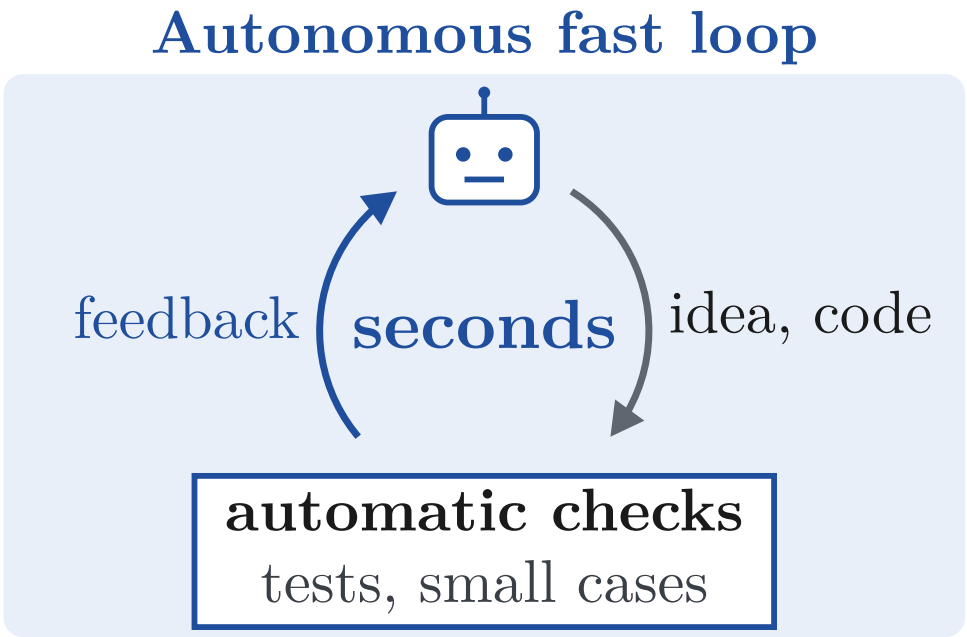
Rebuilding Computational Physics in Rust with AI Agents

Hiroshi Shinaoka
Saitama University

Slides (PDF)

Hiroshi Shinaoka (h.shinaoka@gmail.com)

Take-home message

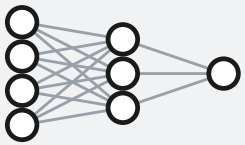


Make our research more autonomous:

***coding*, math, physics, and experiments**

Compressing many-body physics: a parallel route

Wavefunction, free energy



$\Psi(\sigma_1, ..., \sigma_N)$
$F = -T \ln Z$

ground states, finite T

neural networks

Green's function

$G(\boldsymbol{k}, i\omega_n)$  $\approx$

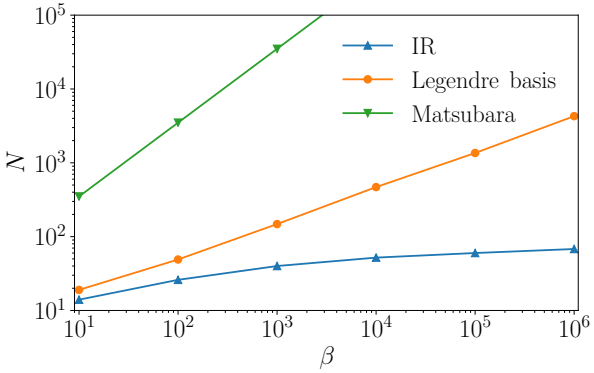
dynamics, response

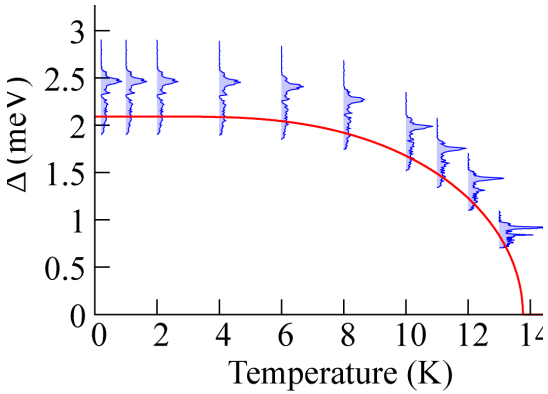
low-rank structure

Our route: compress **Green's functions** by low-rank structure

Carleo, Troyer, Science 355, 602 (2017). Nomura, ..., Imada, PRB 96, 205152 (2017). Xie, ..., L. Wang, PRL 131, 126501 (2023). IR: Shinaoka et al., PRB 96, 035147 (2017). QTT: Shinaoka et al., PRX 13, 021015 (2023). TCI: Ritter et al., PRL 132, 056501 (2024); Núñez Fernández et al., SciPost Phys. 18, 104 (2025).

IR basis: Matsubara Green's functions are low-rank





450 citations

sparse-ir Python

SparseIR.jl Julia

sparse-ir-fortran

used in **EPW**

Basis size: **IR** grows only as $\log \beta$

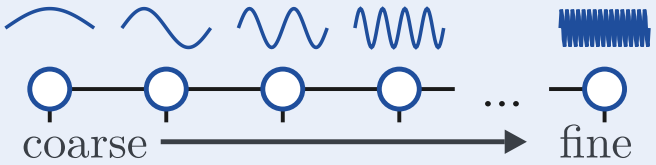
Nb, *ab initio* (EPW): gap down to $O(1)$ **K**

spm-lab.github.io

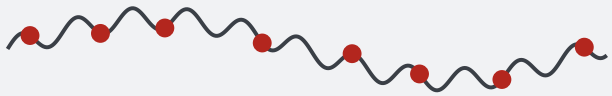
IR: intermediate representation. Shinaoka et al., PRB 96, 035147 (2017).
Sparse modeling for analytic continuation: Otsuki et al., PRE 95, 061302(R) (2017).
sparse-ir: SoftwareX 21, 101266 (2023). Nb: Mori et al., PRB 110, 064505 (2024).

Tensor trains: compression by scale separation

Quantics tensor train (QTT)
approximates a function $f(x)$



Tensor cross interpolation (TCI)
learns from **a few samples**



379 citations in

fluids | plasma | cold atoms | large solids | chemistry | finance | quantum computing | machine learning | HPC

Yuriel Núñez Fernández C++/Python — Univ. Grenoble Alpes, CNRS, Institut Néel

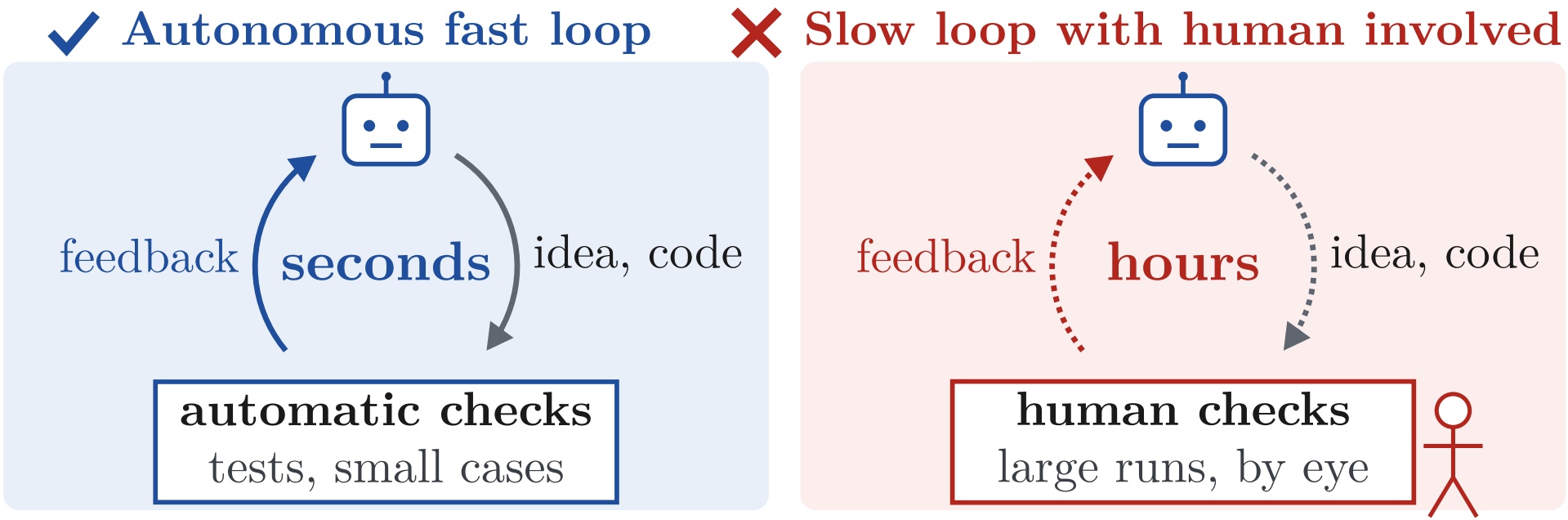
Marc K. Ritter Julia — CCQ, Flatiron Institute

So why rebuild it all in Rust with AI agents, now?

Shinaoka et al., PRX 13, 021015 (2023); Ritter et al., PRL 132, 056501 (2024);
Núñez Fernández et al., SciPost Phys. 18, 104 (2025). Google Scholar (2026-11).

AI speed needs a fast, autonomous loop

AI agent: writes, builds, tests and fixes code on its own



Goal: a platform to make full use of AI's speed

Two questions

Why Rust?

Why not C++ or Julia?

How to grow and maintain it with AI?

A huge code base: about 1 million lines



A fragmented ecosystem

Machine learning	QMC
Python	**C++**
PyTorch, JAX	ALPS, TRIQS, QMCPACK

Tensor networks	Ab initio
Julia / Python	**Fortran / C++**
ITensors.jl, TeNPy, quimb	VASP, Quantum ESPRESSO



How to integrate compression into every field's code?

Makes the feedback loop fast

Pains of current tools

- × **Legacy Fortran:** no unit tests
- × **C++:** segfaults at run time
- × **Julia:** slow compile, type instability

- × **Julia:**
  hard to call from C/C++/Fortran

- × **Fortran / C / C++:**
  no unified build and package system

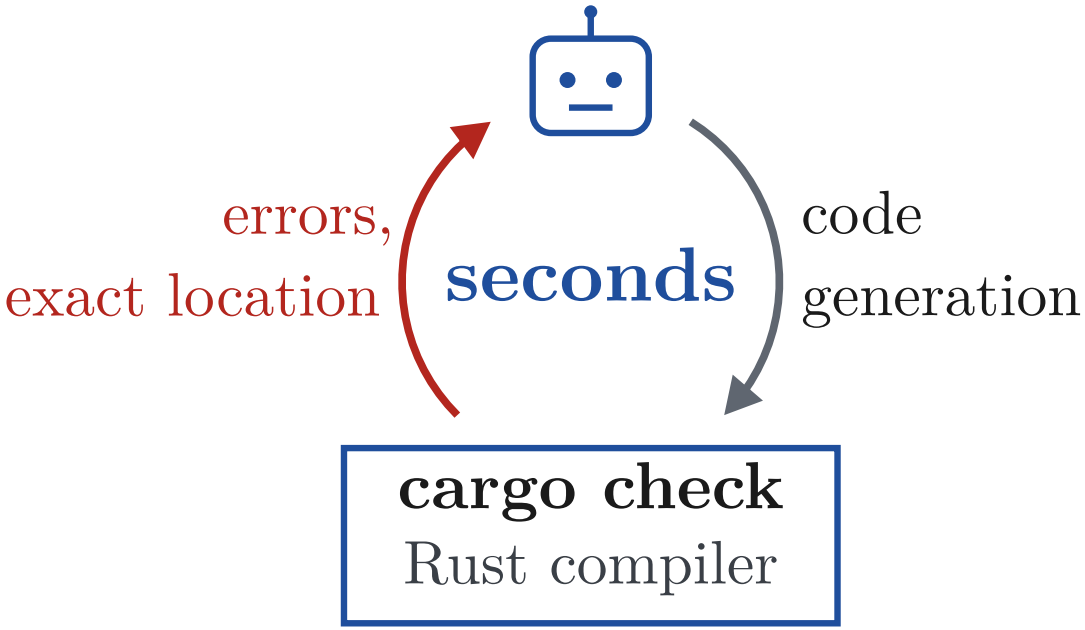
Rust

→ **1** **Fast error detection**

→ **2** **Reusable from every language**
via C API

→ **3** **Stable deployment**
fast, on any machine





Caught before running:

- ✗ type mismatch (f32 vs f64)
- ✗ data race between threads
- ✗ use of freed memory
- ✗ unhandled case in `match`

Errors caught in seconds, before running

But a steep learning curve

Julia

```
function normalize!(x)
  s = zero(eltype(x))
  for v in x
    s += v * v
  end
  n = sqrt(s)
  for i in eachindex(x)
    x[i] /= n
  end
end
```

Rust: hard for most physicists

```
fn normalize<T>(x: &mut [T])
where T: Float + DivAssign {
  let mut s = T::zero();
  for &v in x.iter() {
    s = s + v * v;
  }
  let n = s.sqrt();
  for v in x { *v /= n; }
}
```

$$x_i \leftarrow x_i / \sqrt{\sum_j |x_j|^2}$$

`&mut`, `Float + DivAssign`:
written for the compiler

If AI writes and reads it, **only Rust's strengths remain**

Industry has noticed too

- **Linux kernel**: Rust supported since 6.1 (2022)
- **Android**: >1000x lower memory-bug density
- **Microsoft**: Tier-1 language alongside C++ and C#
- **Discord**: desktop app Electron → Rust (planned)
- **Ruby on Rails creator**: **“Agents like Rust.”**



Google Security Blog (2025), Rust Foundation blog (2026), CTO post on X (2026-11).
Rails World keynote: youtu.be/vDjW_dRyKXY?t=1725

Rust in computational physics: still early

Machine learning	Candle (Hugging Face), Burn in Rust, with GPU support
Computational physics	Tensor networks, QMC, ab initio few Rust libraries yet

A blue ocean for us and younger generations



Candle: github.com/huggingface/candle. Burn: github.com/tracel-ai/burn.

Our software stack

tensor4all-rs **350k**
tensor networks: TCI, QTT

sparse-ir-rs **66k**
compressed Green's functions

tenferro-rs **360k lines**
tensors, linear algebra, autodiff



20 contributors, 7 countries, 1,362 PRs in 30 days
Human contributors in 2026, tensor4all repos (Rust, Julia, no forks), all using AI agents

tensor4all.org **spm-lab.github.io** **★ Please star us!**

tenferro-rs **360k lines since 2026-02-13**

tensors, einsum, FFT, linear algebra, autodiff



einsum cpueinsum-rs	**contraction** tcontract-rs	**decompositions** tlinalg-rs
strided arrays strided-rs	**dense kernels** t4a-faer (fork)	**threads** threadteam-rs

Built on top of our composable kernels

Reusable from other languages via C API

Benchmarks vs PyTorch: tensor4all.org/tenferro-benchmark

github.com/tensor4all/tenferro-rs

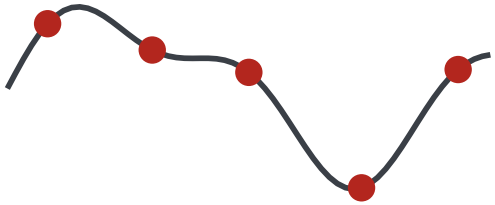
tensor4all-rs

Tensor trains



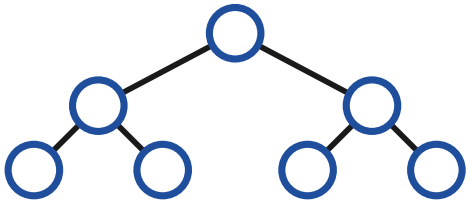
QTT: 2^R grid points

TCI



built from a few samples

Tree networks



any topology

Built on tenferro-rs; C API, ITensor-compatible Julia interface

Yuriel Núñez Fernández
original TCI in C++/Python

Marc K. Ritter
original TCI in Julia

TCI: tensor cross interpolation. QTT: quantics tensor train.
github.com/tensor4all/tensor4all-rs

What the optimized stack makes possible



K. Inayoshi

Hubbard quench with QTT

Memory: 0.29 GB $\rightarrow \sim$ 1e-3 GB

First result 2 months after starting Rust

Inayoshi et al., arXiv:2607.11055 (2026)

EPW

Electron-phonon code

Port to Rust (8,200 lines) in **4 hours with 1 agent**

Matches MgB_2 $T_c = 26.414$ K (unpublished)

mvmc-rs

mVMC (variational Monte Carlo) in Rust

T. Misawa, S. Terasaki

Already faster than the C and Julia versions

Two questions

✓ **Why Rust?**

Why not C++ or Julia?

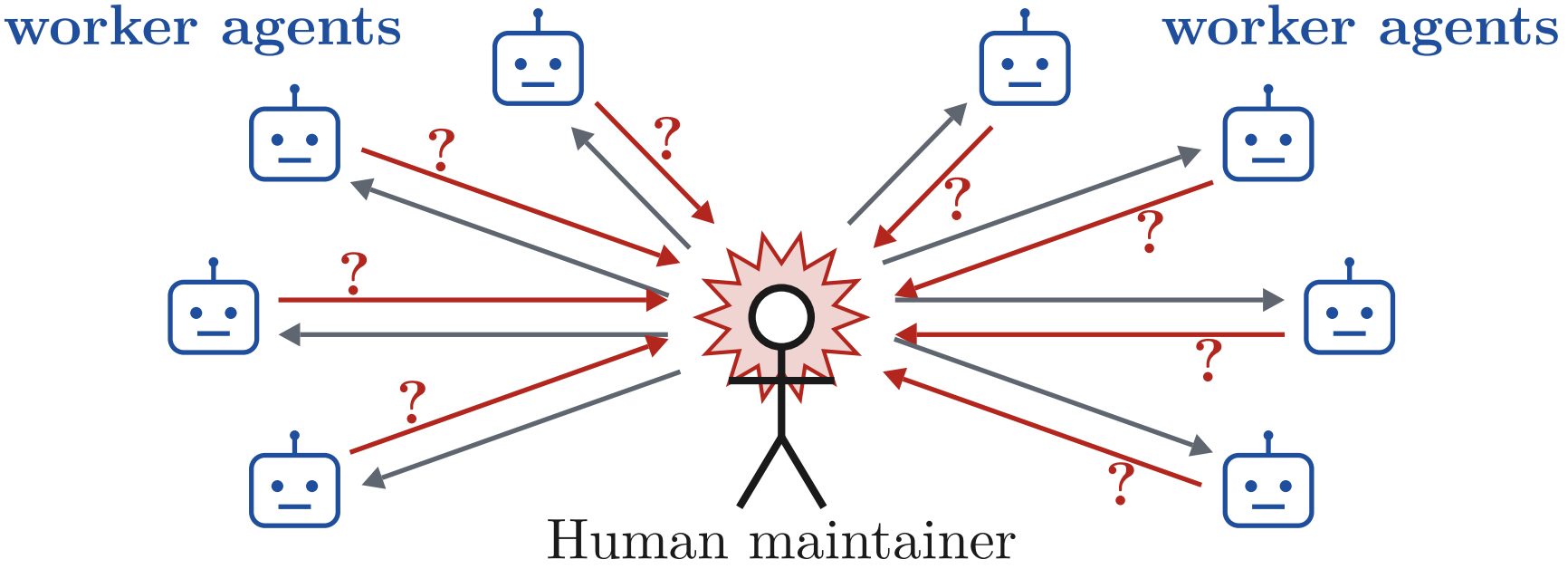
How to grow and maintain it with AI?

A huge code base: about 1 million lines



The human is the bottleneck

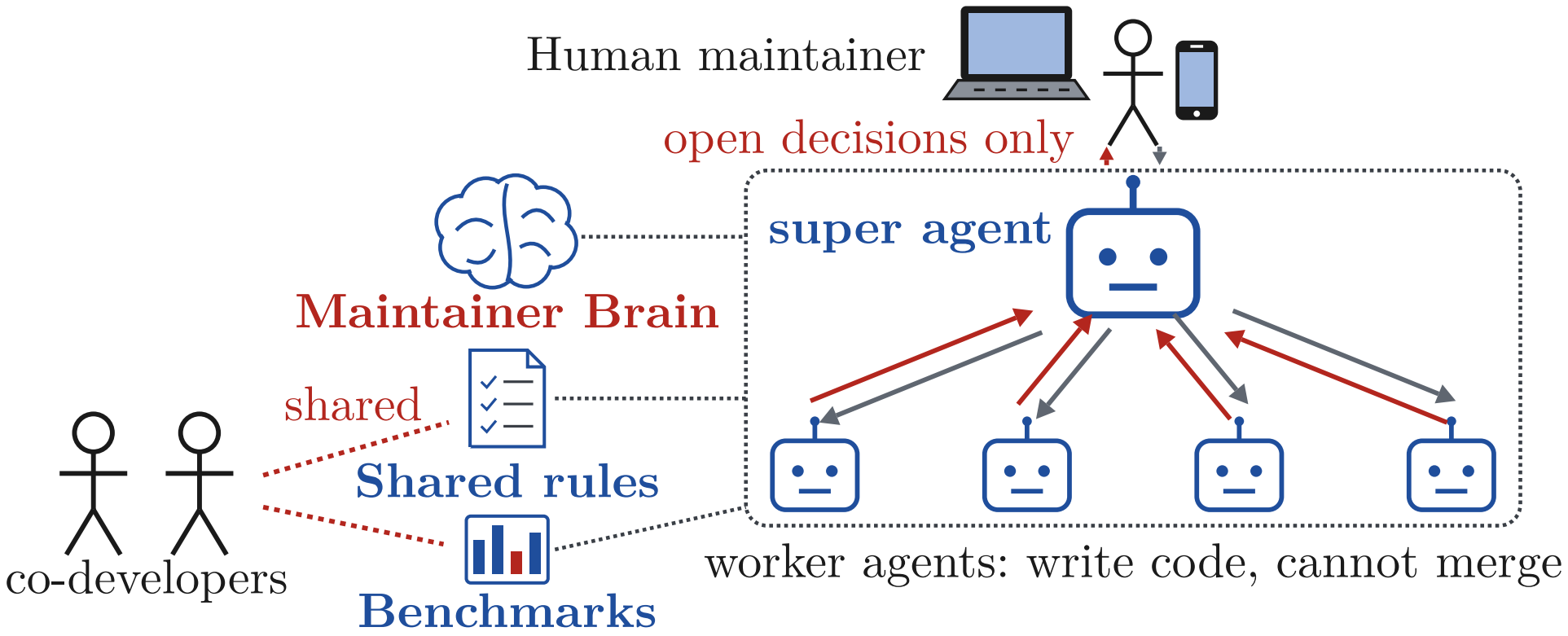
I maintain about 1 million lines of Rust as the lead maintainer.



Human attention is a scarce resource

spent on answering, deciding, and bookkeeping

A super agent handles a swarm of worker agents



tensor4all.org/blog/how-we-check-agent-written-code

DeepSeek's super-agent prototype (2026-10-31) follows the same principle

What the super agent relies on



Maintainer Brain

my judgment,
from past decisions

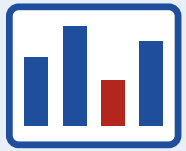
310 rules,
1,536 decisions



Shared rules

expert tricks and
project conventions

19 rule files,
4 skills



Benchmarks

public sites,
run regularly

tenferro, faer,
tlinalg, ...

maintainer only

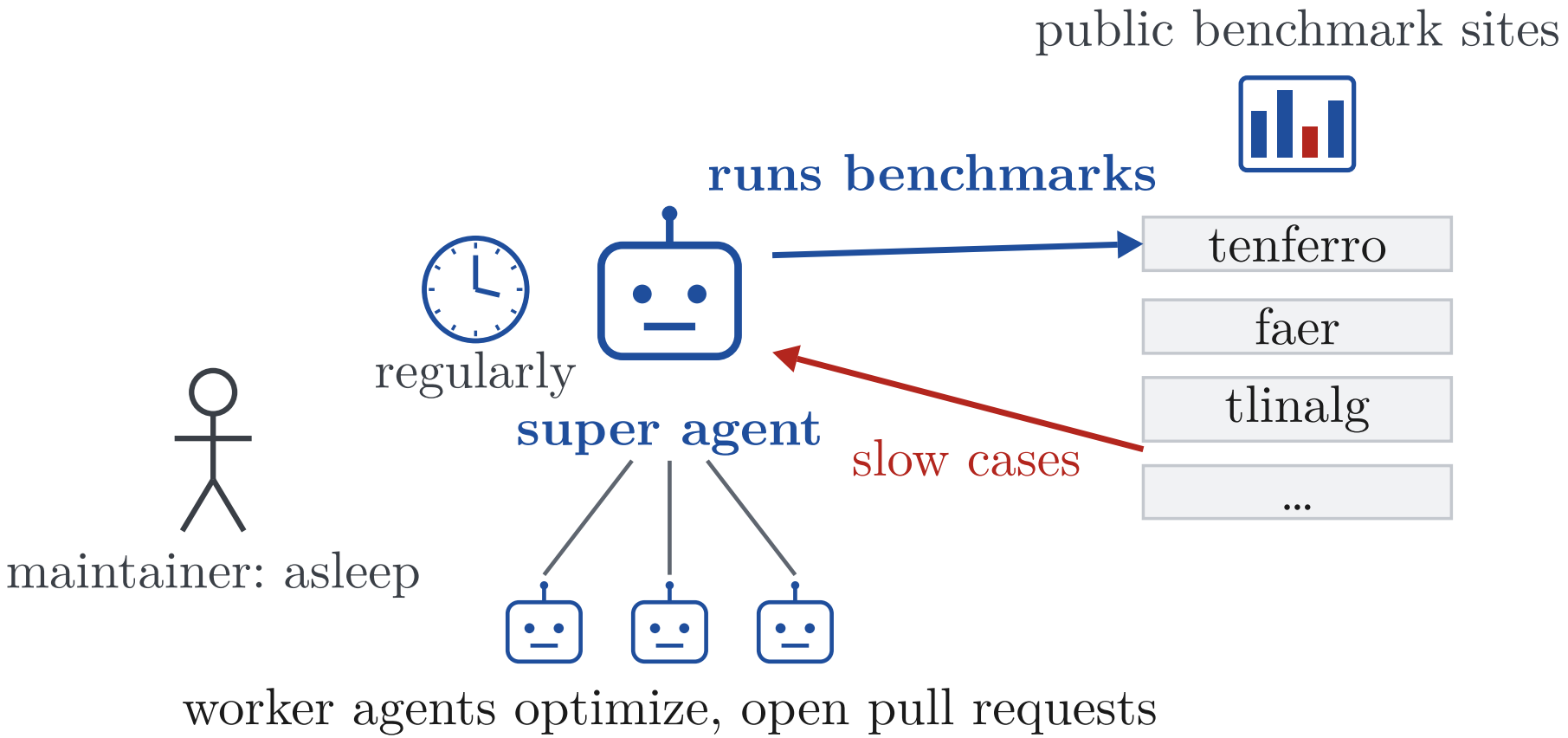
shared with co-developers

Growing: all three are updated adaptively after each mistake



Brain: decisions from 2026-05-20 to 10-28. Idea: github.com/QuantumBFS/sci-brain

Automatic optimization with public benchmarks

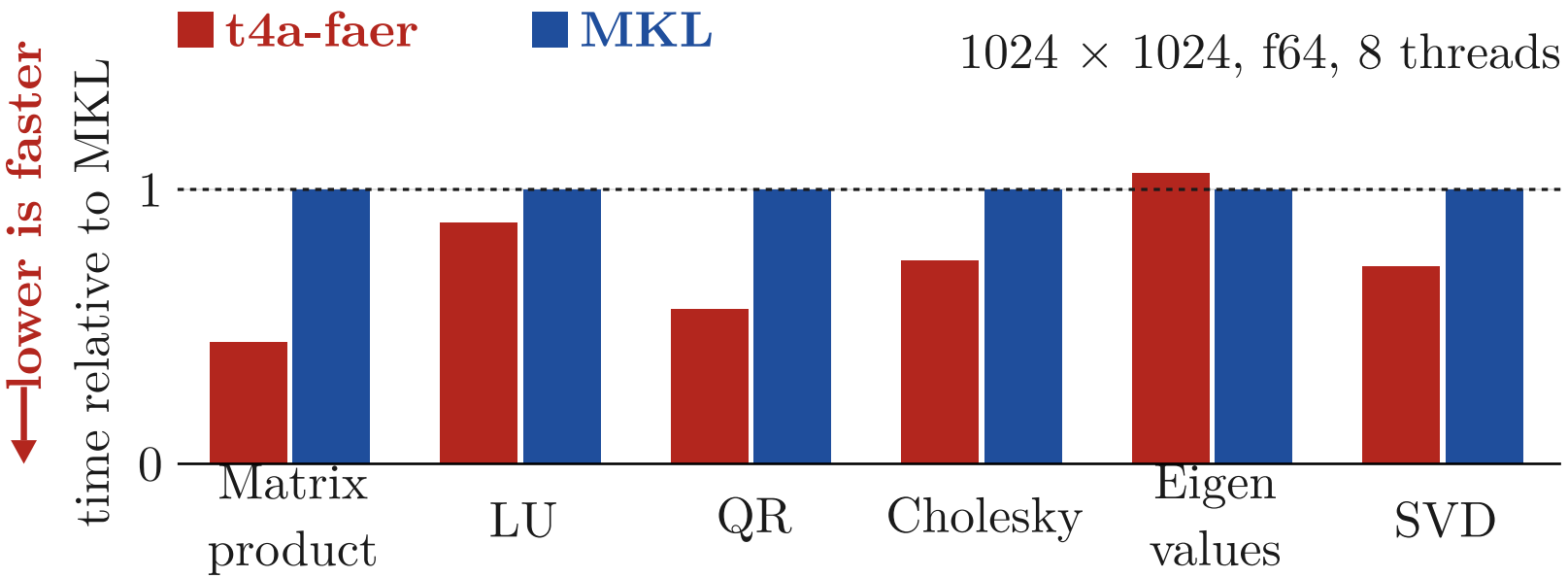


Sites: tensor4all.org/tenferro-benchmark, /faer-benchmark, /tlinalg-benchmark, ...

Case study: t4a-faer

faer: pure Rust linear algebra library

t4a-faer: our fork, optimized for multithreading, with bug fixes



During this talk, our agents are optimizing the library against AMD's optimized library tensor4all.org/faer-benchmark

Correctness: tests ported by AI find bugs



Example: SVD workspace fix for f32 (tensor4all/faer#73)

LAPACK test port and fixes: **M. Mikkelsen**. Further fixes: **S. Terasaki**.

Agentic AI development is financially sustainable

$10/month (DeepSeek) would be enough for a student

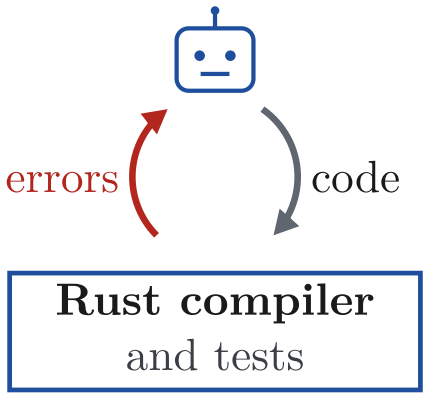
Collaborators		
Claude	Claude Team (premium seat)	**$20/month** (list $100)

Me (lead maintainer)		
Claude hard problems	Claude Team (premium seat)	$20/month
GPT6.1-sol reviewer	ChatGPT / Codex	$100/month
DeepSeek V4.1 Flash workers, super agent	Ollama Cloud OpenCode Go	$100/month $40/month

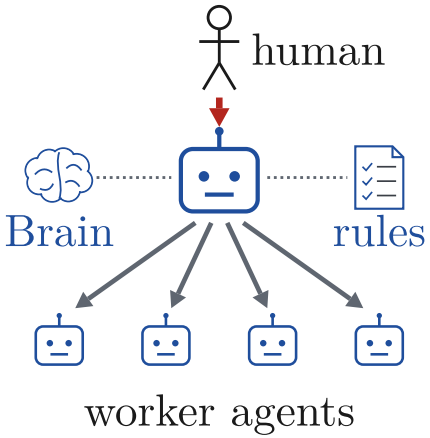


Summary

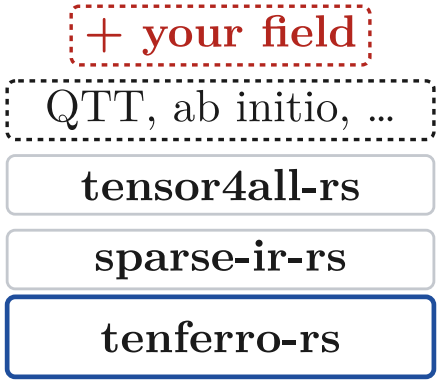
Why Rust?
Checks in seconds



How to maintain?
Super agent



What we built
Growing stack



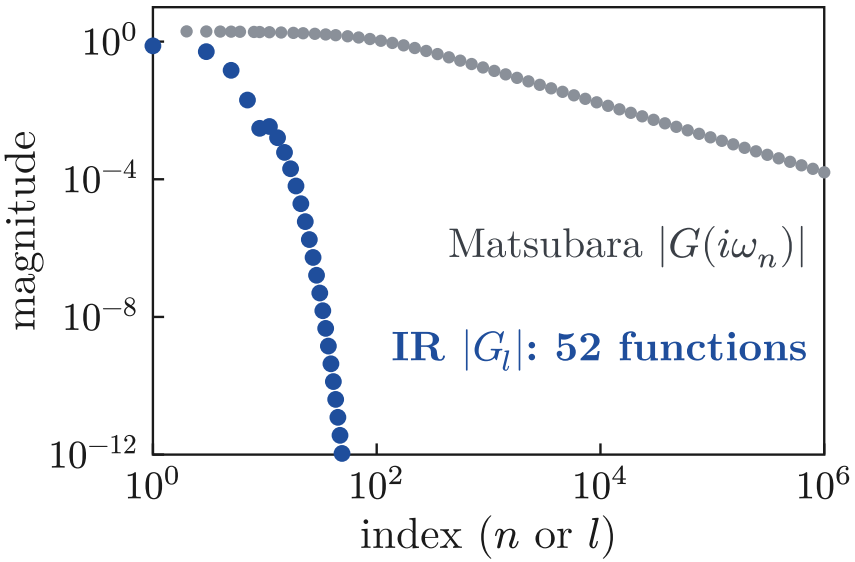
Spring School on Computational Physics 2027

Feb 28 to Mar 4, Okinawa **Hackathon session in English**

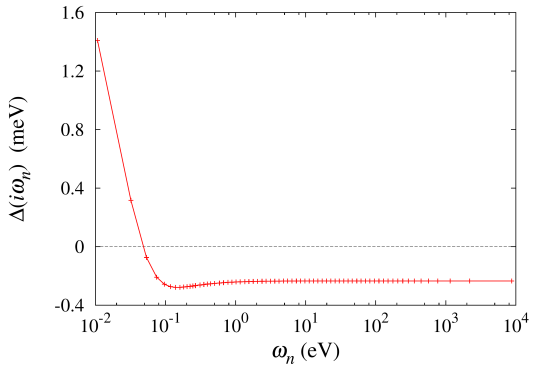
Supported by many grants, including ML physics

Join us! **tensor4all.org** **compphysschool.github.io/2027**

Backup: the IR basis



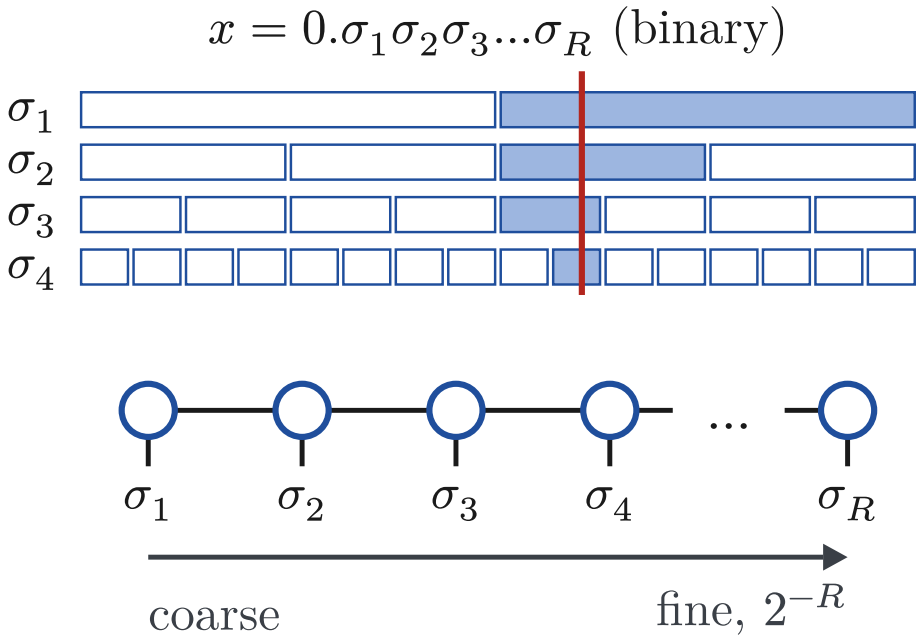
IR: **52 numbers** reach 10^{-12}

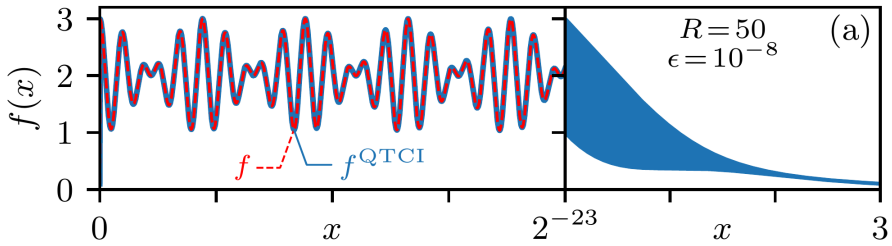


Eliashberg, Nb: **138 points**
instead of dense Matsubara grids

IR (intermediate representation): Shinaoka et al., PRB 96, 035147 (2017).
Sparse sampling: Li et al., PRB 101, 035144 (2020).
Right: Wang, ..., Arita, PRB 102, 134503 (2020).

Backup: QTT, one tensor per scale





2^{50} points, scales 2^{-23} to 3

bond dimension 15

Right: Ritter et al., PRL 132, 056501 (2024).

Backup: compile-time check of a data race

C++ / OpenMP

```
#pragma omp parallel for
for (int i = 0; i < n; ++i)
  total += x[i];
```

Compiles; wrong sum at run time

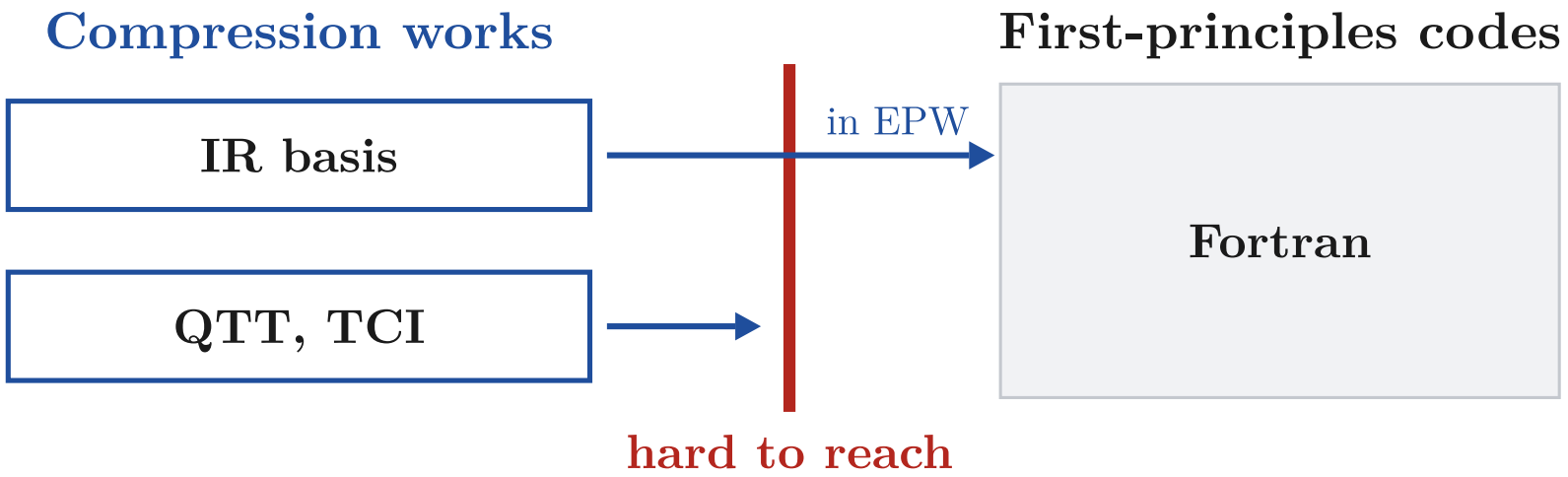
Rust / rayon

```
x.par_iter()
  .for_each(|xi| total += xi);
```

```
error[E0596]: cannot borrow
`total` as mutable
```



Backup: compression is hard to bring into ab initio codes



Our own libraries: four languages, each maintained separately

Python | Julia | C++ | Fortran

So we rebuilt one Rust base with AI agents

QTT: quantics tensor train. TCI: tensor cross interpolation.
EPW: electron-phonon code. IR in EPW: Mori et al., PRB 110, 064505 (2024).