Materials modeling: Building bonds across atoms, code and people

Michael F. Herbst[1] and Rachel C. Kurchin[a]

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

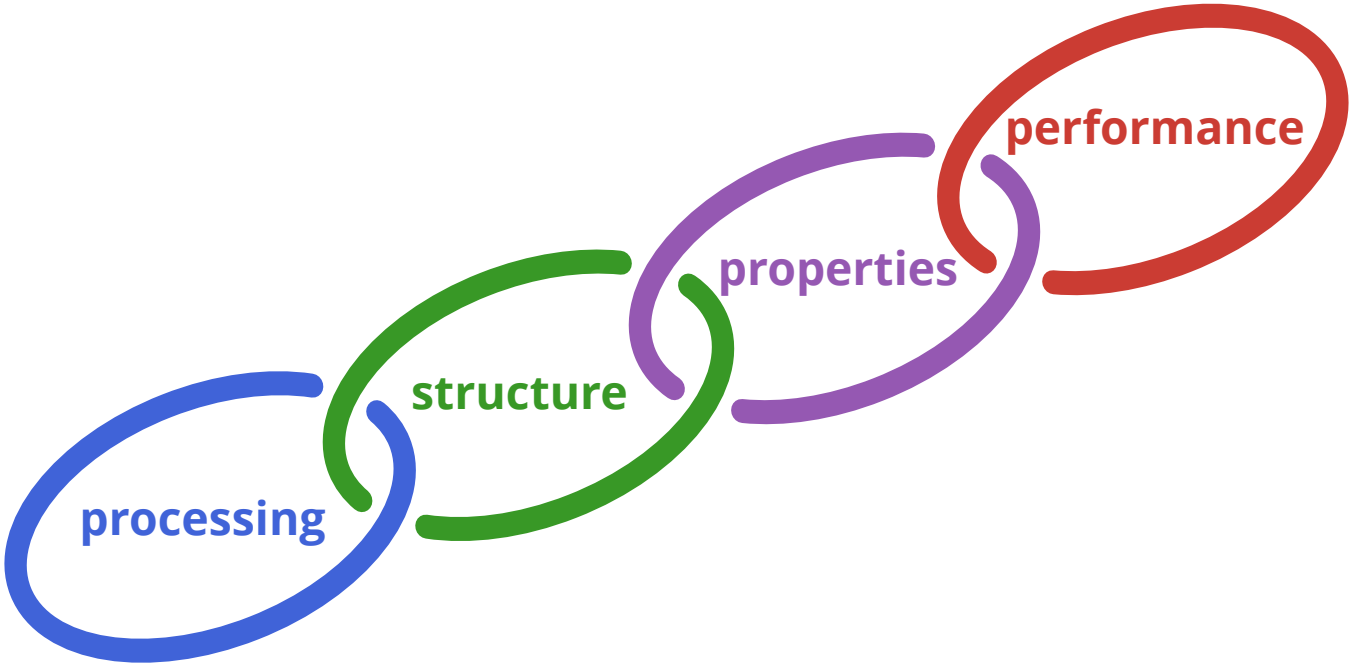
a. Accelerated Computation of Materials for Energy (acme-group-cmu.github.io), Carnegie Mellon University

10 July 2024

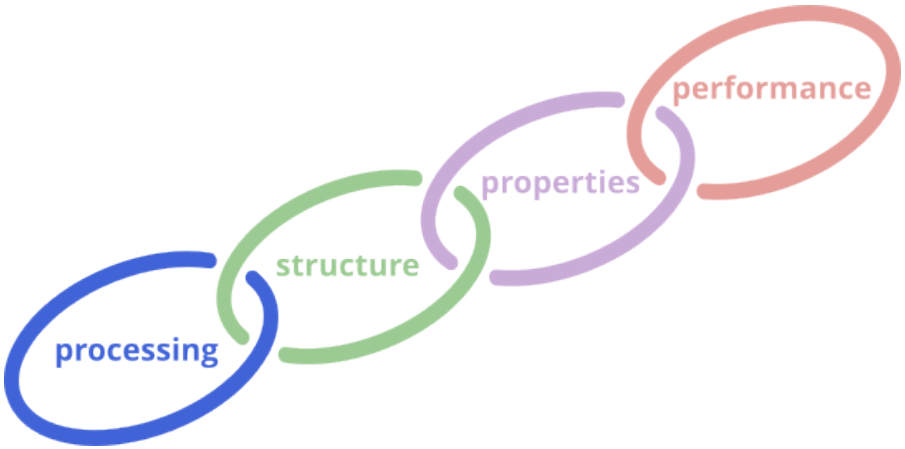
https://michael-herbst.com/slides/juliacon2024

What is Materials Science and Engineering?

MSE is about understanding (and leveraging) relationships between key attributes of materials:



MSE example: Cheese! (processing)



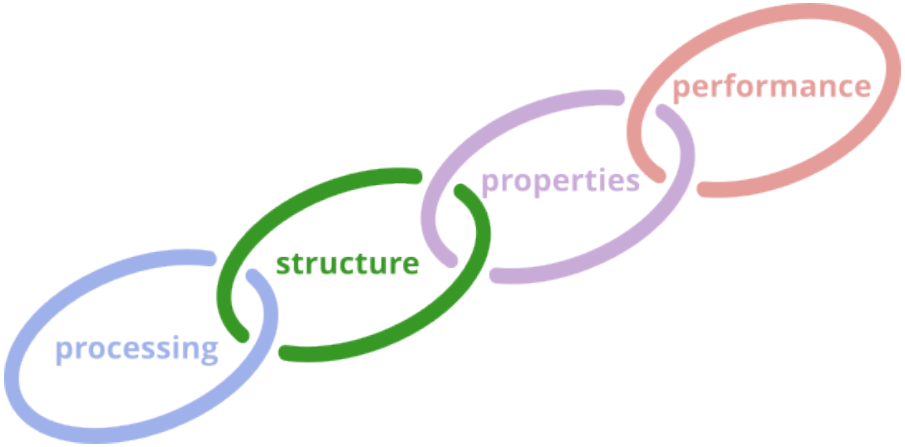
 [1]

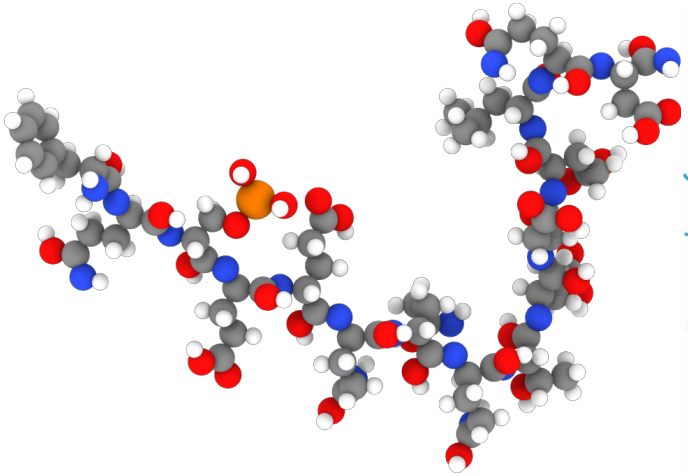
 [2]

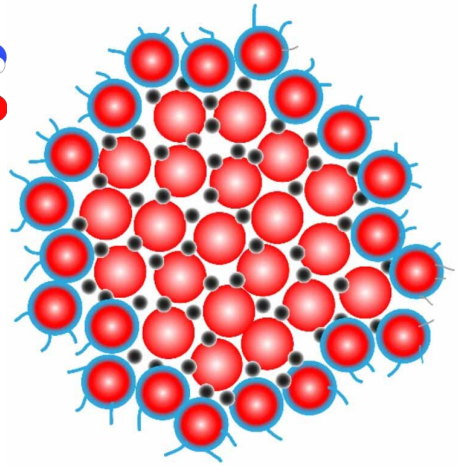
[1] Photo from https://www.dairyfoods.com/articles/96116-moving-towards-carbon-neutral-cheesemaking

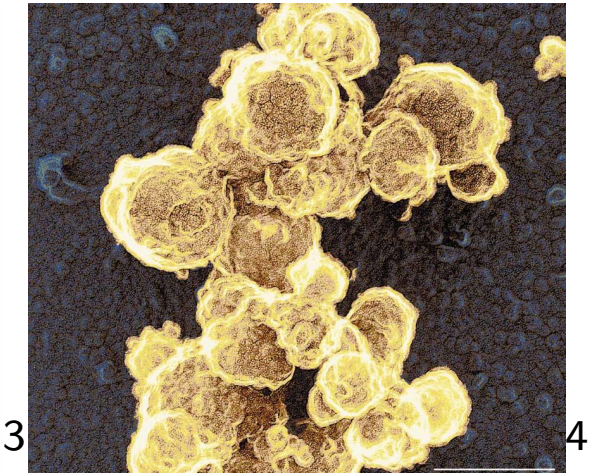
[2] Photo from https://www.cheeseprofessor.com/blog/brooklyn-cheese-caves

MSE example: Cheese! (structure)





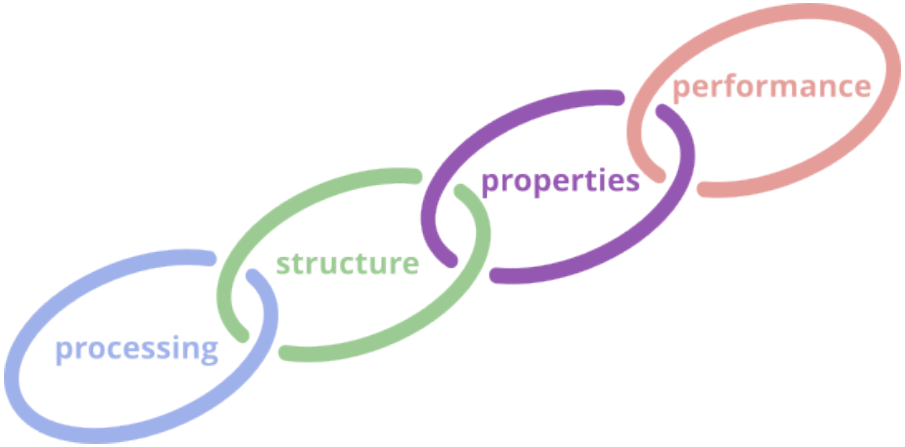


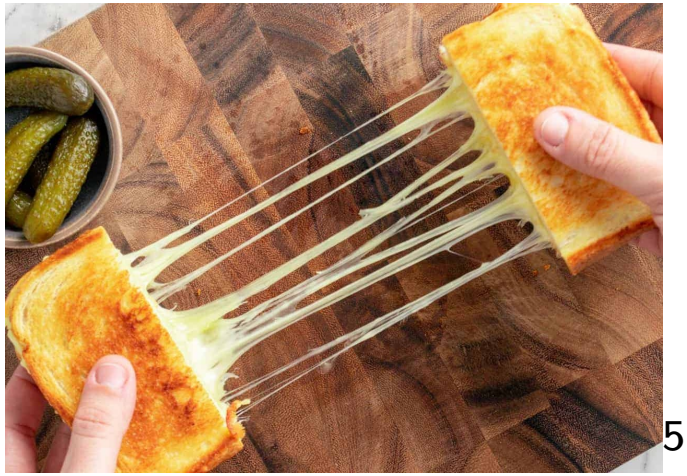
[3]  [4]

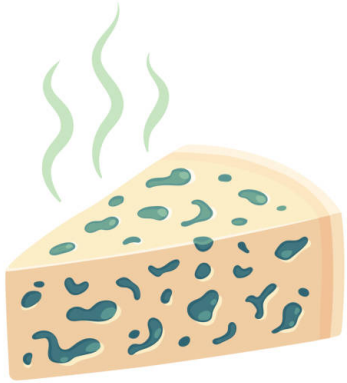
[3] Image from DOI: 10.1088/2515-7639/aba2b7

[4] Image from DOI: 10.22443/rms.inf.1.55

MSE example: Cheese! (properties)



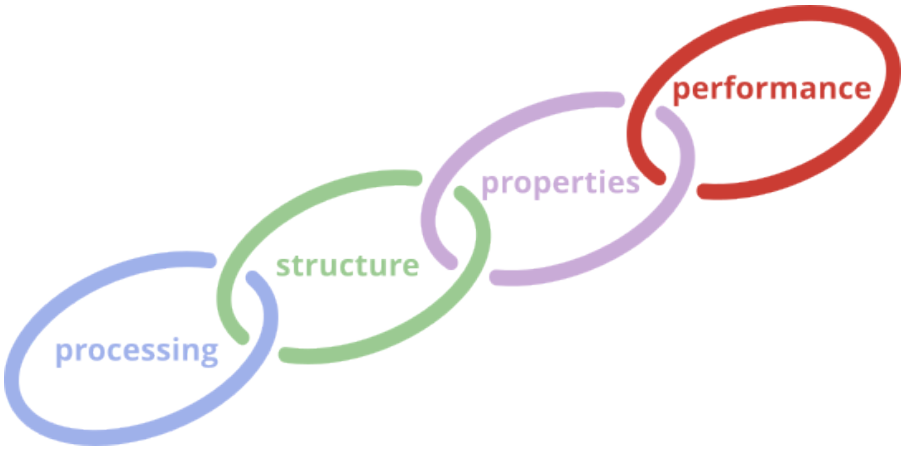

[5]


[6]

[5] Photo from `https://thescranline.com/ultimate-grilled-cheese/`

[6] Art from Sudowoodo on iStock

MSE example: Cheese! (performance)




[7][8]

[7] Photo from `https://www.thetimes.com/life-style/health-fitness/article/are-you-a-cheese-addict-time-to-quit-ph3thqvcd`

[8] Did I chose this photo because the source URL was funny? Mayyybe...

What is Materials Science and Engineering?

What do we mean by a material?
$\rightarrow$ basically, anything solid or solid-ish

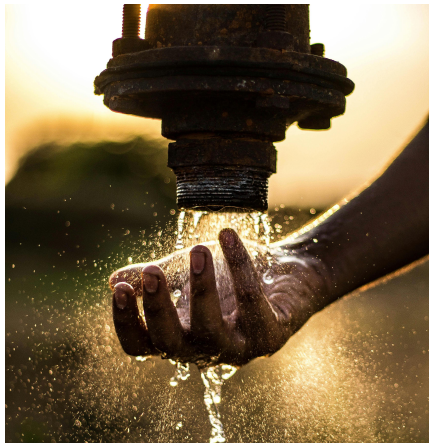
This includes many materials classes such as metals, ceramics, polymers, and composites, which can be atomically ordered (crystalline) or not, as well as exist in a variety of geometries (bulk, films, nanostructures...)



What is Materials Science and Engineering?

MSE relies/builds on concepts from physics, chemistry, and engineering fields (mechanical, chemical, electrical, ...) – the field is highly interdisciplinary!

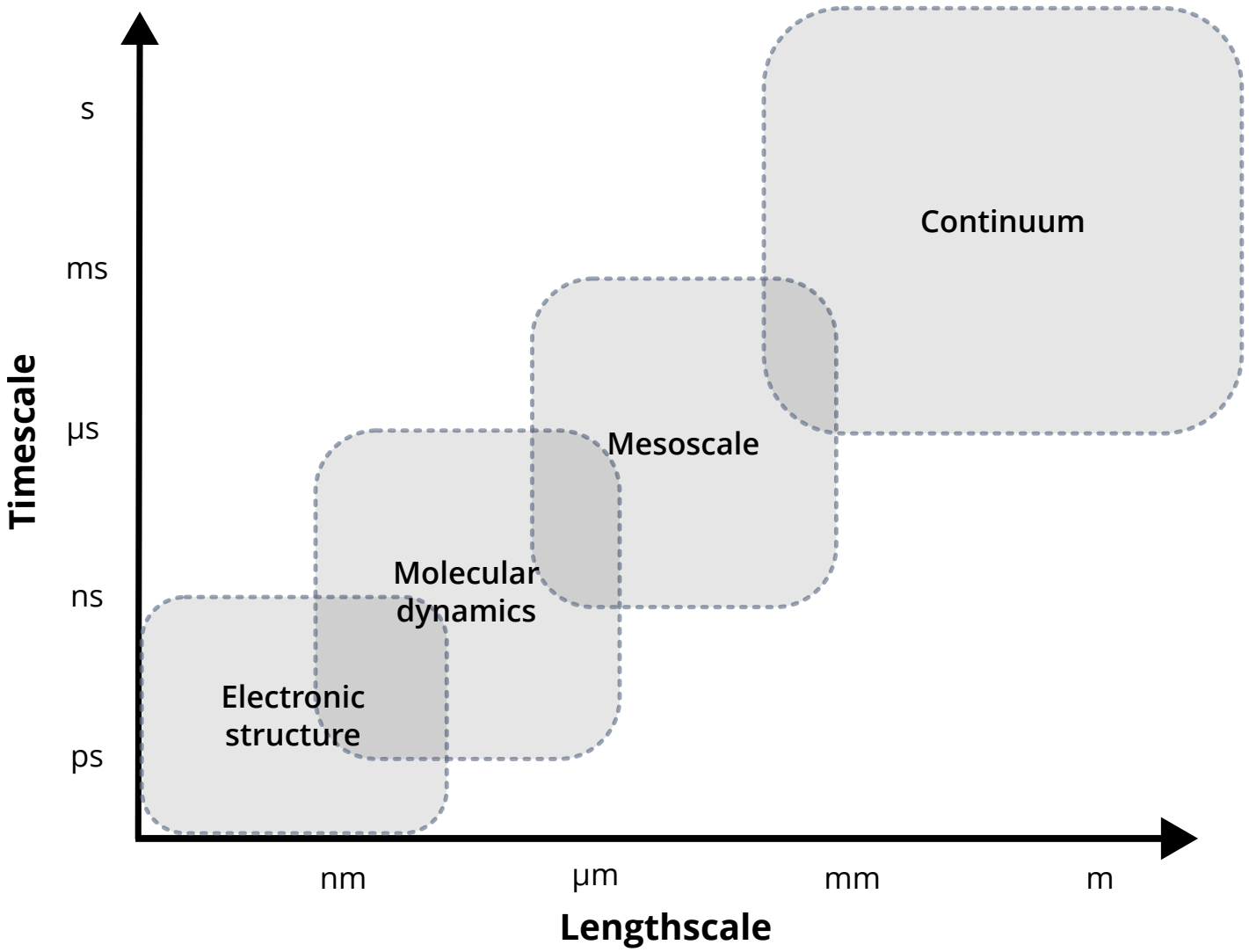
New/better materials are also crucial to the innovations that will enable society to address the grand challenges of the 21st century, including fighting the climate crisis, improving global health, and enabling next-generation technologies!



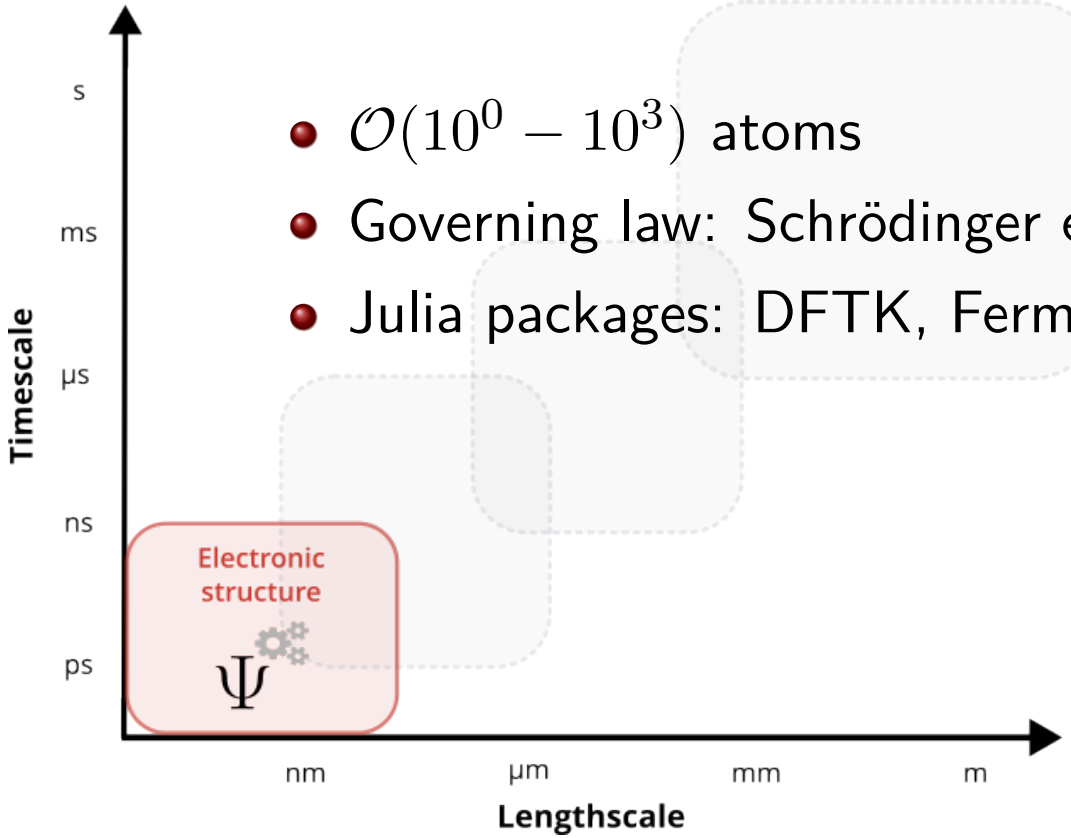
Computational materials science

- Modeling materials through computer solutions of the governing equations for physical models that describe them
- "Computational experiments" can offer valuable complementarity to lab experiments and pencil/paper theory
- Computational MSE offers tremendous value for acceleration of materials design/discovery/optimization

Computational materials science



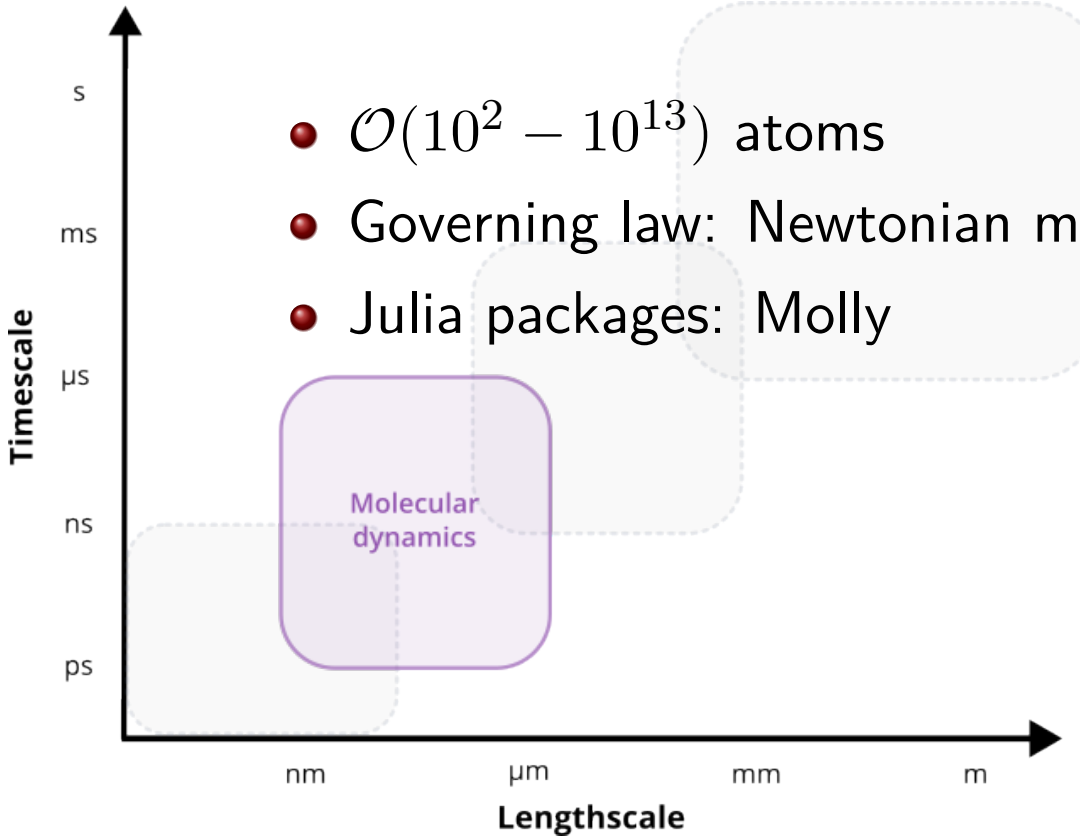
Computational materials science: Electronic structure



- $\mathcal{O}(10^0 - 10^3)$ atoms
- Governing law: Schrödinger equation
- Julia packages: DFTK, Fermi

Key math/numerics ideas: nonlinear complex-valued eigenvalue problems, fixed-point methods, preconditioners, constrained non-convex optimization, ...

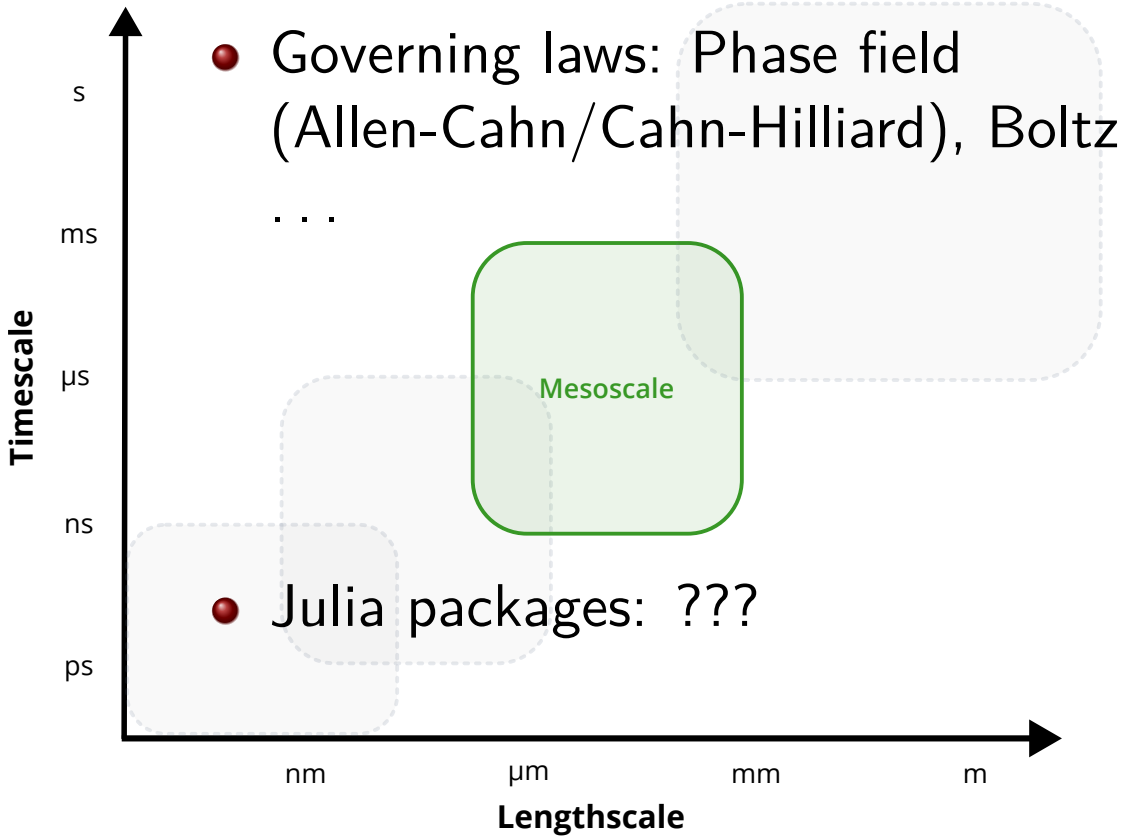
Computational materials science: Molecular dynamics



- $\mathcal{O}(10^2 - 10^{13})$ atoms
- Governing law: Newtonian mechanics
- Julia packages: Molly

Key math/numerics ideas: massive systems of coupled ODE's (with symplectic integration), control schemes, constrained optimization, high-dimensional statistics and parameter inference, dynamical systems, ...

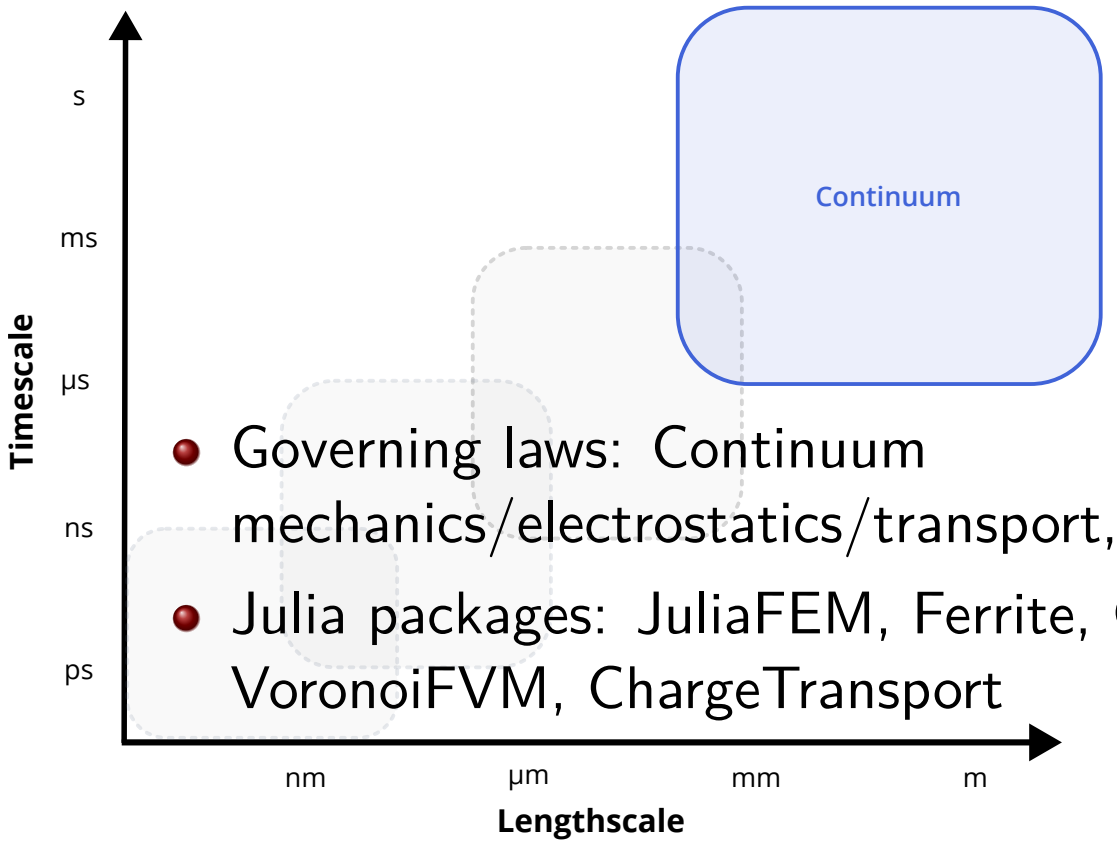
Computational materials science: Mesoscale modeling



- Governing laws: Phase field (Allen-Cahn/Cahn-Hilliard), Boltzmann Transport, ...
- Julia packages: ???

Key math/numerics ideas: partial differential equations, adaptive time-stepping, Monte Carlo, ...

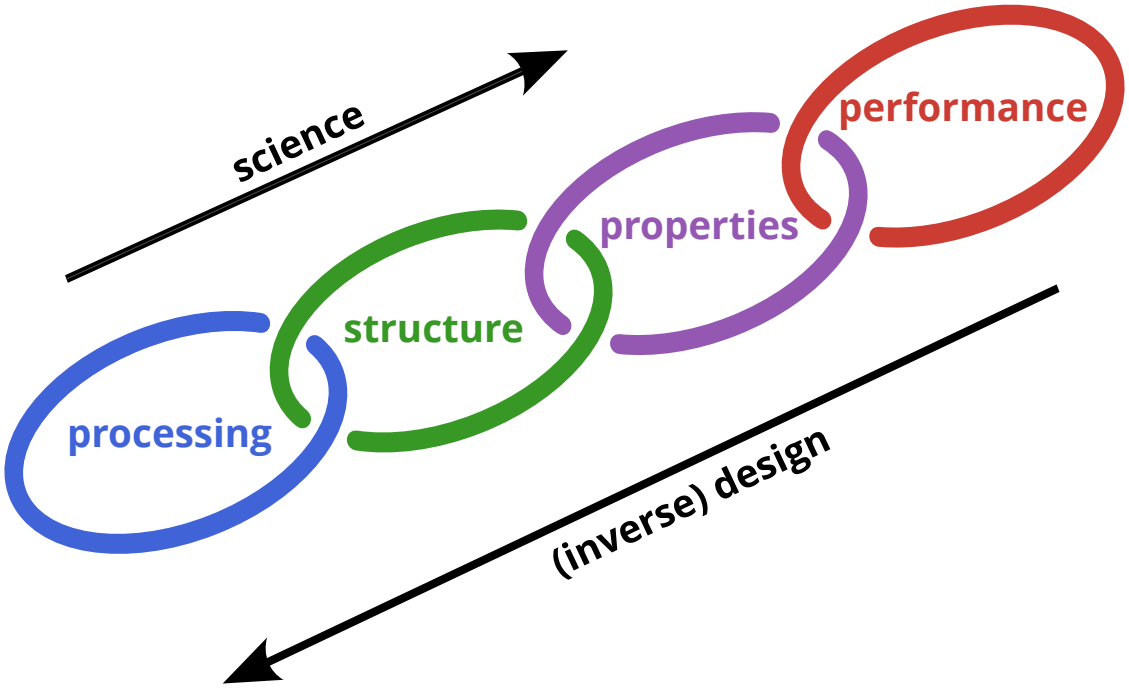
Computational materials science: Continuum modeling



- Governing laws: Continuum mechanics/electrostatics/transport, thermodynamics
- Julia packages: JuliaFEM, Ferrite, Gridap, JuliaSim, VoronoiFVM, ChargeTransport

Key math/numerics ideas: partial differential equations, finite elements, meshing, ...

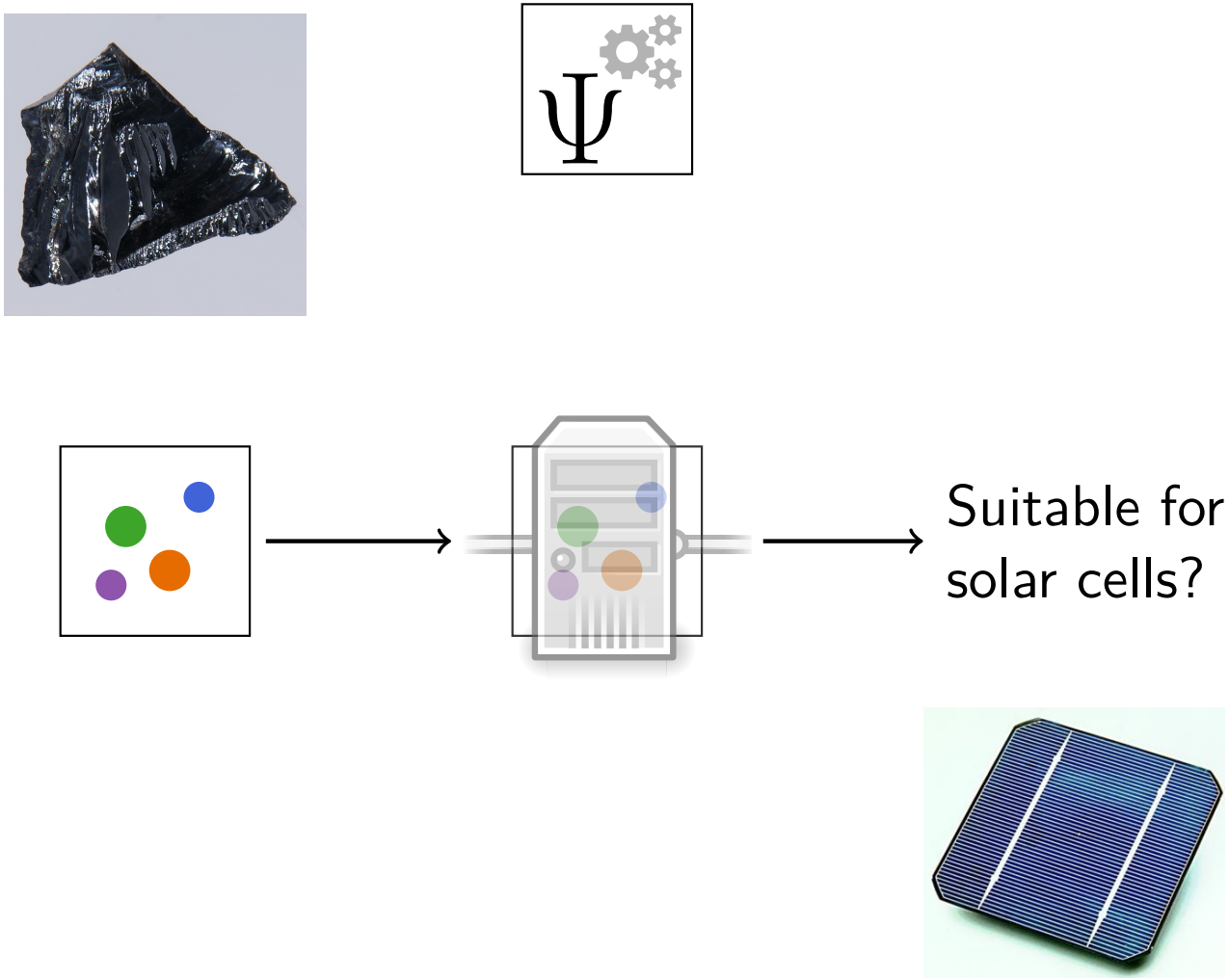
Towards materials design



Computation has a key enabling role to play in materials discovery/design due to slow speed and high monetary/energetic cost of experiment (1 fume hood $\simeq$ 2-3 average households[9])

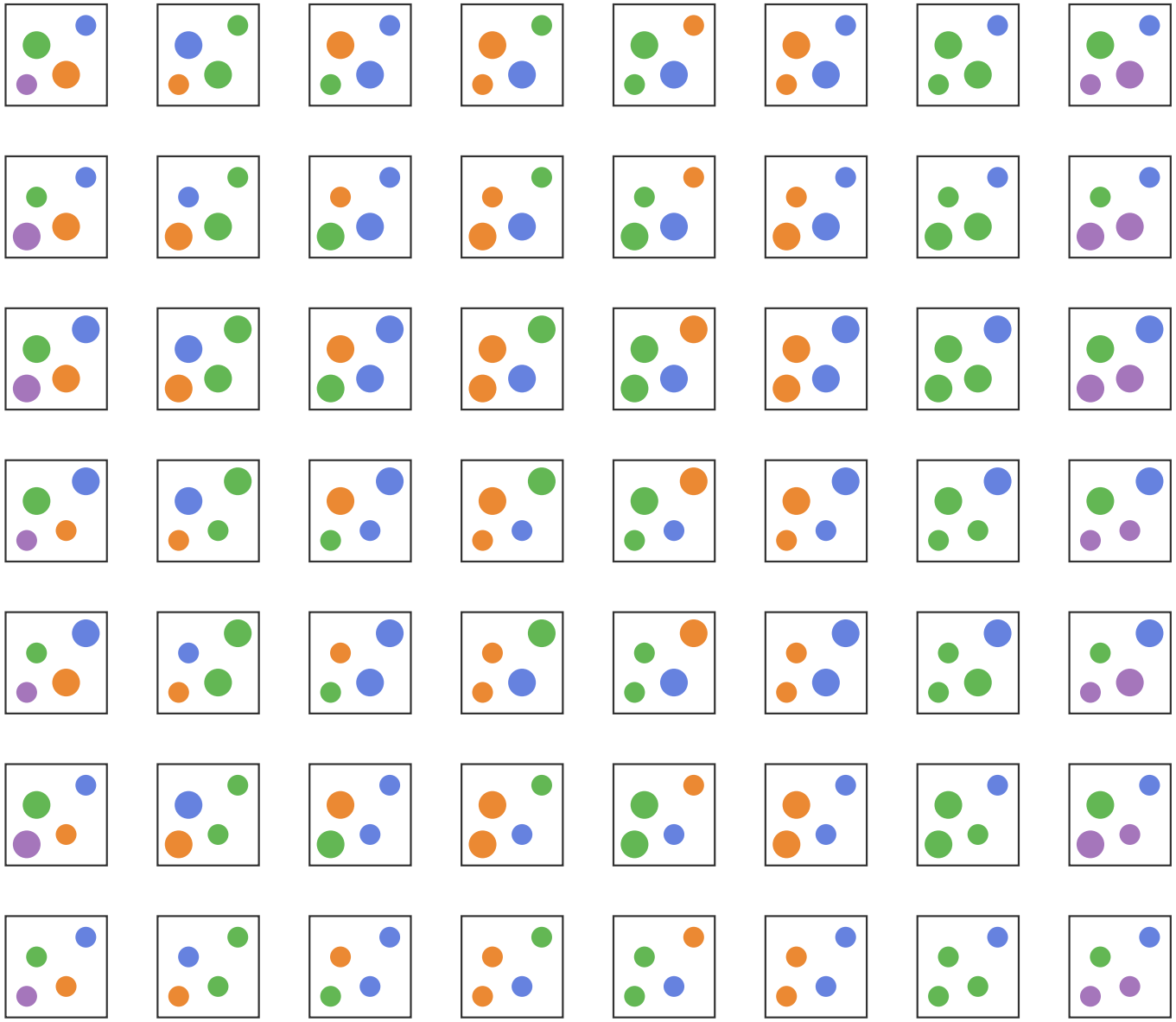
[9] D. Wesolowski *et. al.* Int. J. Sustain. High. Edu. **11**, 217 (2010).

High-throughput materials screening



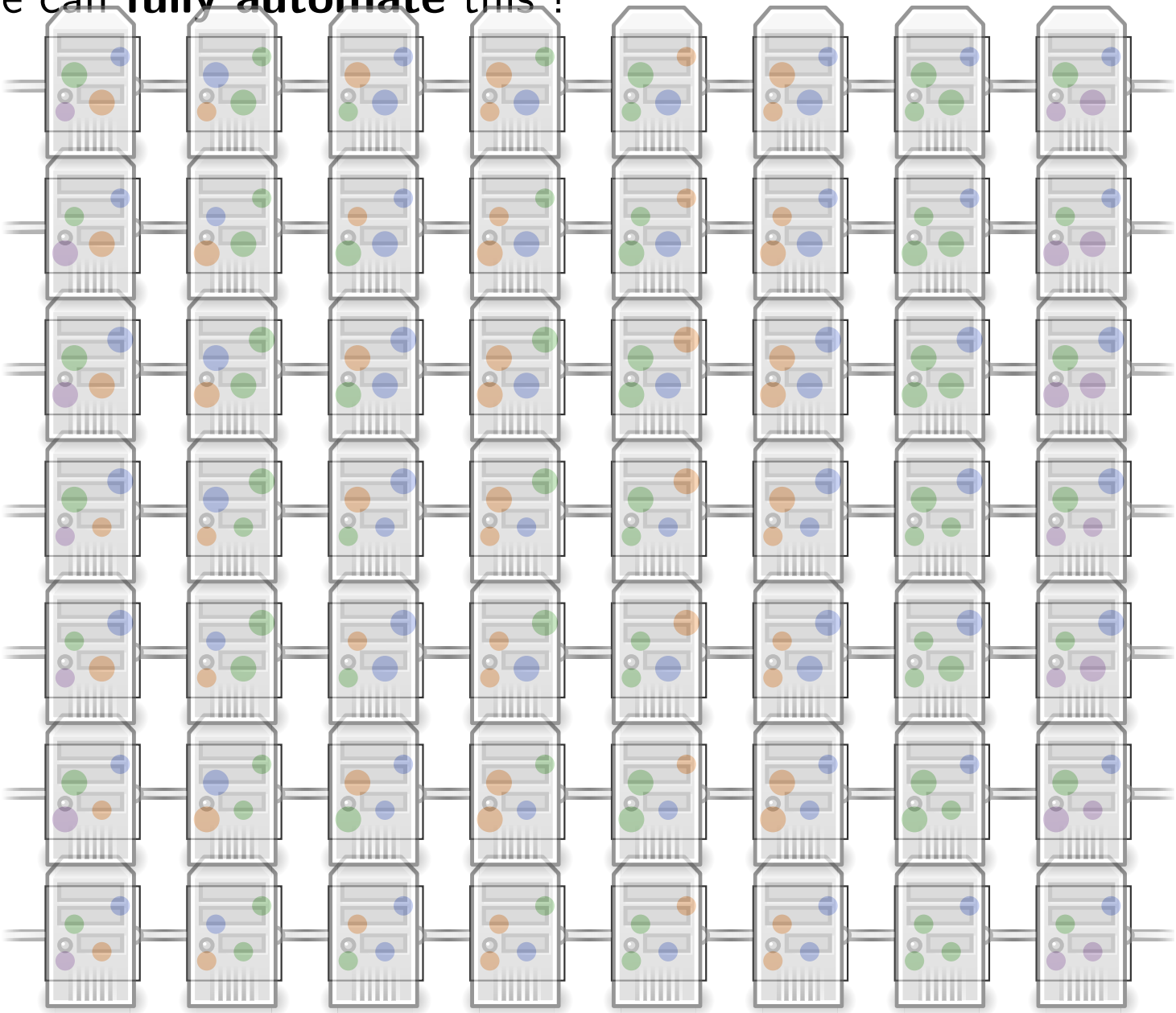
High-throughput materials screening

- We can **fully automate** this !

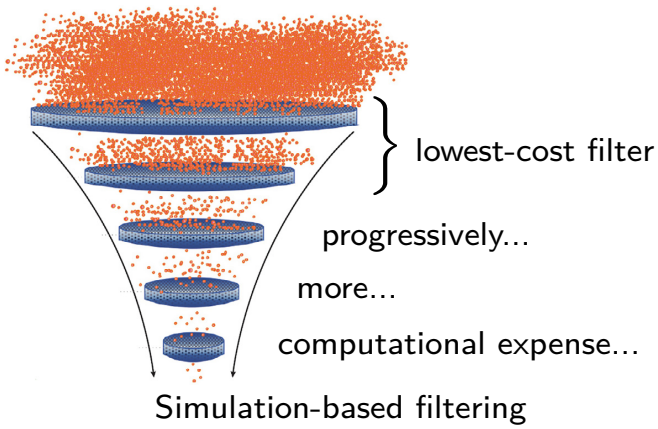


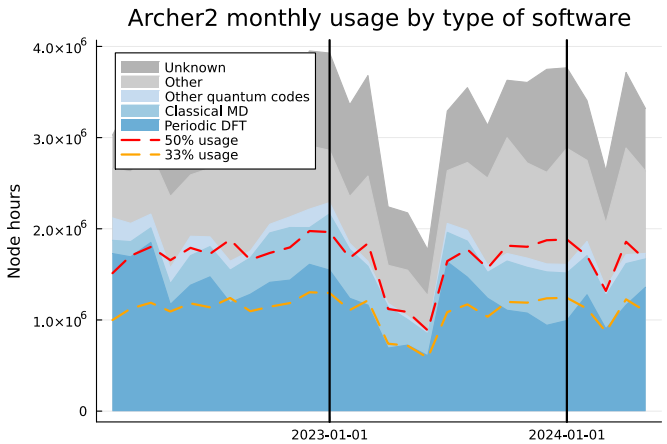
High-throughput materials screening

- We can **fully automate** this !



Computational materials discovery





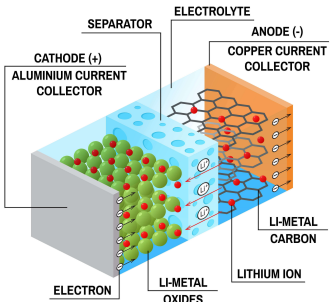
- Goal: Only promising candidates made in the lab
- Systematic simulations on $\simeq 10^4 - 10^6$ compounds
  - Noteworthy share of world's supercomputing resources
- This approach (and computational MSE more broadly) is used extensively in industrial contexts as well as academia!

AIRBUS BASF BOSCH CITRINE
Google GORE-TEX® PRODUCTS Meta Microsoft

Contents

Computing complex properties

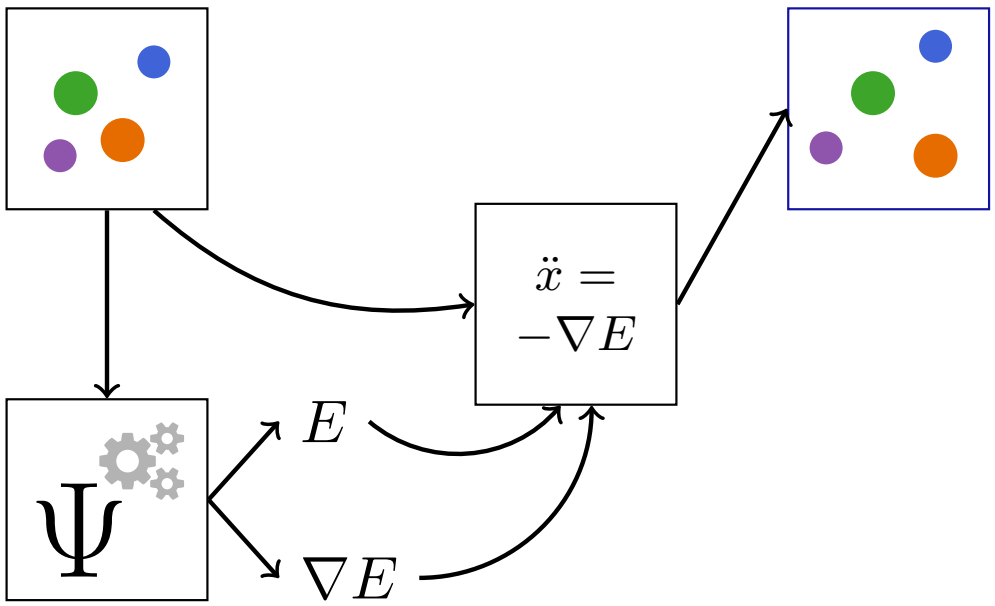
Melting point



(Ionic) transport

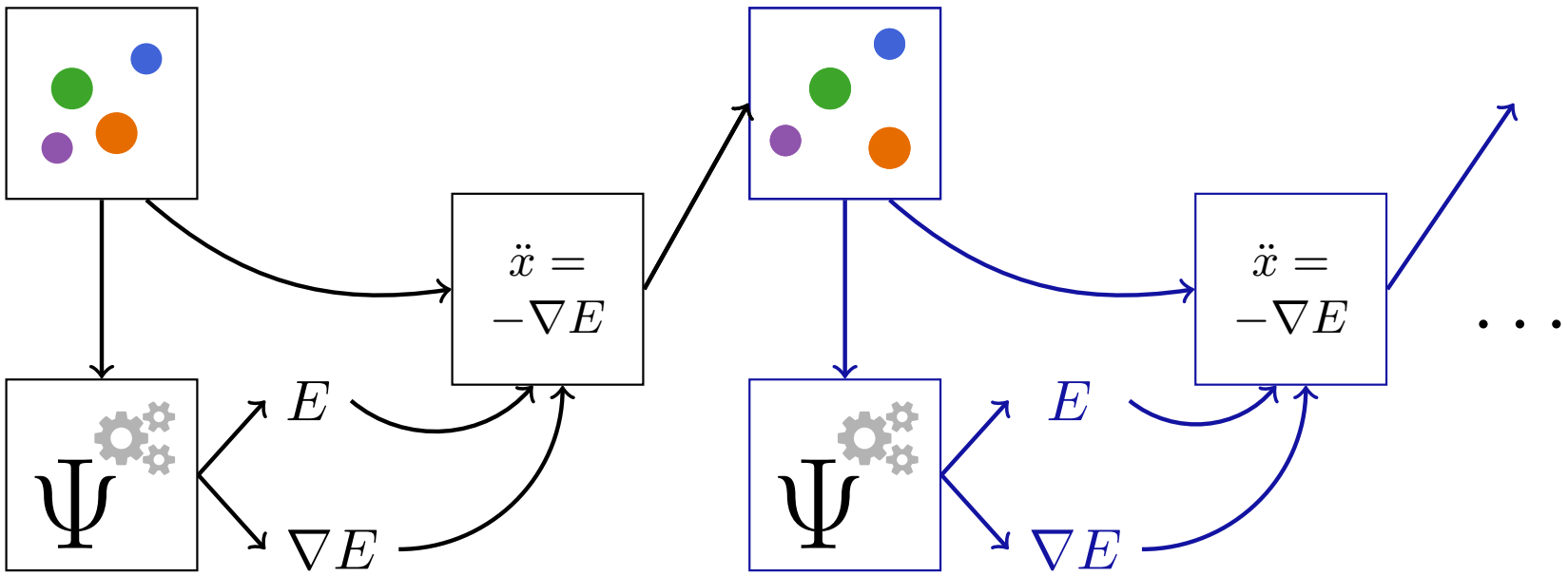
- Dynamics important for many properties: Involved workflows
- Example: Aluminium crystal (low-end of the scale)

Aluminium dynamics: One time step



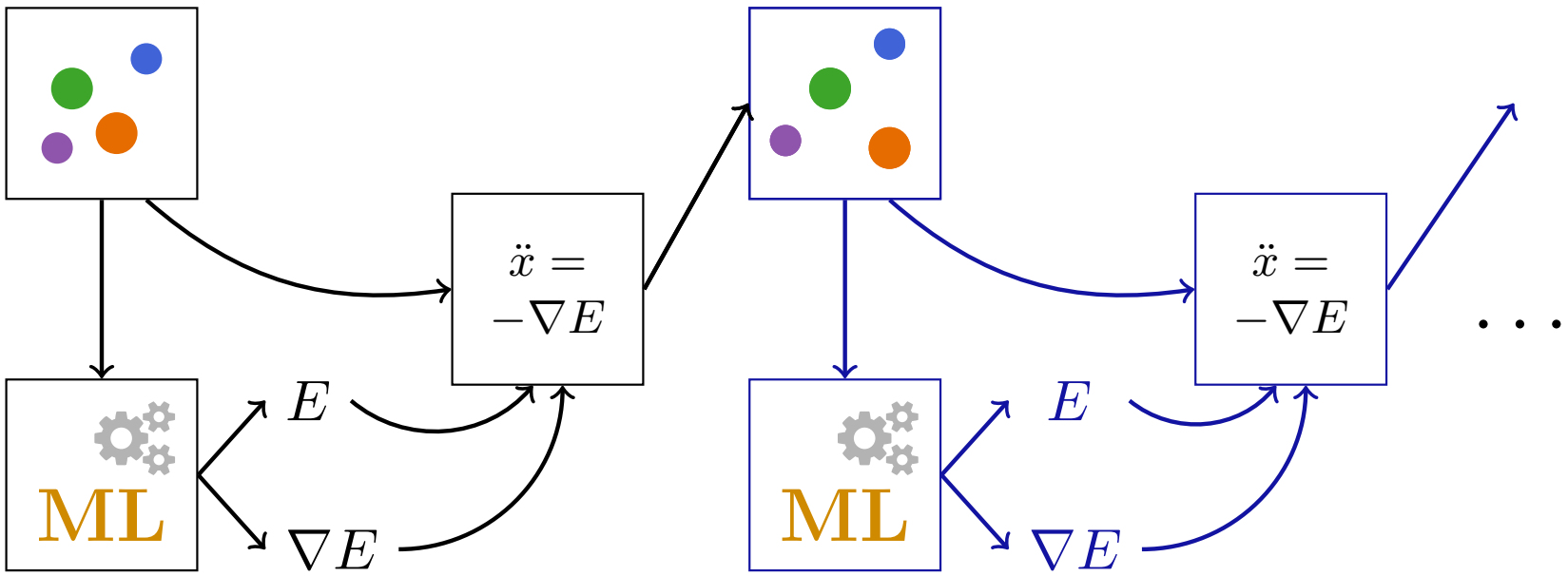
- Our video: 1000 time steps

Aluminium dynamics: Many time steps



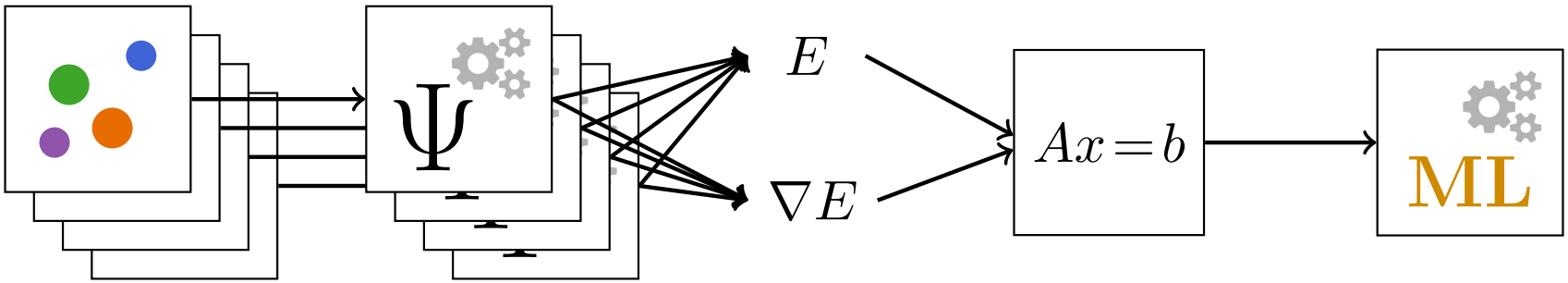
- Our video: 1000 time steps
- Runtime for one  : hours

Aluminium dynamics: Many time steps



- Our video: 1000 time steps
- Runtime for one Ψ: hours
- Runtime for one ML: $<$ 1 second

How to obtain ML



- Large-scale data generation & regression/training
- Many balances to strike:
  - Expressiveness of model
  - Accuracy of model
  - Data requirements for constructing sensible model
  - Cost of data generation
  - Cost of training procedure
  - Cost of evaluation of ML model
  - ...

$\Rightarrow$ No one size fits all

$\Rightarrow$ Flexibility in simulation (e.g. ML model) is key

Flexibility in practice . . .

- Text-based interfaces are still standard (to lesser and lesser extent)
- Vastly different syntax & features
  - Units, conventions, implicit assumptions, . . .
- Sometimes surprising changes between versions . . .

```
_symmetry_space_group_name_H-M 'P 1'
_space_group_IT_number           1
_cell_length_a 4.05
_cell_length_b 4.05
_cell_length_c 4.05
_cell_angle_alpha 90.0
_cell_angle_beta  90.0
_cell_angle_gamma 90.0
loop_
  _symmetry_equiv_pos_as_xyz
  '+x,+y,+z'

loop_
_atom_site_label
_atom_site_type_symbol
_atom_site_occupancy
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_Cartn_x
_atom_site_Cartn_y
_atom_site_Cartn_z
Al Al 1.0  0.00  0.00  0.00  0.000  0.000  0.000
Al Al 1.0  0.00  0.50  0.50  0.000  2.025  2.025
Al Al 1.0  0.50  0.00  0.50  2.025  0.000  2.025
```

CIF structure file

```
/
&FCP
/
&RISM
/
ATOMIC_SPECIES
Al 26.9815385 pd-Al.upf

K_POINTS automatic
12 12 12  0 0 0

CELL_PARAMETERS angstrom
4.05 0.00 0.00
0.00 4.05 0.00
0.00 0.00 4.05

ATOMIC_POSITIONS angstrom
Al 0.000 0.000 0.000
Al 0.000 2.025 2.025
Al 2.025 0.000 2.025
Al 2.025 2.025 0.000
~
~
~
```

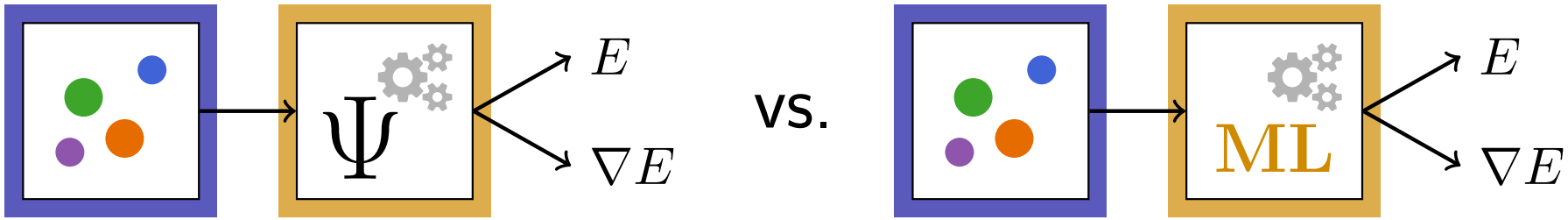
QuantumESPRESSO

```
4 atoms
1 atom types

0.0  4.05  xlo xhi
0.0  4.05  ylo yhi
0.0  4.05  zlo zhi

Atoms # atomic

     1   1      0      0      0
     2   1      0      2.025  2.025
     3   1      2.025  0      2.025
     4   1      2.025  2.025  0
~
~
~
~
~
~
~
~
~
~
~
~
```

LAMMPS

Some standard input files for our aluminium case

Solution: Invoke the power of abstraction



- **JuliaMolSim**: AbstractSystem (AtomsBase) and AtomsCalculator
- We are not the first to suggest such interfaces, examples:
  - Atomistic Simulation Environment (ASE)
  - Workflow managers like

    FireWorks, AFLOW, AiiDA, NOMAD, pyiron
- Important foundational work (and inspiration for us), but:
  - Often wrappers around text IO
  - New calculator is non-trivial effort
  - Two (or more) language problem

⇒ Not in line with opportunities & strengths of julia
  - Hard to exploit unique features in existing julia materials tools
  - Let's see some flagship examples . . .

Our aluminium simulation: Driven by julia tools[10]

DFTK

(DFT data generation)

```julia
using AtomsBase
using AtomsIO
using DFTK
using LazyArtifacts
using Unitful
systems = load_trajectory("Al_structures.extxyz")
all_data = map(systems) do system
    system   = attach_psp(system; Al=artifact"pd_nc_sr_pbe_stringent_0.4.1_upf/Al.upf")
    model    = model_PBE(system; smearing=Smearing.Gaussian(), temperature=1e-3)
    basis    = PlaneWaveBasis(model; Ecut=26, kgrid=(12, 12, 12))
    scfres   = self_consistent_field(basis; tol=1e-10)
    forces   = compute_forces_cart(scfres)
    stresses = compute_stresses_cart(scfres)

    properties = (; ene
                    vir
    atoms = map(system,
        Atom(; pairs(at
    end
    FlexibleSystem(syst
end
save_trajectory("Al_tra
```

acesuit

(Fit ACE ML model)

```julia
using ACEpotentials
using ExtXYZ
all_data = ExtXYZ.Atoms.(ExtXYZ.read_frames("Al_training_data.extxyz"))
model = acemodel(
    elements=[:Al, ],
    order=3,
    totaldegree=[20, 16, 12],
    Eref = [:Al => -2.180568046, ],
)
acefit!(model, all_data;
        solver=ACEfit.BLR(), prior=smoothness_prior(model; p=4))
save_potential("Al_potential.json", model)
~
~
~
~
~
~
~
~
~
~
```

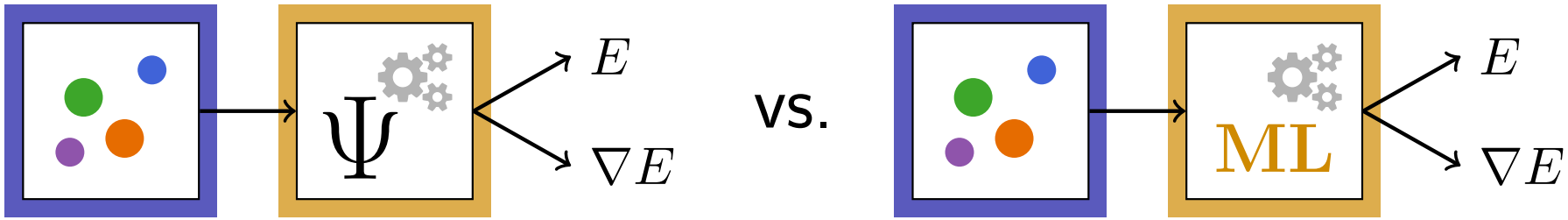
(Run molecular dynamics)

```julia
using ACEpotentials
using AtomsBase
using AtomsBuilder
using Molly
using AtomsIO
using GLMakie

potential = load_potential("Al_potential.json"; new_format=true)
structure = load_system("Al_test.extxyz", 2)
sys = Molly.System(structure, potential)

temp = 298.0u"K"
random_velocities!(sys, temp)
sys = Molly.System(sys; loggers=(; coords=CoordinateLogger(1)))
simulator = VelocityVerlet(;
    dt=0.5u"fs", coupling=AndersenThermostat(temp, 0.5u"ps"), )
simulate!(sys, simulator, 1000)  # Simulate 1000 time steps
visualize(sys.loggers.coords, sys.boundary, "simulation.mp4";
          color=:grey, markersize=0.2, show_boundary=false)
~
~
~
```

- Structures passed as AbstractSystem
- AtomsCalculator interface currently being finalised

[10] https://github.com/mfherbst/juliamolsim-demo

Solution: Invoke the power of abstraction

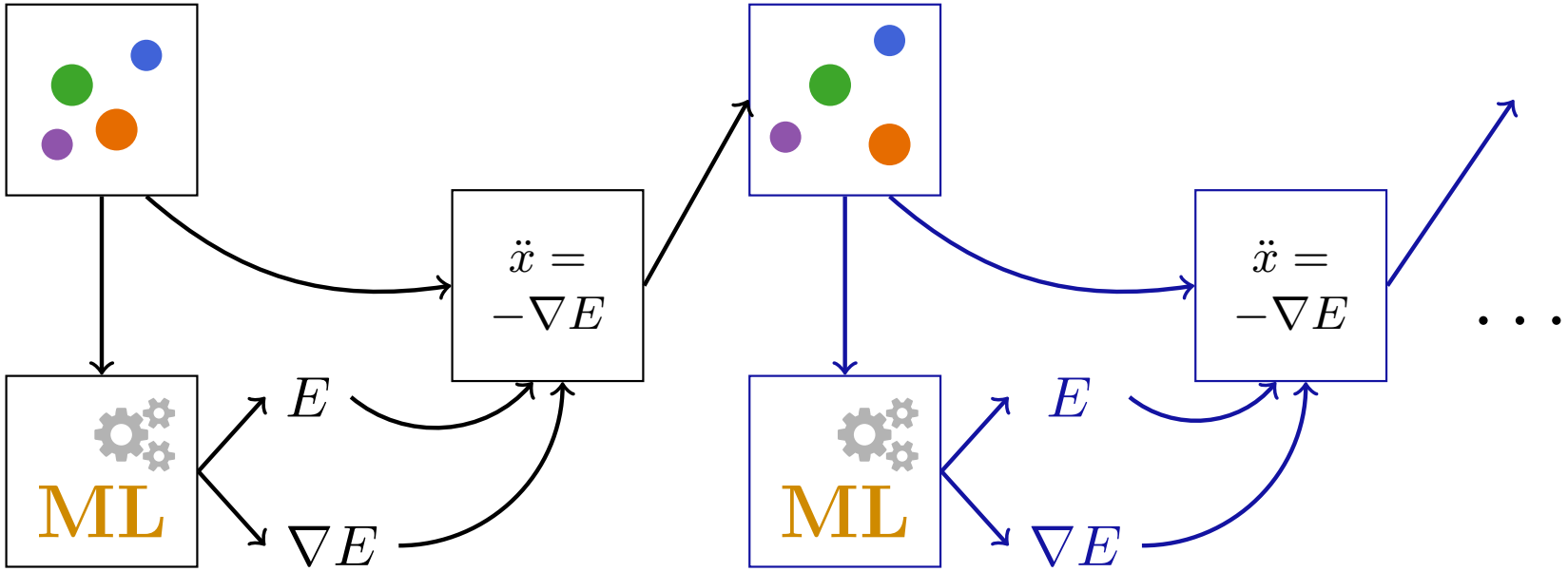


- **JuliaMolSim**: AbstractSystem (AtomsBase) and AtomsCalculator
- We are not the first to suggest such interfaces, examples:
  - Atomistic Simulation Environment (ASE)
  - Workflow managers like

    FireWorks, AFLOW (Automatic - FLOW for Materials Discovery), AiiDA, NOMAD (NOVEL MATERIALS DISCOVERY), pyron
- Important foundational work (and inspiration for us), but:
  - Often wrappers around text IO
  - New calculator is non-trivial effort
  - Two (or more) language problem

⇒ Not in line with opportunities & strengths of julia
  - Hard to exploit unique features in existing julia materials tools
  - Let's see some flagship examples . . .

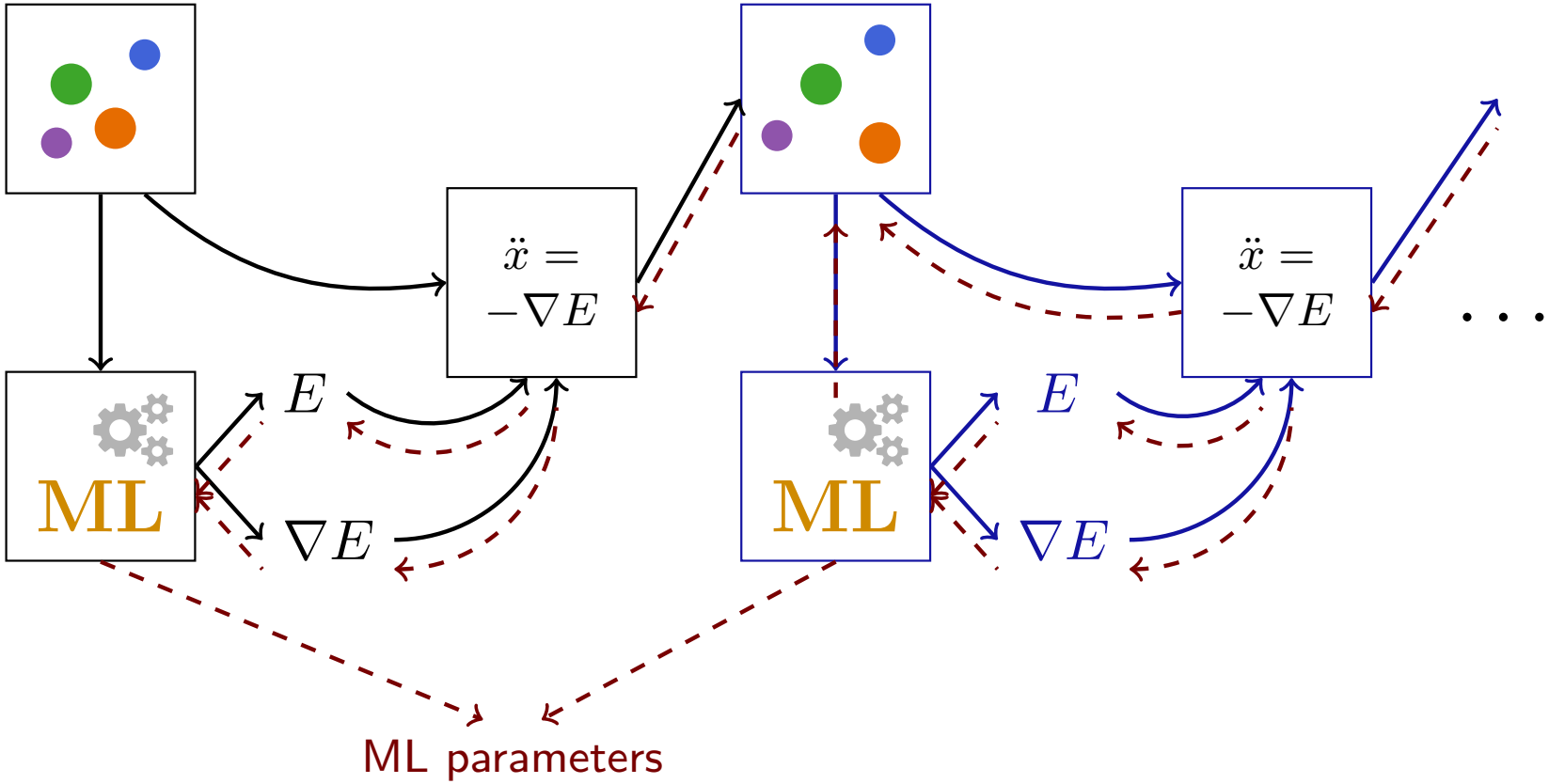
Inverse design of ML



- **Goal:** Fit parameters of ML
- Traditionally: Parameters fitted by "proxy quantities"
- Here end-to-end: Use AD *through the simulation*

$\Rightarrow$ Need parameters exposed in interface !

Inverse design of ML



- **Goal:** Fit parameters of ML
- Traditionally: Parameters fitted by "proxy quantities"
- Here end-to-end: Use AD *through the simulation*

⇒ Need parameters exposed in interface !

Inverse design of ML

Chemical Science

ROYAL SOCIETY OF CHEMISTRY

EDGE ARTICLE

View Article Online
View Journal | View Issue

Check for updates

Cite this: *Chem. Sci.*, 2024, **15**, 4897

All publication charges for this article have been paid for by the Royal Society of Chemistry

Differentiable simulation to develop molecular dynamics force fields for disordered proteins†

Joe G. Greener

Implicit solvent force fields are computationally efficient but can be unsuitable for running molecular dynamics on disordered proteins. Here I improve the a99SB-*disp* force field and the GBNeck2 implicit solvent model to better describe disordered proteins. Differentiable molecular simulations with 5 ns trajectories are used to jointly optimise 108 parameters to better match explicit solvent trajectories. Simulations with the improved force field better reproduce the radius of gyration and secondary structure content seen in experiments, whilst showing slightly degraded performance on folded proteins and protein complexes. The force field, called GB99dms, reproduces the results of a small molecule binding study and improves agreement with experiment for the aggregation of amyloid peptides. GB99dms, which can be used in OpenMM, is available at https://github.com/greener-group/GB99dms. This work is the first to show that gradients can be obtained directly from nanosecond-length differentiable simulations of biomolecules and highlights the effectiveness of this approach to training whole force fields to match desired properties.

Received 3rd October 2023
Accepted 8th February 2024

DOI: 10.1039/d3sc05230c

rsc.li/chemical-science

- **Goal:** Fit parameters of ML
- Traditionally: Parameters fitted by "proxy quantities"
- Here end-to-end: Use AD *through the simulation*

⇒ Need parameters exposed in interface !

npj | computational materials

www.nature.com/npjcompumats

ARTICLE OPEN

Check for updates

Hyperactive learning for data-driven interatomic potentials

Cas van der Oord[1]✉, Matthias Sachs[2], Dávid Péter Kovács[1], Christoph Ortner[3] and Gábor Csányi[1]

Data-driven interatomic potentials have emerged as a powerful tool for approximating ab initio potential energy surfaces. The most time-consuming step in creating these interatomic potentials is typically the generation of a suitable training database. To aid this process hyperactive learning (HAL), an accelerated active learning scheme, is presented as a method for rapid automated training database assembly. HAL adds a biasing term to a physically motivated sampler (e.g. molecular dynamics) driving atomic structures towards uncertainty in turn generating unseen or valuable training configurations. The proposed HAL framework is used to develop atomic cluster expansion (ACE) interatomic potentials for the AlSi10 alloy and polyethylene glycol (PEG) polymer starting from roughly a dozen initial configurations. The HAL generated ACE potentials are shown to be able to determine macroscopic properties, such as melting temperature and density, with close to experimental accuracy.

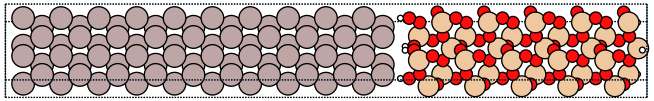
npj Computational Materials (2023)9:168 ; https://doi.org/10.1038/s41524-023-01104-6

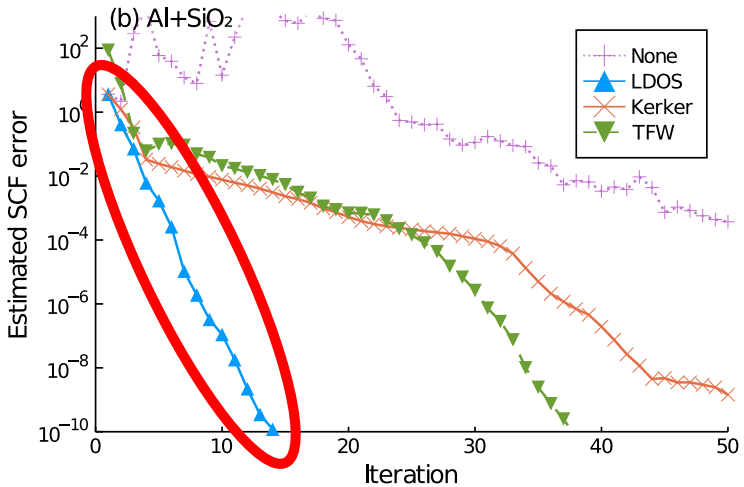
- Active learning: Train ML while running dynamics
  - ⇒ If prediction not confident for : Run extra Ψ

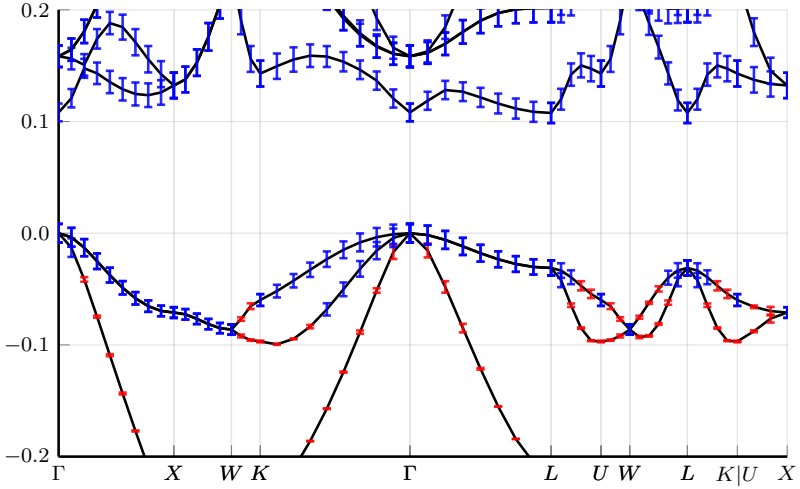
⇒ Switch back and forth between Ψ ↔ ML ↔ $\ddot{x} = -\nabla E$

- Breaks traditional workflows !

Robust algorithms and error control for







Silicon band structure with error bars

- First-principle models challenging & lack of mathematical analysis
- Numerics features considerable trial and error with parameters
- Recent work: Parameter-free and self-adapting algorithms[1,2]

- Numerical error estimation or model uncertainties hardly established
- First works[3,4]: Error in bands, densities and forces
- Need close maths / physics collaboration
- DFTK is major research tool here

[1] MFH, A. Levitt. J. Phys. Condens. Matter **33**, 085503 (2021).
[2] MFH, A. Levitt. J. Comput. Phys. **459**, 111127 (2022).
[3] MFH, A. Levitt, E. Cancès. Faraday Discus. **223**, 227 (2020).
[4] E. Cancès, G. Dusson, G. Kemlin *et. al.* SIAM J. Sci. Comp., **44**, B1312 (2022).

Efficient AD for

Letters in Mathematical Physics (2023) 113:21
https://doi.org/10.1007/s11005-023-01645-3

Numerical stability and efficiency of response property calculations in density functional theory

Eric Cancès[1,2] · Michael F. Herbst[3] · Gaspard Kemlin[1,2] · Antoine Levitt[1,2] · Benjamin Stamm[4]

```
function dft_forces(θ)
    system = ...
    model  = model_DFT(system, PbeExchange(θ))
    basis  = PlaneWaveBasis(model; Ecut=..., kgrid=... )
    scfres = self_consistent_field(basis).energies.total
    compute_forces_cart(scfres)
end
sensitivities = ForwardDiff.gradient(dft_forces, θ)
```

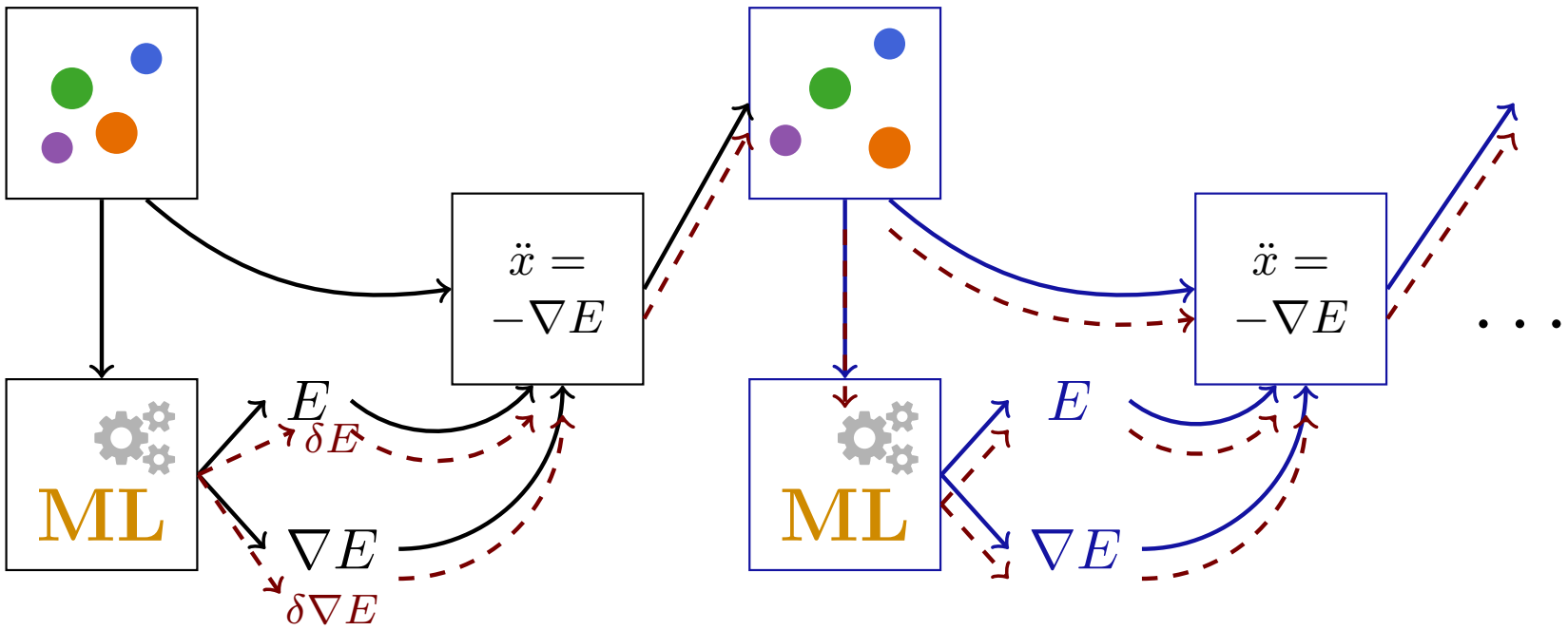
$$\rho_*(\theta) = \text{argmin}_\rho E(\rho, \theta)$$

$$\text{sensitivities} = \frac{\mathrm{d}\,\nabla E(\rho_*(\theta), \theta)}{\mathrm{d}\theta}$$

- Generic and efficient solvers for AD gradients (linear response)
  - $\Rightarrow$ A first for plane-wave DFT !
- Towards routine sensitivity analysis, inverse problems
  - But needs care: Talk by Niklas Schmitz (Th 15:40, Function (4.1))

$\Rightarrow$ To fully exploit: Interface supporting error propagation

Efficient AD for Ψ



- Generic and efficient solvers for AD gradients (linear response)
  - $\Rightarrow$ A first for plane-wave DFT !
- Towards routine sensitivity analysis, inverse problems
  - But needs care: Talk by Niklas Schmitz (Th 15:40, Function (4.1))

$\Rightarrow$ To fully exploit: Interface supporting error propagation

What we want: A shopping list for modern research thrusts

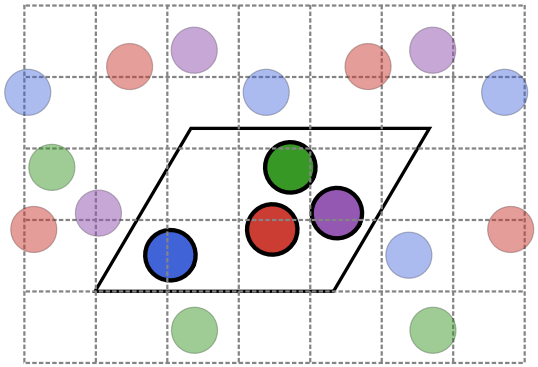
- Improve composability of flagship julia materials tools
  - Error control and robust algorithms
  - New ML models and learning approaches
  - Algorithmic differentiation
- Integration across ecosystems:
  - Vast portfolio of relevant methods (e.g. in LAMMPS, ASE, QE)
  - ⇒ Need two-way communication (for testing & adoption!)
- Dynamic and heterogeneous modelling pipelines:
  - Interdependent steps (active learning, on-the-fly data generation)
  - Heterogenous hardware (ML on GPU, DFT on CPU)
  - ⇒ Dynamic job scheduling is crucial
- Support cross-disciplinary research:
  - Speed of research problem
  - Not all fundamental maths / CS ideas work in practical setting
  - ⇒ Need flexibility inside hot loops (ML kernels, algorithms, . . .) & HPC performance (for testing & adoption!)
  - ⇒ People compose if software composes

Contents

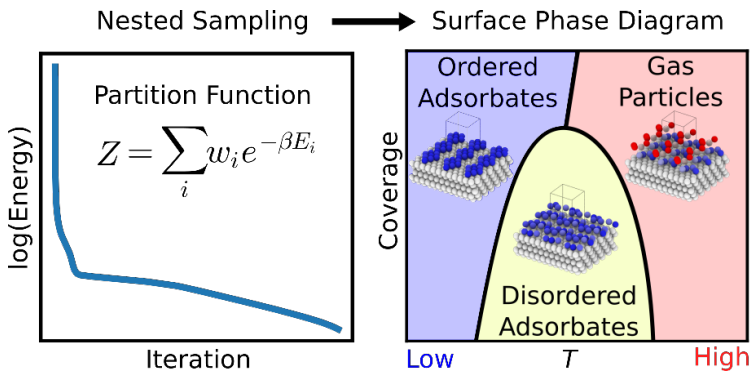
julia materials ecosystem: What you can do today

- Interact with AbstractSystems:
  - AtomsBuilder.jl: Build structures (surfaces, defects, . . .)
  - AtomsIO.jl: Read/write structure files
- Some calculators (not all AtomsCalculators . . . yet):
  - DFTK.jl: Density-functional theory (1D & 1000 electrons)
    (Talk on Friday, 12:00, Function 4.1)
  - InteratomicPotentials.jl: Simple parametrised materials potentials
  - acesuit: ACE-based machine learning potentials
  - Various active learning tools (e.g. HAL, PotentialLearning.jl, Cairn.jl
    — see talk on Friday, 16:20, Method 1.5)
- Dynamics and post-processing:
  - Molly.jl: Differentiable, GPU-enabled molecular dynamics
    (Talk on Friday, 11:00, REPL 2)
  - Wannier.jl: Localisation of electronic structures
- External ecosystem integrations:
  - ASEconvert, LAMMPS.jl, Spglib.jl . . .

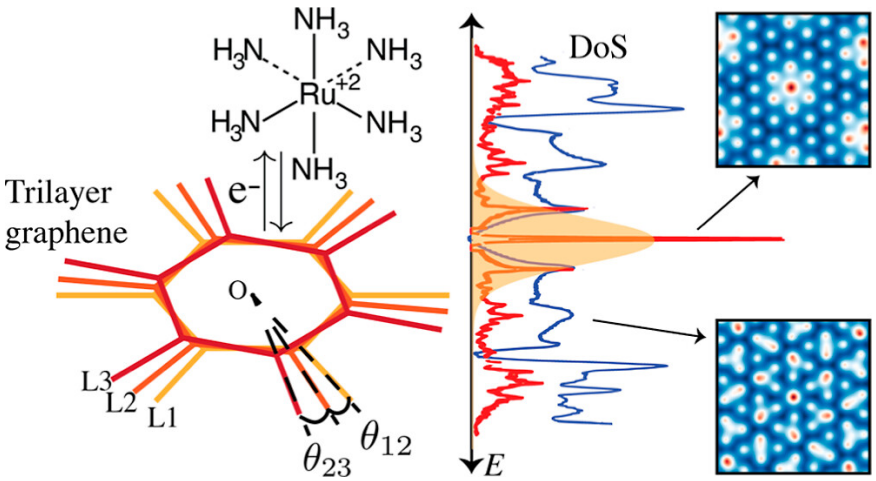
Other cool things we did not have time to talk about



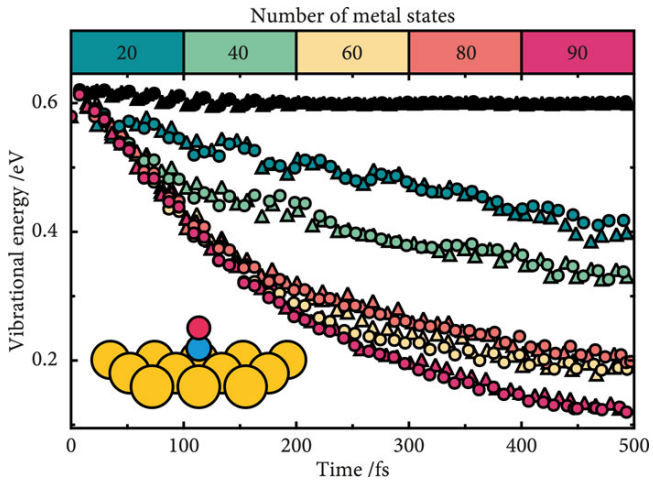
CellListMap.jl[1]: Neighbour lists



FreeBird.jl: nested sampling for phase diagrams



ElectrochemicalKinetics.jl[2,3,4]: Electrochem. reaction rates



NQCDynamics.jl[5,6]: quantum-classical dynamics

[1] L. Martínez Comp. Phys. Comm. **279**, 108452 (2022)

[2] RCK and V. Viswanathan J. Chem. Phys. **153**, 134706 (2020)

[3] RCK, D. Gandhi, and V. Viswanathan J. Phys. Chem. Lett. **14**, 35 (2023)

[4] M. Babar, Z. Zhu, RCK, *et. al.* J. Am. Chem. Soc. **146**, 23 (2024)

[5] J. Gardner, O. Douglas-Gallardo, W. Stark, *et. al.* J. Chem. Phys. **156**, 174801 (2022)

[6] J. Gardner, D. Corken, S. Janke, *et. al.* J. Chem. Phys. **158**, 064101 (2023)

Current state of JuliaMolSim

- Package development driven by research questions
  - Ecosystem is co-evolving with problems tackled
  - ⇒ Packages still lack breadth, but became more mature recently
  - julia tools have been key ingredient to push state of the art
  - ⇒ Community recognition (3 of JuliaMolSim now junior faculty)
- Interoperability and collaboration have been side projects
  - JuliaMolSim started around **2019** by Christoph Ortner, then AtomsBase (JuliaCon **2021**), AtomsCalculator (end **2023**)
  - **Past year:** Increased cross-group collaboration & community-centred packages (next slides)
  - ⇒ Integration is early days
  - ⇒ Potential is there, need funding to improve integration

JuliaMolSim: Some ongoing projects

- Increase internal interoperability and collaboration:
  - Redesign of AtomsBase & AtomsCalculator
  - GeometryOptimization.jl: Based on above, generic algorithms to find minimal-energy material structure
    (standard time-consuming modelling task)
  - NeighbourLists.jl: Fast, multi-hardware implementation
    (crucial ingredient to molecular dynamics)

- Cross-ecosystem integration
  - DFTK + AiiDA: julia backend within high-throughput DFT workflows
  - AtomsCalculator + ASE: Python-based interface standard
  - AtomsCalculator + i-PI: Socket protocol for dynamics

  ⇒ **Goal:** Improve testing and adoption !

JuliaMolSim: Gripes and issues

- Obstacles to julia adoption
  - Overly ambitious promises by the community
  - Too rudimentary binary support (dynamic libraries)
  - Gripes with python → julia calls
- Obstacles to JuliaMolSim adoption
  - Limited portfolio of available physical models
  - Insufficient performance (often low-hanging fruits!)
  - Lack of documentation
    - Integration between JuliaMolSim packages
    - Across-ecosystem integration
    - Examples
- Lack of important utility packages
  - Structure visualisation, plotting, printing & standard analysis

⇒ If you are interested to help, reach out !

What we want: A shopping list for modern research thrusts

(✓) • Improve composability of flagship julia materials tools
  - Error control and robust algorithms
  - New ML models and learning approaches
  - Algorithmic differentiation

(✓) • Integration across ecosystems:
  - Vast portfolio of relevant methods (e.g. in LAMMPS, ASE, QE)

  ⇒ Need two-way communication (for testing & adoption!)

× • Dynamic and heterogeneous modelling pipelines:
  - Interdependent steps (active learning, on-the-fly data generation)
  - Heterogenous hardware (ML on GPU, DFT on CPU)

  ⇒ Dynamic job scheduling is crucial

(✓) • Support cross-disciplinary research:
  - Speed of research problem
  - Not all fundamental maths / CS ideas work in practical setting

  ⇒ Need flexibility inside hot loops (ML kernels, algorithms, . . .) & HPC performance (for testing & adoption!)

  ⇒ People compose if software composes

How to get involved

Try things out! Integrate your stuff! File issues! Please don't hesitate to reach out and tell us what works and what doesn't.

Connect with us at **JuliaMolSim**.org and join our Zulip server to chat! We aim to have ∼monthly calls soon.

Here at JuliaCon:

- Find us after the poster session: socialise over drinks!
- Join us at the hackathon: we have a few project ideas

Advertisement break

Open PostDoc in the EPFL MtMat group

Topic: Efficient inverse materials design

- Interdisciplinary environment
- Bayesian optimisation
- AD & gradient approaches
- See `https://matmat.org/jobs/`

MolSSI workshop *(R. C. Kurchin)*:

"Julia for computational molecular and materials science"

- **20–23 October 2024** at Carnegie Mellon University, Pittsburgh
- Targets: People within and outside julia & JuliaMolSim communities

⇒ Stay tuned on JuliaMolSim Zulip for registration details !

Psi-k workshop *(M. F. Herbst, A. Levitt, J. Haegeman)*:

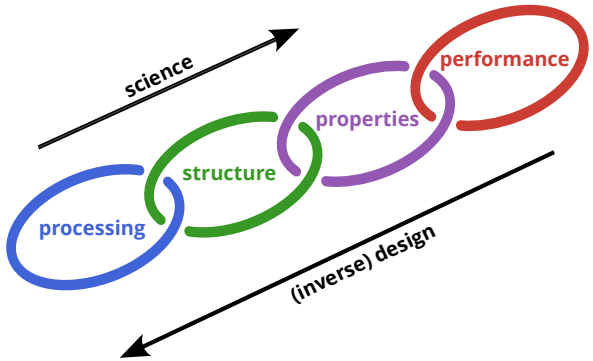
"Julia for numerical problems in quantum and solid-state physics"

- **26–28 November 2024** at EPFL, CECAM-HQ, Lausanne
- Targets: Linear algebra, physics and julia communities

⇒ `https://www.cecam.org/workshop-details/1355` (Deadline: 20th Sep)

Conclusion / Outlook

- Computational materials science
  - Established in academia & industry
  - High-throughput simulations:
    A different kind of HPC



- Importance of dynamic & heterogeneous workflows
  - Novel tools and paradigms (inverse design, active learning)
  - Composability is key ⇒ julia tools can be at the forefront

- **JuliaMolSim** ecosystem
  - Key simulation functionality available
  - Selected state-of-the-art features
  - Consensus building important, but (often) tiresome
  - Software integration (across ecosystems) is real work !

- Challenges looking forward
  - Expanding the user base beyond package developers
  - Reducing ecosystem (and julia) entrance barrier
  - More performance, documentation & core utilities

Acknowledgements

AtomsBase/AtomsCalculators

- Christoph Ortner (UBC)
- Teemu Järvinen (UBC)
- Joe Greener (Cambridge)
- Spencer Wyant (MIT)

JuliaMolSim community

- MxtMat group
- ACME group
- MIT CESMIX team

Questions?

Michael F. Herbst

mfherbst

michael.herbst@epfl.ch

matmat.org

Rachel Kurchin

rkurchin

rkurchin@cmu.edu

mse.engineering.cmu.edu/acmegroup

JuliaMolSim

https://juliamolsim.org/

https://michael-herbst.com/slides/juliacon2024