Using Rust in RSE

Alex H. Room

2025/10/29

Introduction

- Experiences using Rust for pySpinW
- Binding Python to Rust
- Linear algebra
- Parallelism

How are we using it for pySpinW?

- Core spin-wave calculation
- Mostly small/medium matrix linear algebra
- In parallel over **Q** points

The headlines

- 50x-100x speedup over NumPy
- Writing Rust code is very quick (once you're over the learning curve)
- Paralellism is so, so easy
- Linear algebra is a underdeveloped but great tools exist
- Python bindings are a little harder...

Benchmarks

Language					
Python (`NumPy`)	2.37	2.02	4.76	4.11	312.74
Rust (`faer`)	0.07	0.08	0.17	0.20	3.59

The learning curve

- Rust is weird if you're used to Python/C/C++
- Some of that is new weirdness, some is just 40 years of programming language evolution
- You're forced to get good straight away, sloppy code isn't allowed
- Code either runs safely or doesn't compile!
- *Programming Rust* by Blandy, Orendorff and Tindall(!!)

Parallelism with rayon

```
fn my_function(inputs: Vec<f64>) -> Vec<f64> {
  inputs.into_iter().map(do_something).collect();
}

fn do_something(input: f64) -> f64 {
  // Some expensive calculation
}
```

Parallelism with `rayon`

```rust
use rayon::prelude::*;

fn my_function(inputs: Vec<f64>) -> Vec<f64> {
  inputs.into_par_iter().map(do_something).collect();
}

fn do_something(input: f64) -> f64 {
  // Some expensive calculation
}
```

Linear algebra in Rust

- Lots of options, none as mature as `NumPy/SciPy`
- `nalgebra` - 1D/2D arrays, decently fast, some issues...
- `ndarray` - similar to NumPy, general arrays, slow!
- `faer` - focused on medium/large matrix algebra, very fast, easiest to use

Python bindings with `pyo3`

- `pyo3` is the main way to bind Python to Rust
- Best used with `maturin` to build packages!
- Most basic Python types supported out of the box

Caveat emptor

- Memory management between Python and Rust is tricky
- `pyo3` forbids some Rust functionality to manage this
- Numpy arrays need special handling which can be unintuitive

Python bindings with pyo3

Let's get hands on!