VOLUME 4 I OCTOBER 2023

NOMAD Center of Excellence

NEWSLETTER

EDITORIAL

2023 has been a busy year so far with great strides for the NOMAD CoE. We have held numerous events, including workshops, summer schools, and hackathons, and had productive exchanges between the partners.

In this release of the NOMAD CoE Newsletter, you will find a range of topics: We cover tensor-reduced atomic descriptors, an AI method within the SISSO++ code for GPU-accelerated supercomputers, the importance of workflows for harnessing the power of exascale computers, and LUMI-G enabled by NOMAD workflow software. Finally, we provide you additional information about our developments and our upcoming workshops and hackathons organized for the community.

We wish you a happy and successful rest of 2023 and look forward to meeting you at one of our events!

The NOMAD CoE team.

TENSOR-REDUCED ATOMIC DESCRIPTORS

James Darby, James R. Kermode, and Gábor Csányi

Atomic descriptors (and representations of entire structures) have become a standard tool used in the visualization and analysis of material datasets and in the machine learning of interatomic potentials. The core idea underpinning density-based descriptors is to construct a fictious atomic density, take tensor products of this density and then symmetrize with respect to the rotation group. As a separate density is used to represent different chemical elements, the tensor product causes the size of the descriptor to scale as $O(NS^{\nu})$ where S is the number of chemical elements, N is the number of radial basis functions used in the density expansion and ν is the correlation order – equivalent to a body-order of ν+1. This scaling is particularly problematic for the chemically diverse NOMAD database, where the goal is to store descriptors alongside structural information and computed properties to help facilitate data analysis and visualization.

To tackle this scaling issue, tensor-reduced versions of both the SOAP (smooth overlap of atomic positions) and ACE (atomic cluster expansion) density-based descriptions were recently introduced in a paper by NOMAD team members[1]. These strategies achieve compression by exploiting the tensor-product structure either by approximating the descriptor itself using tensor-sketching, or by approximating the coefficients of a hypothetical linear model using symmetric rank-1 tensor decomposition, as illustrated in Figure 1, then propagating this approximation into the descriptor. Both strategies are systematic and reduce the formal scaling of the descriptor size to $O(K)$, where K is a convergence parameter that simultaneously controls chemical and spatial resolution: it can be thought of as the number of 'channels' to retain.

The weights (components of the rank-1 tensors) used

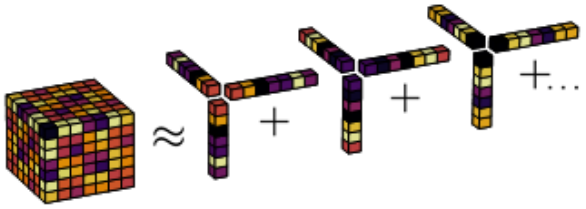

Figure 1: A schematic of the decomposition of a symmetric tensor, c, using rank-1 tensors is shown above the corresponding equation.

[1] J. P. Darby et al., Phys. Rev. Lett. **131**, 028001 (2023).

in the decomposition can be optimized during model fitting, as is done in the recently developed MACE code, but in some cases it is desirable to use random weights instead. For instance, using random weights allows tensor-reduced ACE models to remain strictly linear and retain the associated advantages such as a closed form solution and Bayesian uncertainty quantification. Using random weights is also advantageous in data visualization, where there is no obvious target quantity to optimize the weights with respect to and avoiding weight optimization keeps the visualization fast.

An example of such a visualization is shown in Figure 2. A SOAP descriptor was computed for each molecule and then Kernel-PCA was used to embed the descriptors into 2D. Repeating this process using tensor-reduced SOAP descriptors containing 10x fewer components resulted in a very similar map, as can be seen by comparing the left- and right-hand columns.

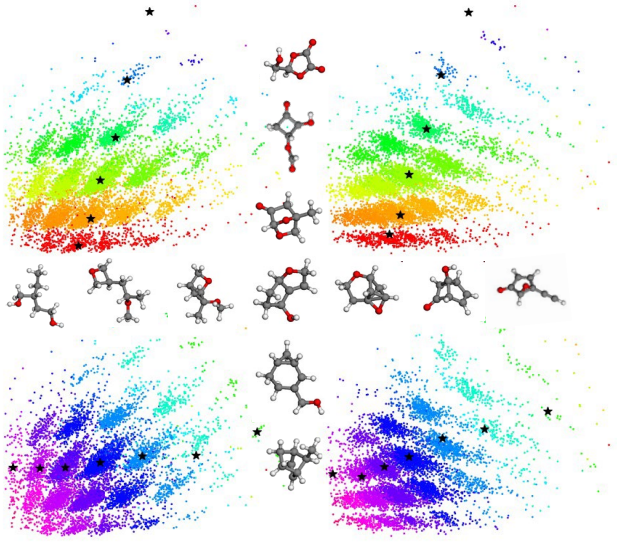

Figure 2: Four kernel-PCA maps of ~7500 molecules containing only C, H, O and F from the QM9 dataset are shown. The maps in the left column use regular SOAP descriptors while those in the right use tensor-reduced SOAP. Points in the top and bottom rows are colored according to the O and H content in the corresponding molecule, respectively. The molecules in the center correspond to the black stars on the maps.

Tensor-reduced SOAP descriptors will soon be pre-computed for all entries in the NOMAD database, providing a convenient starting point for anyone interested in analyzing the data. They are also available for general use via the quippy python package.

SISSO++: TOWARD EXASCALE EXPLAINABLE AI MODELS

Yi Yao and Tom Purcell

Big data-driven science is increasingly important in materials science[2]. Explainable AI models, such as symbolic regression, are valuable for discovering novel materials and new applications. Scientists from the NOMAD Laboratory at the Fritz Haber Institute of the Max Planck Society (FHI) and from the Max Planck Computing and Data Facility (MPCDF) have recently optimized an explainable AI method within the SISSO++ code for GPU-accelerated supercomputers. This was achieved through an implementation using the Kokkos performance-portable programming model, enabling the maintenance of a single codebase compatible with GPUs of different vendors. The adapted code not only surpassed the economic break-even point on both an Intel/Nvidia and an AMD cluster,

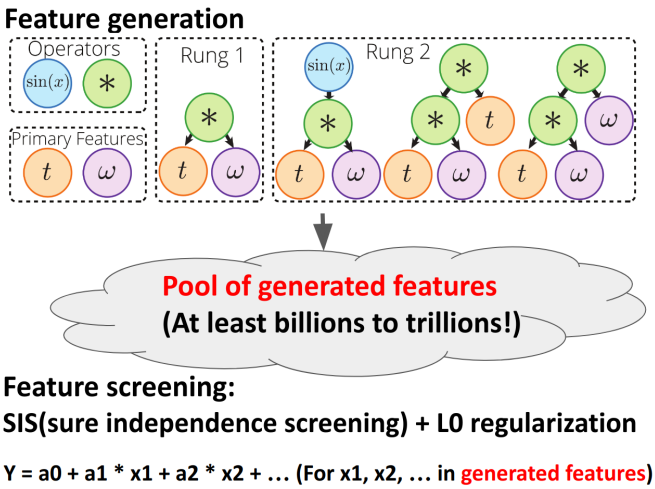

Figure 3. Schema for the SISSO steps.

[2] C. Draxl, and M. Scheffler. "Big data-driven materials science and its FAIR data infrastructure." Handbook of Materials Modeling, 49-73, (2020).

but also retained its near linear scaling behavior. This GPU porting paves the way for applying their explainable AI method in the realm of exascale computing.

Symbolic regression extracts mathematical expressions directly from data, offering benefits such as uncovering new scientific laws and enabling fast inference[3]. The SISSO (sure-independence screening and sparsifying operator) method involves two steps (see Figure 3): creating a pool of analytical expressions through iterative combinations and then selecting optimal candidates for desired properties through analysis of these expressions and their linear combinations[3]. Despite its potency, SISSO's computational intensity arises from the combinatorial scaling of both steps. This computational challenge can be tackled by combining supercomputing with the SISSO method.

NOMAD is leading the development of the SISSO method and the SISSO++ code for large-scale high-performance computing[4]. Our recent successes include the development of SISSO++, a new implementation of SISSO in C++. It has a user-friendly python interface, and is also designed to use both OpenMP and MPI for shared memory and distributed memory parallelization. As shown in Figure 4, our implementation in SISSO++ demonstrates near-perfect strong scaling, indicating its excellent performance.

To respond to the changing landscape of supercomputers, which commonly employ a CPU+accelerator architecture, e.g. GPUs, the group of scientists in NOMAD and MPCDF have integrated an enhanced algorithm into SISSO++. This algorithm uses the Kokkos performance-portable programming model[5] to offload the performance-critical part of their algorithm onto accelerators such as Nvidia, AMD or Intel GPUs. One notable

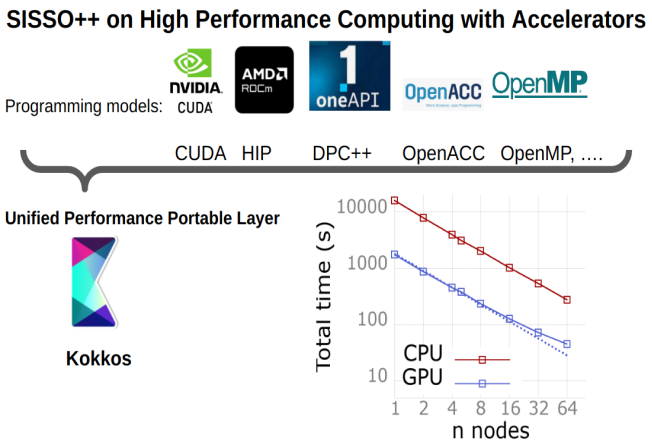

Figure 4. SISSO++ implementation on accelerators.

advantage of using the Kokkos model is that they needed to maintain a single codebase that is compatible with GPUs of different vendors.

With this approach, the researchers have achieved about 8x speedup on nodes with 40 Intel Xeon Skylake CPU cores and 2 Nvidia V100 GPUs compared to the CPU implementation, and a 5.5x speedup on nodes with 96 AMD EPYC CPU cores and 4 AMD MI250 GPUs. This speedup already exceeds the estimated economic break-even point (3x speedup on the Intel/Nvidia cluster and 5.2x speedup on the AMD cluster) for the GPU cluster they are employing. For the Nvidia GPUs this is most significant, while for the comparably new AMD GPUs they expect further speedups by application tuning and improvements of the Kokkos backend for AMD. In addition, the resulting code still shows linear scaling with respect to the number of nodes used.

We are currently working on adapting an enhanced version of the SISSO method – the parametric SISSO method – to accelerators. The parametric SISSO method introduces the flexibility of non-linear parameters into the SISSO method, enabling the algorithm to discover improved models[6].

[3] R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler, and L. M. Ghiringhelli. Physical Review Materials **2**, 083802, (2018).

[4] T. A. R. Purcell, M. Scheffler, C. Carbogno, and L. M. Ghiringhelli, Journal of Open Source Software **7**, 3960, (2022).

[5] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, et al.," IEEE Transactions on Parallel and Distributed Systems **33**, 805-817, (2021).

[6] T. A. R. Purcell, M. Scheffler, and L. M. Ghiringhelli J. Chem. Phys. **159**, 114110 (2023).

WORKFLOWS BEYOND DFT

David Waroquiers

Workflows are crucial for fully harnessing the power of exascale computers in computational materials science. While advances in algorithms, parallelization, and optimization have allowed for the efficient execution of single beyond-DFT calculations across tens of thousands of CPUs, workflows play a vital role in achieving two important goals:

1. Enabling the concurrent execution of tens of thousands of such calculations, to facilitate scalability and high-throughput computational studies.

2. Automating the management of complex interdependencies and error handling among these calculations.

Developing workflows for complex simulations is challenging due to the complexity of existing frameworks. Therefore, the NOMAD CoE has dedicated its efforts to streamline and adapt pipelining tools for beyond-DFT calculations within Work Package 5. The goal goes beyond mere workflow development, as automatic convergence and restart procedures are being incorporated to enhance calculation efficiency and reliability.

As part of WP4 and WP5, a novel framework called Jobflow (JF) has been developed to facilitate the implementation of workflows. JF offers increased flexibility and modularity by enabling the seamless integration and encapsulation of workflows (see Figure 5), so it can handle not only single calculations but also complex logic, such as dynamic generation of new workflows and automatic convergence on parameter grids.

Significant progress has been made in Work Package 5, which addresses workflows beyond DFT, particularly in the development of workflows for GW approximation.

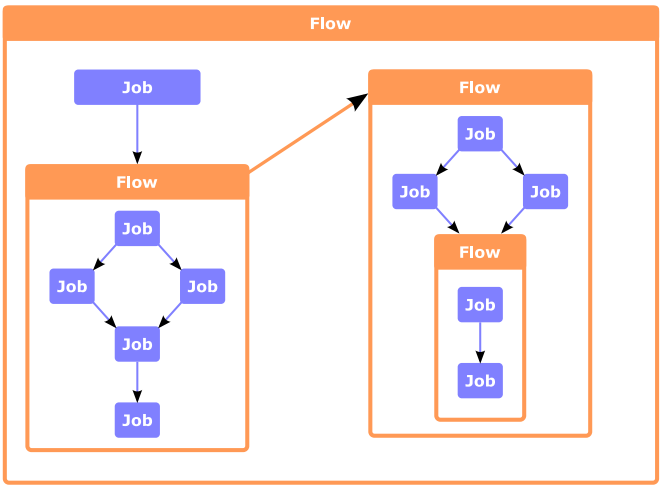

Figure 5: Schematic of a complex workflow with encapsulation.

Automatic convergence of GW calculations can be easily performed thanks to the new JF framework, allowing the efficient testing of multiple convergence parameters. This procedure has been optimized to minimize the number of calculations required (see Figure 6).

Regarding the Coupled Cluster method, it was necessary to develop a Python interface that facilitates the generation of input files and extraction of information from output files produced by the package Cc4s (Coupled Cluster for Solids). This requirement led to the creation of the pycc4s package, which automates these tasks by providing functionalities for input file creation and output parsing. Workflows for Cc4s are currently

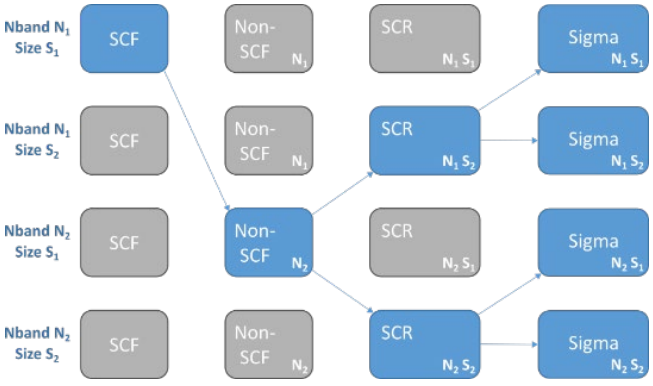

Figure 6: Smart implementation of a convergence workflow for GW calculations. We assume that the dielectric matrix will be computed with two different numbers of bands (N2>N1) and two different sizes (S2>S1). With this implementation, all the calculations in gray are skipped.

under development. These will also incorporate features such as automatic convergence, allowing for the exploration and optimization of calculation parameters to ensure accurate and reliable results.

EXTREME CAPABILITY COMPUTING ON LUMI-G ENABLED BY NOMAD WORKFLOW SOFTWARE

Kristian S. Thygesen

To fully exploit the potential of exascale computing, it is essential to utilize GPUs. To illustrate: The supercomputer LUMI has 98.7% of its FLOPS on the 2560 GPU nodes of the LUMI-G partition (with 0.428 exaflops of pre-exascale LINPACK performance), despite the LUMI-C partition having over 196,000 CPU cores! Hence, all purely CPU/MPI scaling codes are limited to the petascale. Secondly, in the exascale regime, it is not only important to perform a small quantity of jobs efficiently, but one must recognize that managing the plethora of jobs is beyond the control of a single person. Any attempt to do so would not only reduce efficiency but also lead to human error. Consequently, efficient workflow software is required for the handling and real-time documentation of progress, keeping track of metadata, and managing the results.

One of the goals of the NOMAD CoE is to develop a workflow framework that allows users to define complex computational workflows and execute them on supercomputers with minimal human intervention. The workflow framework should be independent of the specific material simulation code so that workflows developed by user A with simulation code B can be easily adopted by user C with simulation code D.

TaskBlaster[7] is an open Python framework for defining and managing simulation workflows. It is fully generic in that it does not depend on any external simulation code, but can be used to setup and execute any type of workflow. A TaskBlaster workflow consists of a series of Tasks, which are Python functions along with a specification of their inputs. One Task can take another Task as an input, thereby defining a dependency graph. Tasks can be created before workflow execution (static Tasks) or during workflow execution (dynamic Tasks). TaskBlaster keeps track of the status of Tasks (new, queued, running, done, failed), submits Tasks as jobs according to the dependency graph, and stores relevant metadata to ensure reproducibility. In short, TaskBlaster makes it easy to setup and run complex computational workflows involving thousands of possibly interdependent tasks. Because managing the workflow involves a lot of record keeping (reading/writing in a database), it is critical to ensure that the workflow management itself does not become a bottleneck under exascale conditions.

Figure 7: The NOMAD-DTU team behind the (pre)exascale demonstration of the TaskBlaster workflow framework. From top left to bottom down: Jens Jørgen Mortensen (software developer), Kristian Sommer Thygesen (Prof. NOMAD PI), Tara Boland (postdoc), Ask Hjorth Larsen (postdoc), and Mikael Kuisma (postdoc).

[7] See: https://gitlab.com/taskblaster/taskblaster

In March 2023, the NOMAD DTU team demonstrated the successful scaling of TaskBlaster to the entire LUMI-G (approx. 10k GPUs). The team was given exclusive access to the LUMI-G for a six-hour slot. After a hectic start with several technical problems, most issues were resolved and for the last hour the workflow filled the entire LUMI computer with about 10k structure optimizations running simultaneously.

The structure optimizations were performed using the GPU version of the GPAW electronic structure code[8]. The aim was to obtain the equilibrium lattice constants and atomic structure of 100,000 inorganic materials. However, due to the initial technical problems only about 20,000 materials were obtained. Although this is far below the initial goal, it is still an impressive outcome considering this was achieved in just one hour!

As previously mentioned, TaskBlaster is fully simulation code-agnostic. However, concrete TaskBlaster workflows will be code specific in practice owing to the large variation in the user interfaces of different simulation codes. More simulation code-agnostic, and thus shareable, workflows can therefore be achieved by defining and implementing a common API for simulation codes. In NOMAD, we are working to achieve this by building on the functionality existing in the Atomic Simulation Environment (ASE). ASE is already widely used by the community and has Python interfaces to more than 30 atomistic simulation codes. However, the ASE interfaces differ significantly from code to code and are often very rudimentary. NOMAD continues to work with code developers representing the major simulation codes to develop more functional and standardized Python interfaces.

OUR LIBRAIRES

Information about our publicly released libraries can be found on: https://www.nomad-coe.eu/nomad-coe/outreach-nomad-coe/libraries-nomad-coe

OUR EVENTS

Keep track on our upcoming events, such as the online Industry workshop (Jan 2024), and the Abinit School (Jan/Feb 2024). For more detailed information and other events, see https://www.nomad-coe.eu/nomad-coe/events-nomad-coe.

INTERNSHIPS FOR FEMALE RESEARCHERS

You are a female junior scientist at master's or PhD level or early postdoc? Join us for an internship to explore NOMAD CoE's exascale efforts! Have a look at https://www.nomad-coe.eu/nomad-coe/outreach-nomad-coe/opportunities-women-nomad-coe.

GET IN TOUCH WITH US

https://www.nomad-coe.eu/

@NoMaDCoE

The NOMAD Laboratory

The NOMAD LinkedIn page

Facebook page

contact@nomad-coe.eu

This project receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951786.

[8] See: https://wiki.fysik.dtu.dk/gpaw/