Co-funded by the European Union

EuroHPC Joint Undertaking

MultiXscale

"The most rewarding aspect of this work is being at the intersection of research and software engineering"

01 Introducing myself

Jean-Noël Grad
Research software engineer
ICP - U. of Stuttgart (Germany)

I'm a research software engineer at the University of Stuttgart. My role is to assist domain scientists in turning their ideas into scientific software that runs reliably on any operating system and hardware, that other research institutions can readily adapt to tackle new scientific questions, and that is thoroughly tested and documented. **The role goes beyond writing code**: I also train new users through annual workshops, mentor new developers as they join the software community, and act as a bridge between the teams that build software libraries and the scientists who use them, translating concrete research needs into the more general, abstract requirements that software developers can act on.

02 My role in the MultiXscale project

Within the MultiXscale project, my main responsibility is maintaining and extending the **ESPResSo simulation software**, a tool that scientists use to study problems as varied as new energy-storage materials, responsive polymer gels, molecular recognition by nanosensors, and bacterial motility in porous media such as self-healing concrete. A key part of my work is to help ESPResSo take full advantage of the latest generation of European supercomputers. This matters because some physical phenomena only emerge at relatively large scales, on the order of micrometres, yet understanding them requires tracking the position and velocity of every single molecule in the system. The amount of information involved quickly exceeds what any single computer or graphics processor can hold in memory, so the calculation has to be distributed across many machines working in parallel. **Supercomputers make this possible**, running enormous simulations across several hundred graphics processing units, each holding up to a billion particles.

A particular challenge is that modern supercomputers are heterogeneous: they combine different kinds of processors, and the same calculation often demands a completely different algorithm or data layout depending on whether it runs on a **conventional processor (CPU) or a graphics processor (GPU)**. In the special case of energy-storage materials, it can be beneficial to partition the modelled system into multiple regions with different levels of resolution. For example, the same ionic liquid can be represented three ways at once: as a continuous fluid on a regular grid when far from the electrode, as a small number of large spheres when close to the electrode (a technique called coarse-graining), and as molecules with individually-resolved atoms inside the nanoporous structure of the electrode material. This approach lets us model the energy-storage physics in fine detail where it is truly necessary, while treating the bulk transport of the electrolyte at a lower level of detail, saving computing resources without compromising accuracy.

This work is inherently collaborative. It has drawn on **partnerships with the POP CoE** to profile ESPResSo's performance, with the waLBerla developers to extend the capabilities for grid-based simulations of electrolytes, and with the EESSI team to make ESPResSo readily available on European supercomputers through a shared, ready-to-use stack of scientific software.

Co-funded by the European Union
EuroHPC Joint Undertaking
MultiXscale

03 Impact of my work

A distinctive strength of ESPResSo is its versatility. Rather than a fixed application, it provides building blocks that users assemble to create their own simulation environments. They can combine particle-based and grid-based models, connect them to numerical solvers for electrostatics and magnetostatics, or change the properties of a system on the fly, either by hand or through automated rules that trigger when predefined conditions are met. The high-performance core is written for speed, but the user-facing interface uses the Python programming language, so researchers can readily combine ESPResSo with well-established Python libraries for machine learning, data analysis, and molecular modelling.

This design makes ESPResSo a platform for rapid prototyping: new simulation methods can be developed, reviewed and validated by peers, and later re-implemented in more specialised software. For this to work, the code has to be modular, extensible and easy to reuse, so that other developers can adapt our methods with minimal effort. To achieve portability across the constantly changing landscape of hardware, we rely on modern libraries that automatically generate optimised code for different CPU and GPU architectures. In practical terms, a developer can write down the mathematical equations to be solved and let the software translate them into code (as with LbmPy), or pair pre-defined algorithms with data structures and let the compiler generate the boilerplate code (as with Kokkos and Cabana). This shields scientists from low-level technical detail and keeps the software running efficiently on current and future hardware.

04 Looking ahead

Researchers increasingly rely on computational models to **accelerate the discovery of new materials** and to focus laboratory experiments on the most promising candidates. This makes information technology a core part of research infrastructure, and it creates a lasting need for dedicated professionals to maintain the computing resources and software libraries that research now depends on.

High-performance computing is also becoming more accessible. Local and regional initiatives are pooling computing resources into shared clusters that use the same interfaces and software stacks as the large European supercomputers. Once scientists are trained on these local systems, they can far more easily scale up to EuroHPC facilities and unlock the more ambitious models that can advance their research.

Artificial intelligence is opening further opportunities. ESPResSo is currently being extended to support machine-learned interatomic potentials (MLIPs), a way of reintroducing chemical detail in strategic regions of a simulation, such as the interface between electrolyte and electrode, where quantum-mechanical effects matter most. MLIPs occupy a middle ground: more accurate than classical models and, while more demanding to compute, still orders of magnitude cheaper than the full quantum-mechanical calculations they approximate. Large language models are likewise assisting software developers in writing better code and writing it faster, for example, by analysing a call stack to narrow down the likely cause of a bug, or by suggesting hardware-specific optimisations for the most performance-critical parts of the program.

The most rewarding aspect of this work is being at the intersection of research and software engineering: taking the specific needs of domain scientists and translating them into general and reusable features that can be developed once and then adapted with minimal effort to problems in entirely different scientific fields. It is a role that demands both **scientific understanding and software engineering skill**. If there is one message I would like readers to take away, it is that well-engineered, sustainable research software, and the people who build and maintain it, are an increasingly valuable part of Europe's research infrastructure.