K. Alberi *et. al.* J. Phys. D, **52**, 013001 (2019).

[b] S. Luo *et. al.* WIREs Comput. Mol. Sci. **11**, e1489 (2021).

[c] L. Kahle *et. al.* Energy & Environ. Science, **13**, 928 (2020).

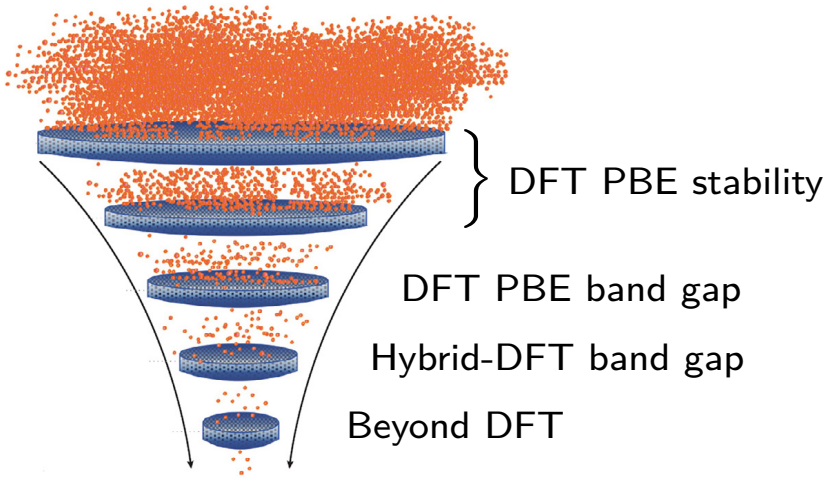
[d] S. Jiang *et. al.* J. Alloys Comp. **867**, 158854 (2021).

[e] A. Babak *et. al.* ACS Nano, **9**, 9507 (2015).

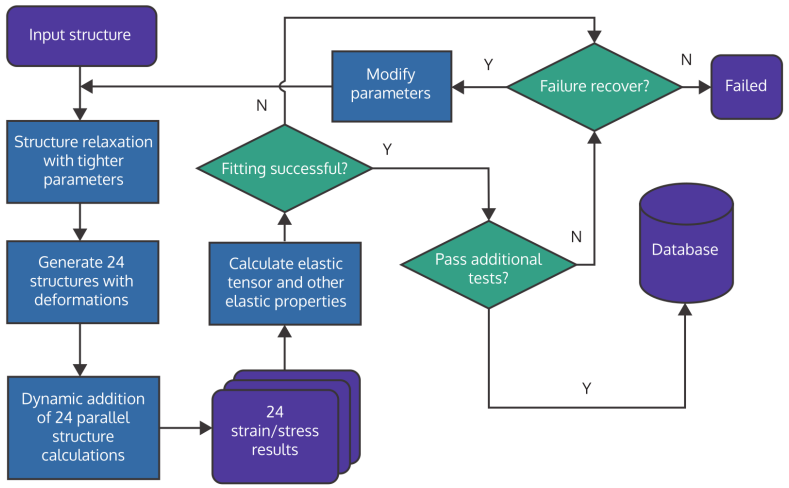
[f] C. Klinkert *et. al.* ACS Nano, **14**, 8605 (2020).

`https://www.edfenergy.com/electric-cars/batteries`;

Sketch of high-throughput workflows



Design funnel for photovoltaic materials



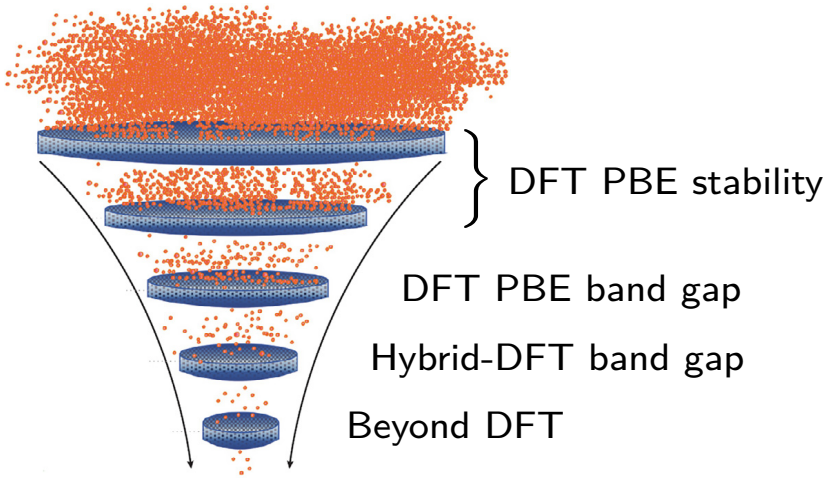
Workflow for computing elasticity tensors

- Many parameters to choose (algorithms, tolerances, models)
  - Elaborate heuristics: Failure rate $\simeq 1\%$
  - Still: Thousands of failed calculations

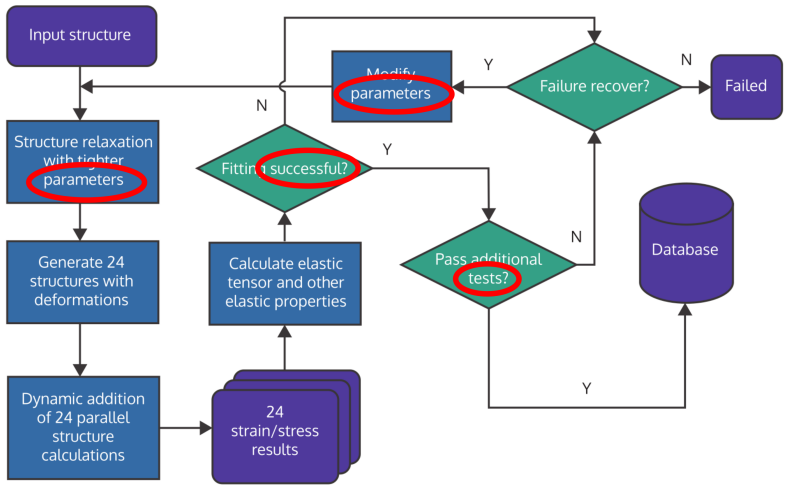
  $\Rightarrow$ Wasted resources & increased human attention (limits througput)
- Carbon footprint? More complex design spaces?
- **Goal:** Self-adapting black-box DFT algorithms
  - Parameter-free, automatically adapt to simulated system
  - Transform empirical wisdom to built-in convergence guarantees

G. Hautier Comput. Mater. Sci. **164**, 108 (2019); L. Himanen *et. al.* Adv. Science **6**, 1900808 (2019).

Sketch of high-throughput workflows



Design funnel for photovoltaic materials



Workflow for computing elasticity tensors

- Many parameters to choose (algorithms, tolerances, models)
  - Elaborate heuristics: Failure rate $\simeq 1\%$
  - Still: Thousands of failed calculations

  $\Rightarrow$ Wasted resources & increased human attention (limits througput)
- Carbon footprint? More complex design spaces?
- **Goal:** Self-adapting black-box DFT algorithms
  - Parameter-free, automatically adapt to simulated system
  - Transform empirical wisdom to built-in convergence guarantees

G. Hautier Comput. Mater. Sci. **164**, 108 (2019); L. Himanen *et. al.* Adv. Science **6**, 1900808 (2019).

Broader vision: Robust & error-controlled simulations

- Error control = Track simulation uncertainties:
  - Self-adapting simulations with mathematical guarantees
  - Integrate with error propagation efforts for surrogates[1]

  $\Rightarrow$ Byproducts: Data quality control, accelerated design[2]
- Error control = Learn missing physics:
  - Data-enhanced models, active learning
  - Integration with experiment (autonomous discovery)

  $\Rightarrow$ Exploit high-fidelity experimental, beyond-DFT data
- Error control = Leverage inexactness:
  - Error balancing: Optimal adaptive parameter selection
  - Randomised methods, selective precision (16-bit, FPGA)
  - Multi-fidelity approaches (reduced basis, surrogates)

$\Rightarrow$ Understand where and how to spend efforts best

---

[1] F. Musil, A. Grisafi *et. al.* J. Chem. Theo. Comput. **15**, 2 (2019).

[2] G. Houchins and V. Viswanathan MRS Bulletin **44**, 204 (2019).

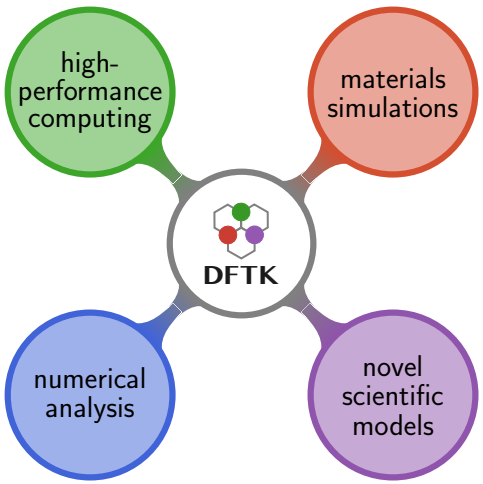
Interdisciplinary field ⇒ Multidisciplinary community

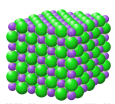
- **Mathematicians:** Toy models and unphysical edge cases
- **High-performance person:** Exploit hardware specialities
- **Scientist:** Design new models, not tweak numerics
- **Practitioner:** Reliable, black-box code, high-level interface

- State-of-the-art first-principle codes:
  - Difficult problem ⇒ Complex codes
  - Hard-coded details: Workflow, algorithms, optimisations
  - Huge code bases: 1M lines and beyond
  - Non-standard input syntax and API
  - Two-language problem: Algorithmic code hardly accessible

⇒ Innovations might not cross community boundaries

Opportunities for mathematical research





$H\Psi = E\Psi$

- Role of mathematics: Abstract and formalise
  - Simplify and unify approaches (three methods are one)
  - New point of view ⇒ New types of methods

  ⇒ Examples: SCF methods[1,2], acceleration[3–5]

  - Understand structure: Shorter & simpler implementation
  - Rapid integration of advances in computer science

  ⇒ Boost in productivity

- **DFTK**: Lower barriers for cross-disciplinary research:
  - Allows restriction to relevant model problems,
  - *and* scale-up to application regime (1000 electrons)
  - Sizeable feature set in 7000 lines of code
  - MPI, self-adapting methods, algorithmic differentiation
  - Integrated in multi-scale pipeline
    (potential fitting, molecular dynamics)

---

[1] A. Edelman, T. Arias *et. al.* J. Mat. Anal. Appl. **20**, 303 (1998).

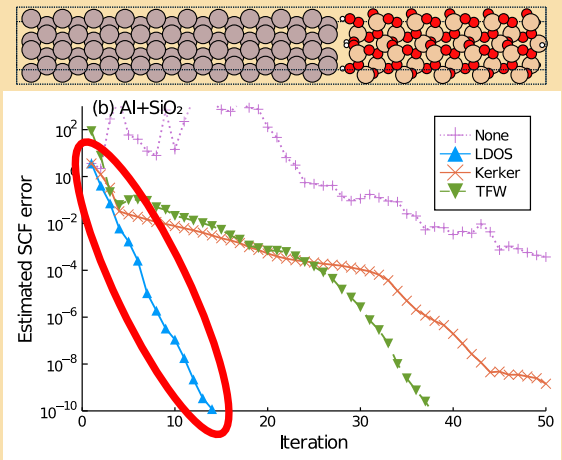
[2] E. Cancès, C. Le Bris *et. al.* Math. Model. Numer. Anal. **34**, 749 (2000).

[3] H. Fang Y. Saad Num. Lin. Alg. Appl. **16**, 197 (2009).

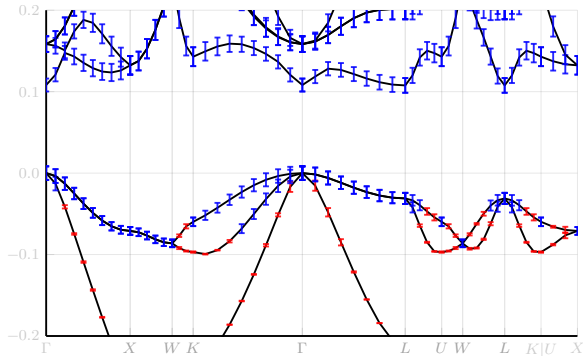
[4] H. Walker, P. Ni J. Num. Anal. **49**, 1715 (2011).

[5] M. Chupin, M. Dupuy *et. al.* Math. Model. Numer. Anal. **55** 2785 (2021).

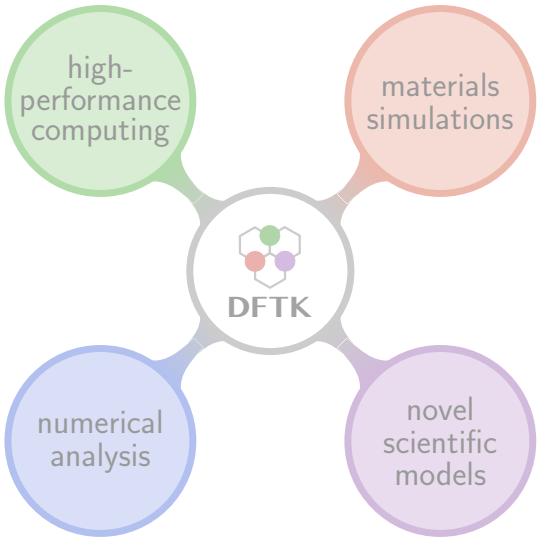
Black-box algorithms and robust error control for DFT



Self-adapting black-box DFT algorithms



Error control of DFT quantities



- Density-functional theory
- Local density of states preconditioner
- Adaptive damping

- DFTK overview
- Errors in DFT
- Model sensitivitites
- Algorithmic differentiation

Density-functional theory (insulators)

- Energy minimisation problem:

$$\min_{D\in\mathcal{P}} \mathcal{E}(D) = \min_{D\in\mathcal{P}} \left[\operatorname{tr}(H_0 D) + E_{\mathsf{Hxc}}(\operatorname{diag} D)\right]$$

with $\mathcal{P} = \left\{D \in \mathfrak{S}_1(L^2) \mid 0 \leq D \leq 1, \operatorname{tr}(D) = N, \operatorname{tr}(-\Delta D) < \infty\right\}$, $[\operatorname{diag} D](\underline{\boldsymbol{r}}) = D(\underline{\boldsymbol{r}}, \underline{\boldsymbol{r}})$

- DFT approximation: Effective single-particle model

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

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

$\Rightarrow$ Self-consistent field (SCF) problem: $V\big(\rho(V)\big) = V$ with

$$\rho(V) = \operatorname{diag}\left[\mathbb{1}_{(-\infty,\varepsilon_F]}\left(-\frac{1}{2}\Delta + V\right)\right] \qquad \text{and } \varepsilon_F \text{ s. t. } \int \rho(V) = N$$

Self-consistent field problem

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

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

- In practice: Combined with acceleration (e.g. Anderson)
  - Dropped to simplify analysis
  - Re-introduced for numerical experiments

- Near a fixed-point the error goes as

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

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

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

$\Rightarrow$ Need $P^{-1} \simeq \varepsilon^{-1}$ (matching preconditioner) or small α

Drawback of established approaches

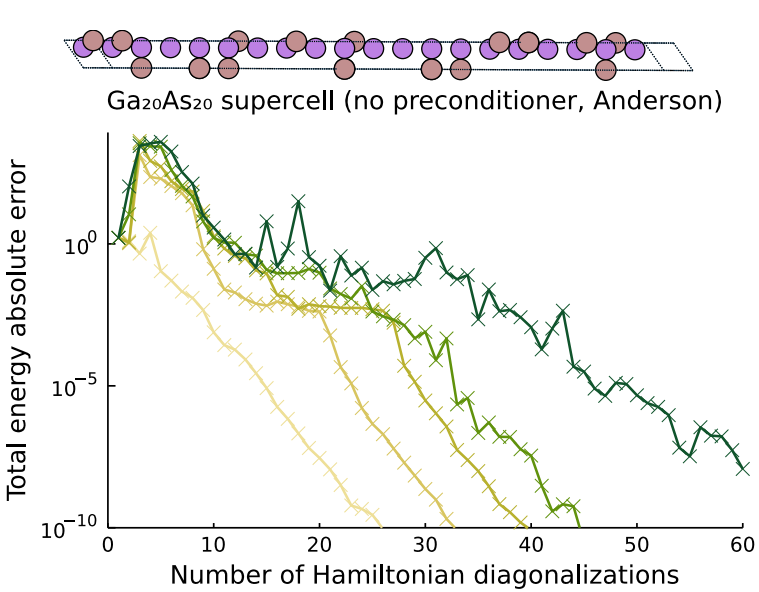
1. Preconditioner P is system-dependent and chosen *a priori*
   - Standard preconditioners: Derived from bulk materials
   - Misses important applications (e.g. inhomogeneous systems)
   - E.g. clusters, passivated surfaces, heterogeneous catalysis, ...

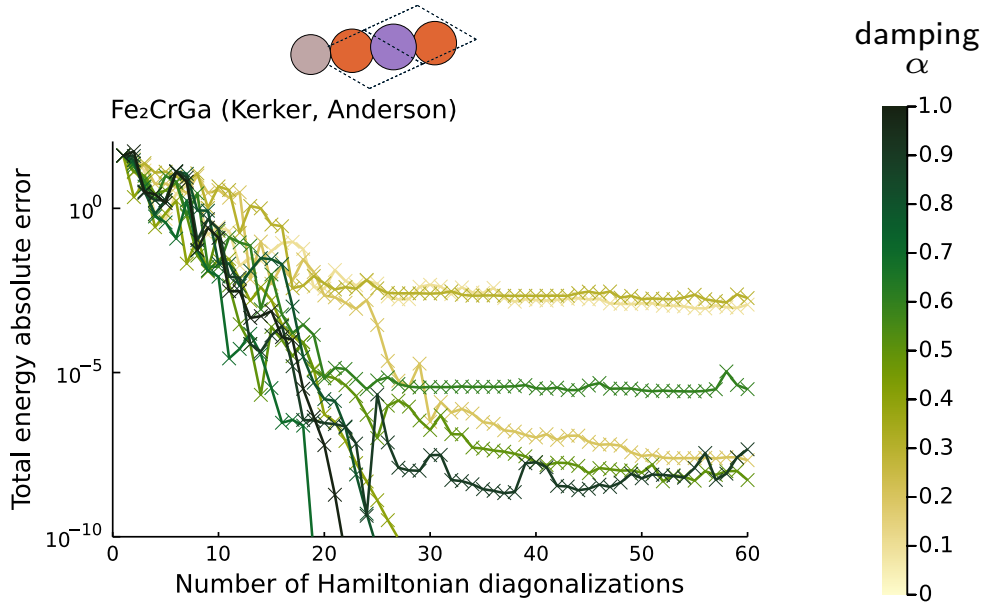
2. If no good preconditioner P known: Trial and error
   - Employ standard heuristics: E.g. **decrease damping** α
   - But: Can fail for interesting cases (the tough 1% ?)

$\Rightarrow$ Wasted computational resources

$\Rightarrow$ **Goal:** Black-box and self-adapting P and α

Illustration: Guessing a suitable damping α can be hard





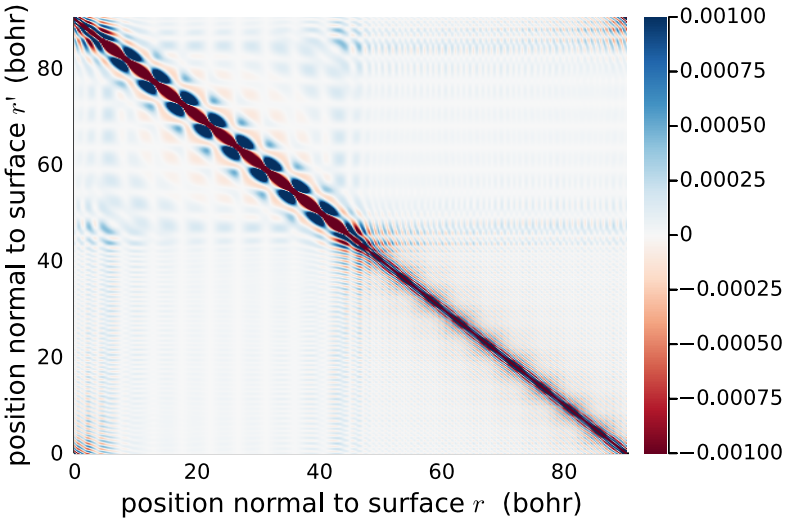
- Inefficient standard damping $(0.6 - 0.8)$
- Surprisingly small damping for smooth convergence

- Heusler alloy: Design space of interest
- High-throughput study at EPFL: Convergence difficulties

- Irregular behaviour: α versus convergence
- Heuristics breaks: Larger damping is better

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

- Bulk preconditioning models approximate inverse $P^{-1} \simeq \varepsilon^{-1}$
- Use $\varepsilon = (1 - K\chi_0)$ with $K(\rho) = V'(\rho), \chi_0(V) = \rho'(V)$
- $\chi_0(r, r')$ unit-cell internal fluctuations, diagonal dominant:



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

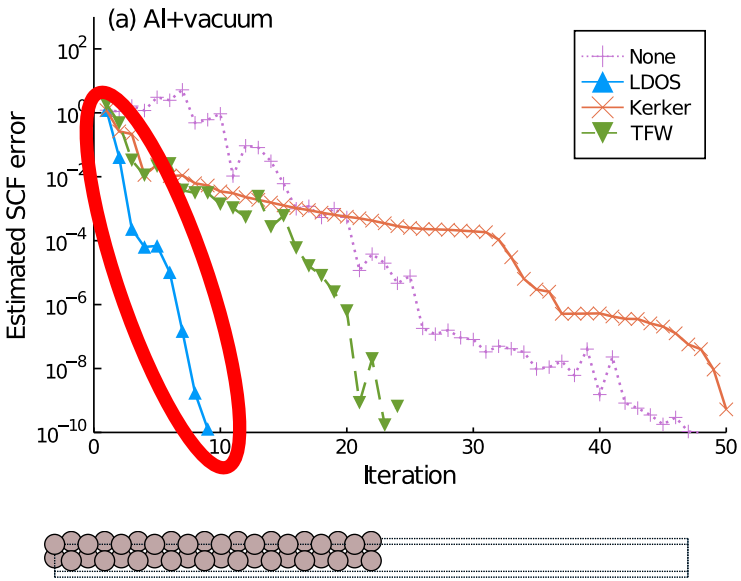
$$\chi_0(r, r') \simeq -\mathrm{LDOS}(r)\delta(r, r') \qquad \text{(homogenised } \chi_0\text{)}$$

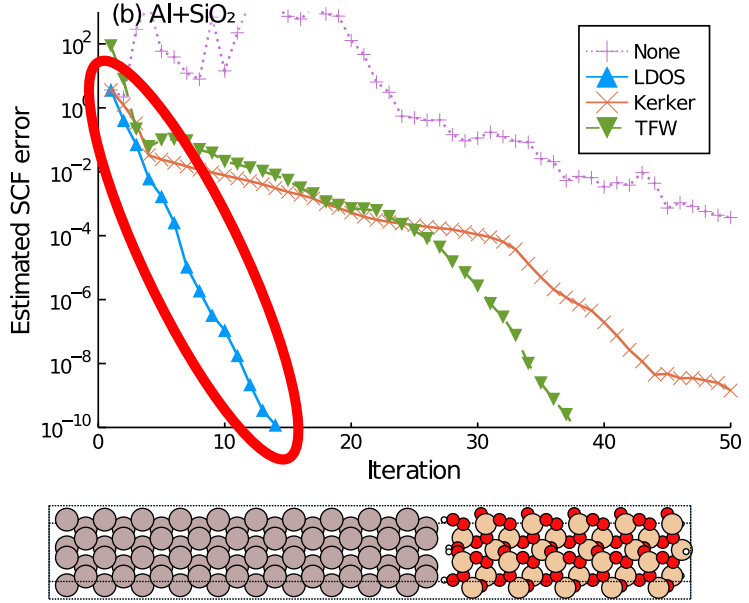
- Apply preconditioner iteratively:

$$P^{-1}V_n = [1 - K\widetilde{\chi_0})]^{-1} V_n, \qquad \widetilde{\chi_0}(r, r') = -\mathrm{LDOS}(r)\delta(r, r')$$

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

LDOS preconditioning (examples)[1]





- Inhomogeneous material: Aluminium metal + Insulator
- TFW: local Thomas-Fermi-von Weizsäcker mixing[2]
  (Ad hoc modification of metallic screening model)
- 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).

[2] D. Raczkowski, A. Canning, L. W. Wang, Phys. Rev. B. **64**, 121101 (2001).

Black-box α: Adaptive damping[1]

- Which damping α in potential mixing?
  - $V_n \to$ Find search direction δV_n (preconditioning, Anderson, . . .)
  - $V_{n+1} = V_n + \alpha\, \delta V_n$
- DFT is an energy minimisation problem $\min_V \mathcal{E}(V)$
- **Theorem (Herbst, Levitt 2022):**[1]
  - Guaranteed convergence if α small enough
- **Ingredient 1:** Backtracking line search:
  - Start from trial damping $\tilde{\alpha}$ and set $\alpha = \tilde{\alpha}$
  - Accept good steps (energy or SCF residual decreases)
  - Otherwise: Shrink α and try again
- But: Expensive step of an SCF is $\rho(V)$ (involves H diagonalisation)
  - Needed for evaluating $\mathcal{E}(V_n + \alpha\, \delta V_n)$

  $\Rightarrow$ Cost of line search step $\simeq$ cost of standard SCF step

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

Adaptive damping (2)

- **Ingredient 2:** Shrink α by approx. quadratic model for $\mathcal{E}(V_n + \alpha\,\delta V_n)$
  - Key approximation:

$$\alpha\chi_0(V_n)\delta V_n = \rho(V_n + \alpha\delta V_n) - \rho(V_n) + O(\alpha^2\,\|\delta V_n\|^2)$$

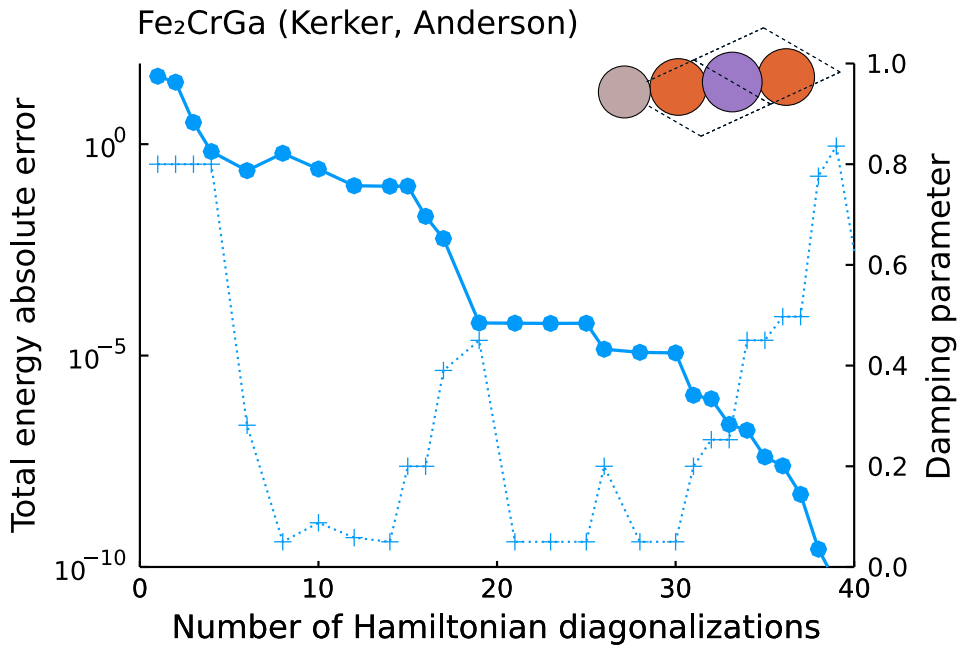
(a) Avoids costly χ_0 application (involves solving linear system)

(b) If accepted: $\rho(V_n + \alpha\delta V_n) = \rho(V_{n+1})$

$\Rightarrow$ Reuse ρ in next SCF step

- No overhead if line search immediately successful

  $\Rightarrow$ Use quadratic model also to adjust trial damping $\tilde{\alpha}$



Selected results[1]: Diagonalisations to convergence

System		fixed damping α										adaptive damping
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	
simple	Al_{40} slab	19	15	14	12	**11**	12	12	12	12	12	12
	$Ga_{20}As_{20}$ slab^N	**26**	33	40	42	45	44	70	70	65	76	26
transition metal	Fe_2CrGa	×	×	×	27	×	×	**19**	25	×	22	39
	Fe_2MnAl	×	48	×	×	×	20	21	17	16	**15**	34
	$FeNiF_6$	×	×	×	×	×	×	×	23	22	**21**	24
	Cr_{19} defect	×	×	×	74	46	48	46	**41**	47	53	48
	$Fe_{28}W_8$ bilayer	**32**	34	37	34	38	43	41	48	×	×	37

×: Energy not converged to 10^{-10} after 100 diagonalisations; ^{N}: Kerker mixing not used

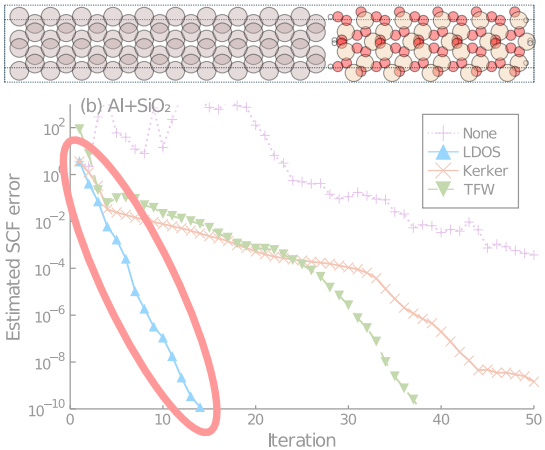
- Simple systems: Adaptive has almost no additional cost
- Transition-metal systems with challenging setup (Details: paper)
  - Successful/best α scattered ⇒ Manual selection challenging
  - Artefact of Anderson acceleration (not covered by Theorem)
  - Adaptive has overhead, but avoids trial and error

  ⇒ Mathematically motivated safeguard mechanism

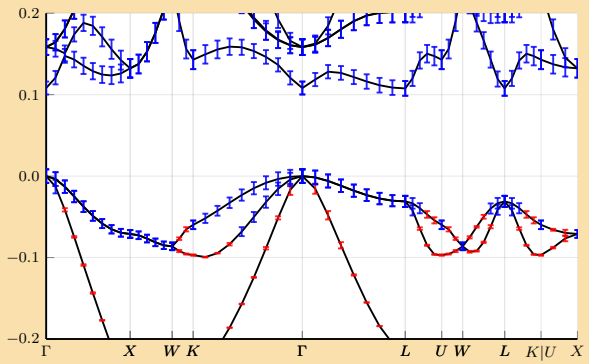
⇒ Limitations in mathematical understanding of acceleration

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

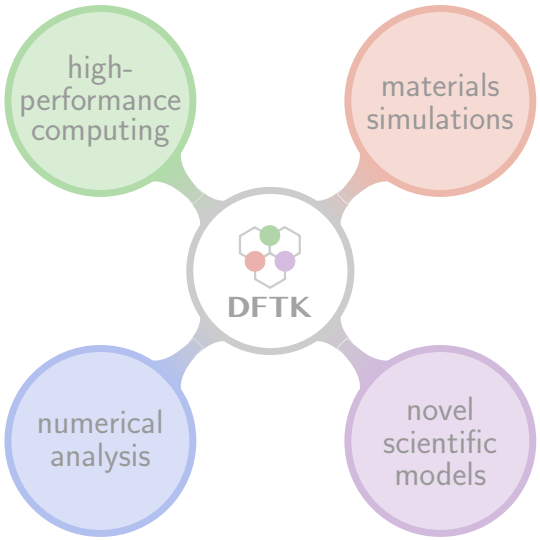
Black-box algorithms and robust error control for DFT



Self-adapting black-box DFT algorithms



Error control of DFT quantities



- Density-functional theory
- Local density of states preconditioner
- Adaptive damping

- Errors in DFT
- Model sensitivitites
- Algorithmic differentiation

- DFTK overview

Error sources in DFT simulations

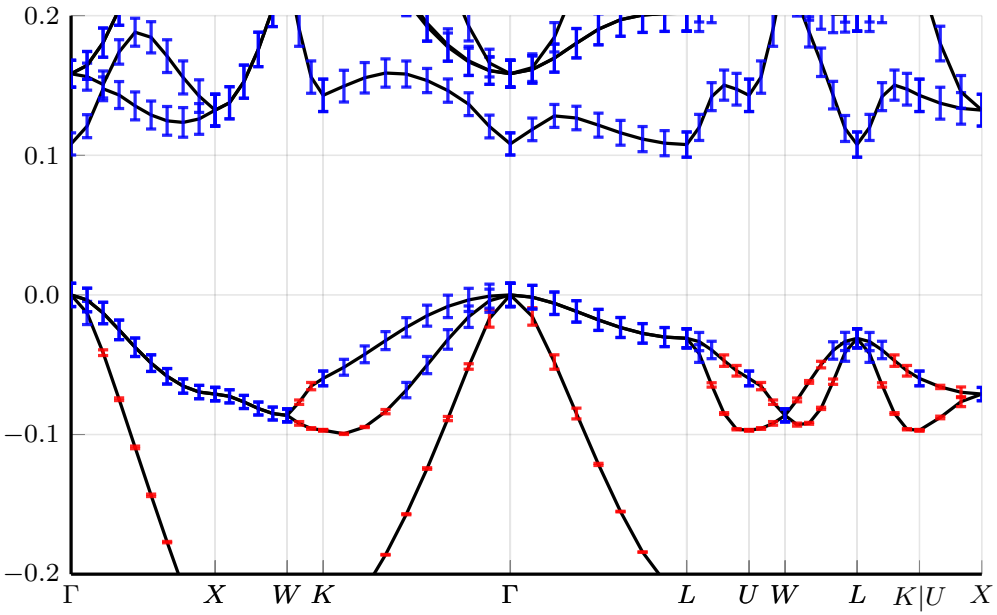
- Model error: Selection of DFT model
- Computational approach:
  - Discretisation error: Basis size, k-point mesh
  - Algorithm error: Convergence thresholds (SCF, eigensolver)
  - Floating-point error: Floating-point arithmetic
- Additionally: Programming error, hardware error (not discussed further)

- Error control: Link parameter selection $\leftrightarrow$ simulation error
  - Enables error balancing
  - Optimised automatic parameter selection

  $\Rightarrow$ Robust, parameter-free & efficient simulations

Status of error control in DFT

- Numerical analysis $\Rightarrow$ Discretisation error
  - Perturbation-based bounds for Gross-Pitaevskii[1] and DFT[2]
  - Current status: Mostly restricted setting & simplified models
  - Guaranteed bounds for band structures in a pseudopotential model[3]
  - Captures basis set error, floating-point error, convergence error



[1] E. Cancès, G. Dusson *et. al.* Comp. Rend. Math. **352**, 941 (2014).

[2] E. Cancès, G. Dusson *et. al.* arxiv 2111.01470v1.

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

Status of error control in DFT

- Numerical analysis $\Rightarrow$ Discretisation error
  - Perturbation-based bounds for Gross-Pitaevskii[1] and DFT[2]
  - Current status: Mostly restricted setting & simplified models
  - Guaranteed bounds for band structures in a pseudopotential model[3]
  - Captures basis set error, floating-point error, convergence error
- Statistical techniques $\Rightarrow$ Model error
  - Ensemble-mediated (BEEF)[4]
  - Representative comparison (Δ-test)[5]
  - Focus here: Model sensitivities by algorithmic differentiation

  $\Rightarrow$ Outlook: Efficient inference of DFT model uncertainties

---

[1] E. Cancès, G. Dusson *et. al.* Comp. Rend. Math. **352**, 941 (2014).

[2] E. Cancès, G. Dusson *et. al.* arxiv 2111.01470v1.

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

[4] V. Petzold, T. Bligaard *et. al.* Top. in Catal., **55**, 402 (2012).

[5] K. Lejaeghere, G. Bihlmayer *et. al.* Science, **351**, aad3000 (2016).

Routine computation of DFT model sensitivities

- Efficient inference strategies for uncertainty quantification:
  - Requires sensitivities wrt. DFT model

  $\Rightarrow$ Unusual, higher-order derivatives
- Combinatorial explosion:
  - "One PhD student per derivative" paradigm not feasible

  $\Rightarrow$ Use algorithmic differentiation ($\approx$ automatic derivatives)
- Illustration: Consider DFT Hamiltonian $H_{a\theta}$
  - a: Lattice constant
  - θ: DFT exchange-correlation parameters
- Self-consistent field yields fixed-point density ρ_{SCF}

$$0 = \operatorname{diag}\left[\mathbb{1}_{(-\infty,\varepsilon_F]}(H_{a\theta}(\rho_{\mathsf{SCF}}))\right] - \rho_{\mathsf{SCF}}$$

- Defines implicit function $\rho_{\mathsf{SCF}}(a,\theta)$

Computing sensitivities

- Consider model sensitivity of stress $S(a,\theta) = \frac{\partial \mathcal{E}(\rho_{\mathrm{SCF}}(a,\theta))}{\partial a}$:

$$\frac{dS}{d\theta} = \frac{\partial S}{\partial \rho_{\mathrm{SCF}}} \frac{\partial \rho_{\mathrm{SCF}}}{\partial \theta} \tag{1}$$

- Computed by implicit differentiation (response theory):

$$\frac{\partial \rho_{\mathrm{SCF}}}{\partial \theta} = [1 - \chi_0 K]^{-1} \chi_0 \frac{\partial H_{a\theta}}{\partial \theta}$$

- Parameters appear in innermost layer (model definition)
  - Each DFT model: Different derivatives $\frac{\partial H_{a\theta}}{\partial \theta}$ (can be horrible)
  - Each quantity of interest: Different sensitivity expression (1)

  $\Rightarrow$ Combinatorial explosion

- Opportunity of algorithmic differentiation (AD):
  - Generic framework for DFT derivatives / response properties
  - Saves manual coding: Request gradient (1), AD delivers

  $\Rightarrow$ New properties/derivatives by non-DFT experts!

How does algorithmic differentiation (AD) work?

```
function F(x)
    y1 = x[1] + x[2]   # F1 = sum
    y2 = 2 * p         # F2 = double
    return y2
end
```

- Goal: Compute derivative of this code
- Function $F : \mathbb{R}^2 \to \mathbb{R}$ with $F(x) = \text{double}(\text{sum}(x_1, x_2))$
- Derivative at $\tilde{x}$ is characterised by its Jacobian matrix

$$[J_F(\tilde{x})]_{ij} = \left(\frac{\partial F}{\partial x}\bigg|_{x=\tilde{x}} \right)_{ij} = \frac{\partial F_i}{\partial x_j}\bigg|_{x=\tilde{x}}$$

- Finite differences: Simple, one column at a time:

$$[J_F(\tilde{x})]_{:,j} = \frac{F(\tilde{x} + \alpha e_j) - F(\tilde{x})}{\alpha}$$

(with e_i unit vectors)

$\Rightarrow$ Inaccurate and slow ($\mathcal{O}(N)$ times primal cost)

Chain rule to the rescue!

```
function F(x)
    y1 = x[1] + x[2]   # F1 = sum
    y2 = 2 * p         # F2 = double
    return y2
end
```

$$F(x) = \mathsf{double}(\mathsf{sum}(x_1, x_2))$$

- "double" and "sum" are simple and frequent primitives

$\Rightarrow$ Key idea of AD:

  - Compose the derivative of F from the Jacobians of primitives
  - Assumed to be known and already implemented

- Use chain rule as glue, e.g. for a Jacobian element at $\tilde{x}$:

$$\frac{\partial F_i}{\partial x_j} = \frac{\partial \mathsf{double}(a)}{\partial a} \left(\frac{\partial \mathsf{sum}(c,d)}{\partial c} \frac{\partial x_1}{\partial x_j} + \frac{\partial \mathsf{sum}(c,d)}{\partial d} \frac{\partial x_2}{\partial x_j} \right)$$

- More compact: $e_i^T J_F e_j = e_i^T J_{\mathsf{double}} J_{\mathsf{sum}} e_j$
- Note: J_{double} is needed at $\mathsf{sum}(\tilde{x}_1, \tilde{x}_2)$

Forward-mode algorithmic differentiation

```
function F(x)
    y1 = x[1] + x[2]   # F1 = sum
    y2 = 2 * p         # F2 = double
    return y2
end
```

$$F(x) = \text{double}(\text{sum}(x_1, x_2))$$

$$e_i^T J_F e_j = e_i^T J_{\text{double}} J_{\text{sum}} e_j$$

- Forward-diff: Evaluate in order with *primal* F:
  1. Set $y_0 = (x_1, x_2)$, $\dot{y}_0 = e_j$
  2. Compute $y_1 = \text{sum}(y_0)$ and $\dot{y}_1 = J_{\text{sum}}(y_0)\dot{y}_0$
  3. Compute $y_2 = \text{double}(y_1)$ and $\dot{y}_2 = J_{\text{double}}(y_1)\dot{y}_1$
  4. Obtain $F(x_1, x_2)$ as y_2 and $[J_F]_{:,j} = \dot{y}_2$

$\Rightarrow$ Again one column of J_F at a time

- Implementation: Numbers $\rightarrow$ dual numbers
- Vectorisation & other tricks: Usually faster than finite diff.
- But: Still $\mathcal{O}(N)$ times primal cost

Optimal cost for differentiation (1)

```
function F(x)
    y1 = x[1] + x[2]   # F1 = sum
    y2 = 2 * p         # F2 = double
    return y2
end
```

$$F(x) = \text{double}(\text{sum}(x_1, x_2))$$
$$e_i^T J_F e_j = e_i^T J_{\text{double}} J_{\text{sum}} e_j$$

Proposition

If $f : \mathbb{R}^N \to \mathbb{R}$ is a differentiable function, computing $\nabla f = J_f$ is asymptotically not more expensive than f itself.

$\Rightarrow$ This is violated for finite diff and forward diff.

- Let's try to be more clever:
  - We could write $F(x) = b^T A x$ for appropriate (sparse) A, b
  - Equivalent formulation: $F(x) = (A^T b)^T x$
  - Differentiate that: $\nabla F = A^T b$ $\Rightarrow$ costs the same as F.
- To generalise this idea note that (for scalar functions)

$$F(x) = b^T J_F x + \mathcal{O}(x^2)$$

Optimal cost for differentiation (2)

```
function F(x)
    y1 = x[1] + x[2]   # F1 = sum
    y2 = 2 * p         # F2 = double
    return y2
end
```

$$F(x) = \text{double}(\text{sum}(x_1, x_2))$$
$$e_i^T J_F e_j = e_i^T J_{\text{double}} J_{\text{sum}} e_j$$

- Let's try to be more clever:
  - We could write $F(x) = b^T A x$ for appropriate (sparse) A, b
  - Equivalent formulation: $F(x) = (A^T b)^T x$
  - Differentiate that: $\nabla F = A^T b \;\Rightarrow$ costs the same as F.
- To generalise this idea note that (for scalar functions)

$$F(x) = b^T J_F x + \mathcal{O}(x^2) \qquad \text{with } b = e_1 = 1$$

$\Rightarrow$ Focus on computing adjoint of Jacobian:

$$e_i^T J_F e_j = \left(J_F^T e_i\right)^T e_j = \left(J_{\text{sum}}^T J_{\text{double}}^T e_i\right)^T e_j$$

Adjoint-mode algorithmic differentiation

```
function F(x)
    y1 = x[1] + x[2]   # F1 = sum
    y2 = 2 * p         # F2 = double
    return y2
end
```

$$F(x) = \text{double}(\text{sum}(x_1, x_2))$$

$$e_i^T J_F e_j = \left(J_{\text{sum}}^T J_{\text{double}}^T e_i \right)^T e_j$$

- Adjoint-mode AD: Derivative in reverse instruction order.
- *Forward pass*:
  1. Set $y_0 = (x_1, x_2)$
  2. Compute $y_1 = \text{sum}(y_0)$ and store it
  3. Compute $y_2 = \text{double}(y_1)$ and store it
- *Reverse pass*:
  1. Set $\bar{y}_2 = e_i$
  2. Compute $\bar{y}_1 = [J_{\text{double}}(y_1)]^T \bar{y}_2$
  3. Compute $\bar{y}_0 = [J_{\text{sum}}(y_0)]^T \bar{y}_1$
- Obtain $[J_F]_{i,:}$ as $\bar{y}_0^T \implies$ One row at a time

Adjoint-mode algorithmic differentiation (2)

- Given $f : \mathbb{R}^N \to \mathbb{R}$ there is only one $e_i = 1$

$\Rightarrow$ Only one reverse pass computes full gradient ∇f

$\Rightarrow$ $\mathcal{O}(1)$ times primal cost

- Many names:
  - Adjoint trick, back propagation, reverse-mode AD
- Some difficulties / challenges:
  - Reverse control flow required!
  - (Hurts your heads sometimes)
  - Storage / memory costs
  - All mutation is bad ...
- One has to be a bit more clever for iterative algorithms ...
  - Let's look at the SCF case next.

Preview: Algorithmic differentiation in DFT practice

- Optimal lattice constant: Optimal size of the unit cell:

$$a_* = \arg\min_a \mathcal{E}\left[\rho_{\mathrm{SCF}}(a, \theta)\right]$$

```
function dft_energy(a, θ)
    model = model_DFT(make_structure(a), PbeExchange(θ))
    basis = PlaneWaveBasis(model; Ecut=..., kgrid=... )
    self_consistent_field(basis).energies.total
end
optimise_lattice(θ) = optimise(a -> dft_energy(a, θ))
```

- How sensitive is a_* for a system? $\Rightarrow$ Need $\frac{da_*}{d\theta}$
- Annoyances for derivation and implementation:
  - Nested iterative methods (eigensolver, SCF, lattice optimisation)
  - Unusual second-order derivatives (e.g. $\frac{\partial S}{\partial \theta} = \frac{\partial^2 \mathcal{E}}{\partial \theta \partial a}$)
  - Will it still work for future DFT models?
    (with completely different kinds of parameters θ)
- With  **DFTK**: User needs to add one line of code

Preview: Algorithmic differentiation in DFT practice

- Optimal lattice constant: Optimal size of the unit cell:

$$a_* = \arg\min_a \mathcal{E}\left[\rho_{\mathrm{SCF}}(a, \theta)\right]$$

```
function dft_energy(a, θ)
    model = model_DFT(make_structure(a), PbeExchange(θ))
    basis = PlaneWaveBasis(model; Ecut=..., kgrid=... )
    self_consistent_field(basis).energies.total
end
optimise_lattice(θ) = optimise(a -> dft_energy(a, θ))

sensitivities =
    ForwardDiff.gradient(optimise_lattice, [κ, β])
```

- How sensitive is a_* for a system? $\Rightarrow$ Need $\frac{da_*}{d\theta}$
- Annoyances for derivation and implementation:
  - Nested iterative methods (eigensolver, SCF, lattice optimisation)
  - Unusual second-order derivatives (e.g. $\frac{\partial S}{\partial \theta} = \frac{\partial^2 \mathcal{E}}{\partial \theta \partial a}$)
  - Will it still work for future DFT models?
    (with completely different kinds of parameters θ)
- With **DFTK**: User needs to add one line of code

Preview: Lattice constant sensitivities of silicon

```
function dft_energy(a, θ)
    model = model_DFT(make_structure(a), PbeExchange(θ))
    basis = PlaneWaveBasis(model; Ecut=..., kgrid=... )
    self_consistent_field(basis).energies.total
end
optimise_lattice(θ) = optimise(a -> dft_energy(a, θ))

sensitivities =
    ForwardDiff.gradient(optimise_lattice, [κ, β])
```

(Å)	a_*	κ	$\frac{da_*}{d\kappa}$	β	$\frac{da_*}{d\beta}$
expmnt.	5.421				
PBEsol	5.449	0.804	0.713	0.0375	0.0058
PBE	5.461	0.804	0.550	0.0667	0.0194
APBE	5.465	0.804	0.482	0.0790	0.0269
PBEmol	5.467	0.804	0.456	0.0838	0.0301
XPBE	5.466	0.920	0.603	0.0706	0.0184
rev-PBE	5.467	1.245	0.744	0.0667	0.0099

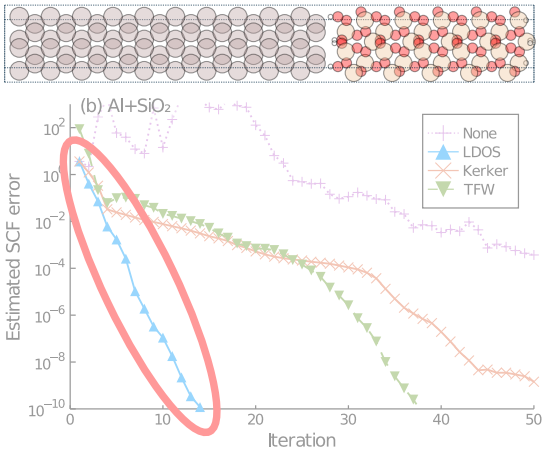
Model sensitivities for the silicon lattice constant

- Generic framework building on:
  - Flexible  **DFTK** architecture (floating-point agnostic)
  - julia tools for algorithmic differentiation
  - Extra work: Generic response implementation in  **DFTK**
- Fully flexible in DFT model or targeted quantity:
  - Only XC energy expression & SCF postprocessing code needed
- Ongoing: Adjoint-mode AD implementation:
  - All parameter sensitivities by a single response problem

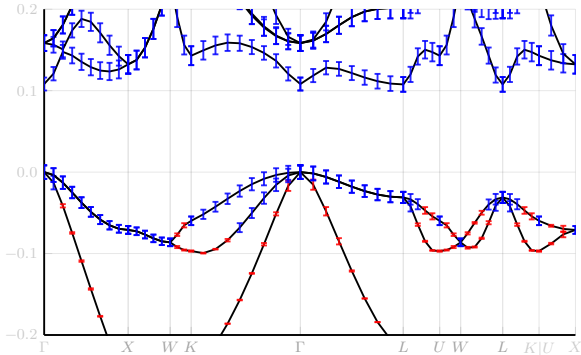
  $\Rightarrow$ Routine computation of model sensitivities

  $\Rightarrow$ Machine-learned solid-state XC models (where θ high-dimensional)

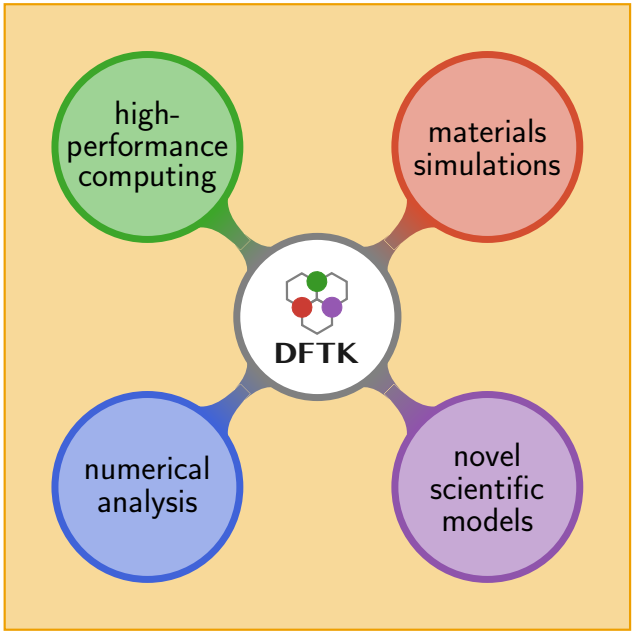
Black-box algorithms and robust error control for DFT



Self-adapting black-box DFT algorithms

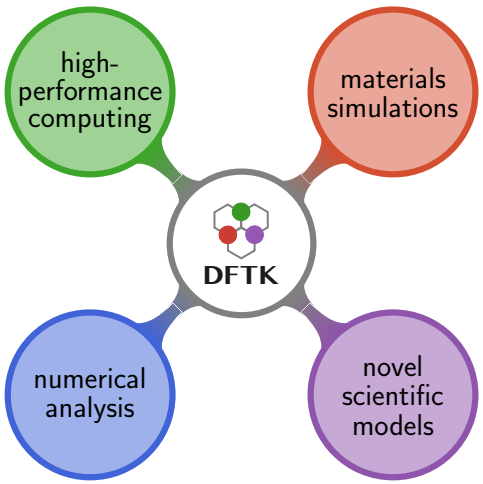


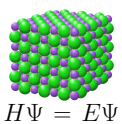
Error control of DFT quantities



- Density-functional theory
- Local density of states preconditioner
- Adaptive damping
- Errors in DFT
- Model sensitivitites
- Algorithmic differentiation
- **DFTK** overview

Density-functional toolkit[1] — https://dftk.org





- julia code for plane-wave DFT, started in 2019
- Fully composable with julia ecosystem:
  - Arbitrary precision (32bit, >64bit, . . .)
  - Algorithmic differentiation (AD)
- Key tool in all presented research:
  - Mathematical analysis (GPE, reduced models)
  - Scale-up to applications ($\simeq 1000$ electrons)
  - Features incl. meta-GGA, response, MPI parallelisation
  - Speed within factor 2–4 to established codes

$\Rightarrow$ Build to enable multidisciplinary synergies

- Low entrance barrier across backgrounds:
  - Only 7000 lines of code, open-source components
  - Avoids two-language problem: Just julia
- High-productivity research framework:
  - 10 weeks to submit band structure error paper
  - GSoC student (10 weeks) for initial AD support

DFTK: Bringing mathematical research to the applications

- Self-adapting black-box DFT methods[a,b]
- Numerical analysis of DFT[c]
- Practical error bounds[d,e]

- Exploring algorithmic differentiation:
  - "Automatic response": Phonons & higher-order properties
  - Full AD-able simulation pipeline: DFT, potentials, MD
- Uncertainty quantification all the way: DFT, potentials, MD
- Approximate computing on modern GPUs

- Outreach and teaching
  - Community building: julia-based first-principle ecosystem
  - Lecture: Mathematics of computational chemistry

Growing user base:

MIT
KU The University of Kansas
ASU Arizona State University
Carnegie Mellon University
École des Ponts ParisTech
Inria
UBFC Université Bourgogne Franche-Comté
서울대학교 Seoul National University
RWTH Aachen University
TUM

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

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

[c] E. Cancès, G. Kemlin *et. al.* J. Matrix Anal. Appl., **42**, 243 (2021).

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

[e] E. Cancès, G. Dusson *et. al.* arxiv 2111.01470v1.

Summary

- High-throughput screening
  - Main obstacle: Large number of parameters
  - Chosen empirically ⇒ Reliability limited
- Black-box strategies for damping & preconditioning
  - Build on combining mathematical and physical insight
  - Safeguard mechanism: Increase robustness for hard cases
  - Readily available in DFTK
- Algorithmic differentiation
  - Routine computation of model sensitivities
  - First step towards data-enhanced DFT models
- Reliable estimates for numerical errors
  - Promising developments on a practical scale
- DFTK: Multidisciplinary software development
  - julia-based framework for new DFT algorithms
  - In one code: Reduced problems and scale-up to realistic applications
  - High-productivity research framework

Acknowledgements

https://michael-herbst.com/talks/2022.06.10_bbox_algos_ad.pdf

École des Ponts
Antoine Levitt
Eric Cancès

RWTH
Benjamin Stamm
Markus Towara

EPFL
Marnik Bercx
Nicola Marzari

MIT
Jeremiah DeGreeff
Katharine Fisher
Youssef Marzouk
Emmanuel Luján

TU Berlin
Niklas Schmitz

all DFTK contributors

RWTH AACHEN UNIVERSITY

Inria

Summer of code

julia

MIT

CESMIX

Questions?

https://michael-herbst.com/talks/2022.06.10_bbox_algos_ad.pdf

mfherbst

herbst@acom.rwth-aachen.de

https://michael-herbst.com/blog

DFTK https://dftk.org

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

Applied and Computational Mathematics

RWTH AACHEN UNIVERSITY