Check for updates

SOFTWARE TOOL ARTICLE

REVISED The NOMAD mini-apps: A suite of kernels from ab initio electronic structure codes enabling co-design in high-performance computing

[version 3; peer review: 3 approved]

Isidre Mas Magre, Rogeli Grima Torres, José María Cela Espín, José Julio Gutierrez Moreno

Barcelona Supercomputing Center (BSC), Plaça Eusebi Güell, 1-3, Barcelona, 08034, Spain

v3 **First published:** 19 Feb 2024, **4**:35
https://doi.org/10.12688/openreseurope.16920.1
Second version: 29 May 2024, **4**:35
https://doi.org/10.12688/openreseurope.16920.2
Latest published: 10 Apr 2025, **4**:35
https://doi.org/10.12688/openreseurope.16920.3

Abstract

This article introduces a suite of mini-applications (mini-apps) designed to optimise computational kernels in *ab initio* electronic structure codes. The suite is developed from flagship applications participating in the NOMAD Center of Excellence, such as the ELPA eigensolver library and the *GW* implementations of the exciting, Abinit, and FHI-aims codes. The mini-apps were identified by targeting functions that significantly contribute to the total execution time in the parent applications. This strategic selection allows for concentrated optimisation efforts. The suite is designed for easy deployment on various High-Performance Computing (HPC) systems, supported by an integrated CMake build system for straightforward compilation and execution. The aim is to harness the capabilities of emerging (post)exascale systems, which necessitate concurrent hardware and software development — a concept known as co-design. The mini-app suite serves as a tool for profiling and benchmarking, providing insights that can guide both software optimisation and hardware design. Ultimately, these developments will enable more accurate and efficient simulations of novel materials, leveraging the full potential of exascale computing in material science research.

Keywords

High-performance computing, exascale, ab initio calculations, materials science, eigensolver library, Green's function methods, co-design, mini-apps

Open Peer Review

Approval Status ✓ ✓ ✓

	1	2	3
version 3 (revision) 10 Apr 2025			✓ view
version 2 (revision) 29 May 2024	✓ view	✓ view	
version 1 19 Feb 2024	✗ view	? view	✗ view

1. **Ernesto Dufrechou**, Universidad de la Republica Uruguay, Montevideo, Uruguay
2. **Jerzy Proficz**, Politechnika Gdanska, Gdańsk, Poland
3. **Sarah Neuwirth**, Johannes Gutenberg University, Mainz, Germany

Any reports and responses or comments on the article can be found at the end of the article.



This article is included in the Horizon 2020 gateway.



This article is included in the Horizon Europe gateway.



This article is included in the Software gateway.

Corresponding author: José Julio Gutierrez Moreno (julio.gutierrez@bsc.es)

Author roles: Mas Magre I: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Grima Torres R**: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Supervision, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Cela Espín JM**: Conceptualization, Formal Analysis, Funding Acquisition, Investigation, Methodology, Project Administration, Resources, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Gutierrez Moreno JJ**: Conceptualization, Formal Analysis, Investigation, Methodology, Project Administration, Resources, Software, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work has been funded and developed within the framework of the NOMAD Centre of Excellence, funded by the European Union's Horizon 2020 research and innovation program under grant agreement No. 951786. We acknowledge partial funding from the European Union's MaX CoE, under grant agreement No. 101093374, co-funded by the European HPC Joint Undertaking (JU) and participating countries, with the contribution of the Spanish State Research Agency under grant reference PCI2022-135083-5602; and from the Machine-learning-driven bottom-up design of atomically-layered heterostructures for green production (MLALH) project, awarded by the EIG CONCERT-Japan call with reference PCI2023-143426 funded by MCIN/AEI/10.13039/501100011033 and the European Union. We acknowledge EuroHPC Joint Undertaking for awarding the project ID EHPC-DEV-2024D02-026 access to LUMI at CSC, Finland.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2025 Mas Magre I *et al*. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Mas Magre I, Grima Torres R, Cela Espín JM and Gutierrez Moreno JJ. **The NOMAD mini-apps: A suite of kernels from ab initio electronic structure codes enabling co-design in high-performance computing [version 3; peer review: 3 approved]** Open Research Europe 2025, **4**:35 https://doi.org/10.12688/openreseurope.16920.3

First published: 19 Feb 2024, **4**:35 https://doi.org/10.12688/openreseurope.16920.1

REVISED **Amendments from Version 2**

Grant information has been corrected, since the MaX co-funding was not correctly cited, and part of the access to HPC resources were not mentioned. There are not any changes in the manuscript.

Any further responses from the reviewers can be found at the end of the article

Introduction

Exascale computing represents a significant advancement in High-Performance Computing (HPC), unlocking unprecedented opportunities to transform computational modelling. In this context, *ab initio*-based materials modelling codes are in the ideal situation to take advantage of this revolution, which will provide all the resources to enable more accurate calculations, larger size scales and high-throughput exploration of data sets for the discovery of novel materials[1]. However, the distinguishing features of these new supercomputers include increased heterogeneity in their architectures, often incorporating specialised accelerators designed for specific applications[2]. To fully realise the potential of the exascale era, developers of *ab initio* electronic structure codes are investing efforts in exploiting the advanced capabilities of the new supercomputers[3]. Some efforts include the development of low-scaling algorithms for existing methods as well as developing efficient shared libraries for the most critical and computationally expensive parts of the codes. Some examples of popular libraries are; ELPA[4], an efficient eigensolver for petaflop (and exascale) systems, libxc[5], a library of exchange and correlation functionals, GreenX[6], an open-source library that supports exascale implementations of Green's-function-based methodologies, SIRIUS[7], a domain-specific library for electronic structure calculations, ELSI[8], an open infrastructure for electronic structure solvers, and PEXSI[9,10], a fast method for electronic structure calculation based on Kohn-Sham density functional theory (DFT). Many efforts are also focused on the development of workflow managers and job schedulers for high-throughput computations (HTC)[11], some examples are AiiDA[12], the Atomic Simulation Environment (ASE)[13] and Atomic Simulation Recipes (ASR)[14], or FireWorks[15]. With all these active developments in software and hardware, co-design is also an important effort that is bound to play a crucial role in an efficient transition to the (post)exascale era.

Co-design is intended to facilitate the paradigm shift by concurrently and cooperatively developing both, hardware and software[16]. There are some tangible examples of co-design efforts focused on atomic-scale simulations. For example, while the Gromacs molecular dynamics (MD) code was ported to GPUs, NVIDIA also introduced stream priority bits in their hardware, which eventually benefited the communications and led to better code performance[17]. Another interesting example is the Anton supercomputer, which was originally designed to efficiently run classical MD simulations[18]. For this purpose, and being one of the most relevant applications and family of applications with more users in the HPC community, the performance of *ab initio* electronic structure codes has to be systematically profiled on new infrastructures. Therefore, effective communication between hardware engineers, software developers and users is especially critical during the systems' design process. However, the extreme complexity of these codes, being many of them collaboratively developed among large research teams with rotating staff and for many decades, makes these tasks quite challenging. Therefore, it is always important to set a visible milestone in the current software status and try to achieve the optimum performance of novel HPC features at the earliest possible stage, when low-level (i.e. bit-level or compiler) adaptations are still possible. Otherwise, benchmarking would only be possible once the development of the system approaches production capabilities. This problem can primarily be addressed using simplified models, such as mini-apps, which are proxies of full code executions[19]. An overview of the co-design workflow and the mini-app suite presented in this paper is shown in Figure 1.

Mini-apps strive for a simple compilation and small computing times, enabling a rapid iteration of code and leaving room for low-level improvements in hardware at experimental platforms. A mini-app comprises a small fraction of the code length and complexity with respect to its parent application while it retains the primary performance-intensive aspects. Therefore, mini-app benchmarking can inform the implementation of new HPC systems more effectively. Ideally, this would be a continuous process throughout the implementation and in the initial design phase of the systems. Mini-apps

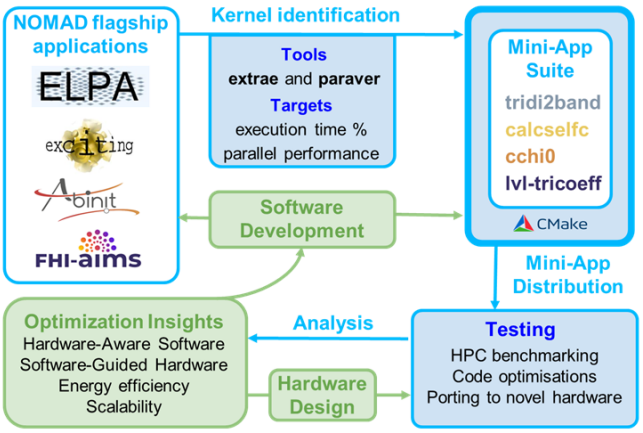


Figure 1. Overview of the co-design workflow for NOMAD CoE flagship applications. The top panels show the one-to-one correspondence between codes and kernels extracted for the mini-apps. Performance analysis tools and metrics are also indicated. Examples of current usage examples of the suite are displayed in the testing section. Possible optimisation targets are displayed in the green panels.

have the potential to support our efforts in co-design as the performance metrics will be easily transferable between selected HPC systems and pre/post-exascale prototypes more effectively. Moreover, mini-apps can also serve to inform developers of the parent applications of the feasibility of potential directions for future developments. An illustrative example of a mini-app suite is the one developed by the *arch* project, which unified several physical simulations such as heat transfer, gravity or hydrodynamics problems within a consistent coding practice under a common infrastructural layer[20]. Also, contributors to the *Mantevo* project have focused on developing tools to accelerate and improve the HPC by providing application and library proxies. This suite includes some applications that could be useful for materials research, such as classical MD or finite elements mini-apps[21]. Also, a methodology paper for showing the link between full application codes and their proxies was published[22]. In addition to all these, and following a similar spirit, the Sustained System Performance (SSP) (and its simplified (SSSP) metric) enable performance projection that correlates with full applications from a suite of mini-apps, providing a more direct estimation of the application performance in contrast to for example the popular Linpack benchmark[23].

On this basis, our article presents a suite of mini-apps developed from a set of representative *ab initio* electronic structure codes that are part of the Novel Materials Discovery (NOMAD) Centre of Excellence (CoE)[24], all using different basis sets and aiming to be a seed of collaboration in the co-design endeavour. The objective of the project is to facilitate systematic studies and predictions of novel materials enabled by upcoming exascale computing. In the following sections, first we introduce the importance of eigensolvers in electronic structure calculations and sketches the *GW* approximation to serve as context and reference to present the kernels identified for each mini-app. The next section describes the methods used to profile the codes, identify the kernels and develop the mini-apps suite. Then we proceed to describe the suite and how to operate the mini-apps within the suite. In the end, we give some conclusions and motivate further research that could be carried out using the NOMAD mini-apps.

Theoretical background

The mini-apps currently present in the suite correspond to isolated kernels both from the ELPA library (version 2022.05.001), an efficient eigenvalue solver for HPC applications[4,25], and from the *GW* implementation of the *ab initio* codes exciting (oxygen version)[26,27], which uses all-electrons with linearised augmented plane-wave plus local orbitals (LAPW+lo), Abinit (version 9.6.2)[28–30], which uses plane waves and pseudo-potentials (PW+PP), and FHI-aims (version 210716_1)[31,32], which uses all-electrons with numeric atom-centered orbitals (NAOs).

Eigenvalue problems

Eigenvalue problems are common tasks in ab initio electronic structure calculations when solving the Schrödinger equation or approximations of it, such as those found in Density Functional Theory (DFT) and many-body perturbation theories (MBPT) like the *GW* approximation. Eigensolvers are often the main computational bottleneck in Density Functional calculations, expending in large cases the vast majority of the total computational cost and, in practice, also limiting the systems' sizes. For researchers in the field of computational materials science, an efficient and scalable solution to the eigenvalue problem is thus of major importance. The ELPA library(*E*igenvalue *So*L*v*ers for *P*etaflop-*A*pplications) is designed for exascale HPC. The ELPA scalability and parallelisation capabilities make it a key tool for handling computationally intensive tasks in material discovery and design[4]. ELPA is a well-established solver library, and today, it has interoperability with the majority of the most widely used *ab initio* packages, making it an indispensable component in the computationaltoolkit for advancing material science research.

The *GW* approximation

The *GW* approximation of Hedin's equations represents a significant advancement in accuracy for electronic structure calculations, for example, enhancing the prediction of band gaps in semiconducting materials[33]. This improvement is notable when compared to traditional Density Functional Theory (DFT) with Local Density Approximation (LDA) or General Gradient Approximation (GGA) functionals. A key element of this advancement is the introduction of the self-energy, Σ, in Hedin's equations, which encapsulates all electron-electron interactions extending beyond the Hartree energy. This incorporation addresses the limitations in conventional DFT methods, which often underestimate electron correlation effects crucial for accurate band gap predictions.

In practical implementations, especially where computational efficiency is paramount, the self-consistent *GW* formalism often gives way to a simplified approach termed single-shot *GW* or G_0W_0. This approximation involves using the non-interacting Green's function, G_0, in place of the fully interacting G. The equation for G_0 is given by:

$$G_0(\boldsymbol{r},\boldsymbol{r}';\omega) = \sum_{nk} \frac{\psi_{nk}(\boldsymbol{r})\,\psi^*_{nk}(\boldsymbol{r}')}{\omega - \epsilon_{nk} - i\eta} \tag{1}$$

Here, ψ_{nk} $(\boldsymbol{r})$ and ε_{nk} represent the DFT eigenfunctions and eigenvalues, respectively, with n and k denoting band and k-point indices in the Brillouin zone and ω is the frequency of the Green's function. The term η is a small, positive (negative) number for occupied (unoccupied) states, ensuring numerical stability.

Similarly, the screened Coulomb interaction W in the *GW* formalism is approximated by W_0, described as:

$$W_0(\boldsymbol{r},\boldsymbol{r}';\omega) = \int v(\boldsymbol{r}_1,\boldsymbol{r}')\epsilon^{-1}(\boldsymbol{r},\boldsymbol{r}_1;\omega)\,\mathrm{d}\boldsymbol{r}_1 \tag{2}$$

In this context, $v(\boldsymbol{r}, \boldsymbol{r}')$ denotes the bare Coulomb interaction, and $\varepsilon^{-1}(\boldsymbol{r}, \boldsymbol{r}_1; \omega)$ is the inverse dielectric function, reflecting the material's response to electronic perturbations.

More detailed descriptions of the theory and applications of the *GW* method can be found in more specific publications[34,35].

Methods

The identification of the relevant kernels on the exciting, Abinit, FHI-aims codes and the ELPA library was done by executing and profiling benchmarks in the MareNostrum-4 supercomputer.

For this we used a set of profiling tools:

- Extrae[36] is a tracing tool. It collects information such as execution times, MPI and OpenMP calls, and performance counter information with a Performance Application Programming Interface (PAPI)[37], supplying a consistent interface and methodology for collecting performance counter information from various hardware and software components.
- Paraver[38,39] takes traces generated with Extrae and provides a visual interface to analyse them. Trace figures in the manuscript were produced with this software.

These tools identified relevant kernels by comparing execution times and performance metrics related to their parallel performance (load imbalance, parallel efficiency, etc.). The compute-intensive regions of the code were found to be associated with the relevant computations in the implementations above. We have performed exhaustive performance analyses using different test cases for each of the applications and performed scaling tests on the number of MPI/OMP processes to accommodate the available memory resources within a reasonable time to solution. We selected at least two realistic test cases for all the applications with increasing complexity computational cost. In ELPA, we used matrices of different sizes. For the *GW* applications, we selected ZrO_2 as a common test case for all the codes, which is a technologically relevant semiconductor with applications in catalytic green energy production. The alternative test cases were differently adapted for each code and described in their corresponding sections. Once kernels in the original source code are identified, the mini-apps are developed through the migration of these to a stand-alone implementation. Data dependencies are addressed through the insertion of a checkpoint right before the kernel in the original execution, which captures all the relevant variables and parameters and serves as the input file for the mini-apps. To avoid extra dependencies, such as HDF5 or other common checkpointing modules, the checkpoints are implemented through a binary I/O wrapper included in the mini-apps distribution.

The mini-apps are integrated into a CMake build system, ensuring ease of compilation across a variety of architectures in HPC environments. All mini-apps are written in FORTRAN, the language that is mostly used in their parent applications. Our testing has covered a range of machines, including those with Intel, POWER9, and AMD processors. Given the diversity of these architectures, it is certainly worthwhile to maintain this broad testing approach. The full suite of mini-apps is compiled simultaneously, streamlining the setup process as the environment remains consistent for all mini-apps during testing. This integrated build system not only simplifies the initial setup but also enhances the suite's extensibility for incorporating new mini-apps or updating existing ones.

Mini-apps suite

The Mini-apps Suite consists of four mini-apps. Each mini-app is described below by naming the source file and subroutines containing the selected kernel and a justification for its selection. The kernels might have dependencies to other code regions; those are assumed to be included in the mini-app as well, but special care was taken not to include dependencies that are not strictly required and to reduce the mini-app source code to the minimum.

ELPA mini-app

The ELPA mini-app isolates the *trans_ev_tridi_to_band* subroutine, located in the source code *elpa2_trans_ev_tridi_to_band_template.F90*. The ELPA library implements different methods to solve the eigenvalue problem; among those methods, we selected the two-stage tridiagonalisation, which includes an intermediate reduction to bidiagonal form before doing a reduction to tridiagonal form, as opposed to the one-stage method, which reduces the original matrix directly to tridiagonal form[25]. The two-stage method is normally preferred to the one-stage one in large problems and when most of the eigenvectors need to be computed. Other choices made in the method selection are the type and precision of the numbers. Our mini-app uses real numbers with double precision and employs a generated random matrix as input.

The kernel selection was done after profiling two different executions. The first was a medium-sized problem with a matrix size of 8000 × 8000 executed with a block size of 16 on 16 processors with an execution time of 8.3 s (see Figure 2), the second was a large matrix of size 100000 × 100000 executed with a block size of 24 on 192 processors with an execution time of 969 s. Both traces showed similar behaviour in terms of task distribution and relative weight of the routines in the trace.

After analysing the several steps to compute the two-stage method, we concluded that the most suitable function for the mini-app is *trans_ev_tridi_to_band*. There are some good reasons for this selection. First, this is the most computationally expensive step in both experiments, becoming more important as the system size increasing. In particular, our perfornace analyses show that *trans_ev_tridi_to_band* takes 32% of the total time in the smaller test case, and 39% in the large one. Also, it does not depend on external functions, while other functions heavily rely on external libraries such as BLAS, ESSL or KML. The function is not dominated by DGEMM or communications. It is also relevant to note that the developers have put substantial efforts into improving this subroutine on multiple architectures. In the hypotetical case of

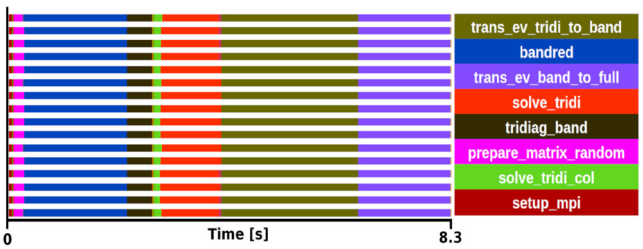


Figure 2. Trace of ELPA run on a matrix size of 8000 × 8000 with the AVX512 kernel. The runtime was 8.3 s in 16 one-threaded MPI processes.

not having this routine optimally vectorized for a given architecture this, kernel would become largely predominant over other subroutines. Therefore, its relative weight makes it a good target to test on experimental hardware where vectorization might not be supported yet. Today, it supports SSE, AVX(2/512), SPARC64 SSE, ARM SVE(128/256/512), BlueGene/(P/Q), NVIDIA, AMD and Intel GPUs.

exciting mini-app

The exciting mini-app isolates the subroutines *expand_products* and *calcmwm* called from the source code *calcselfc.f90*. This source code is called in the last step for the calculation of the self-energy in the main loop of the Brillouin zone's integration[26]. The selection of the kernel was done after profiling different executions of the benchmark ZrO_2 primitive cell (3 atoms) with a 2 × 2 × 2 q/k-point grid, one with all (800) bands and another with a reduced number (12) of bands, see Figure 3 and Figure 4 for the respective traces.

The benchmarks were executed using 4 MPI processes with 48 OpenMP threads each. It was observed that the main loop for the Brillouin Zone was distributed in 8 tasks (1 task per k-point) among the available MPI processes. The execution time of the G_0W_0 implementation of exciting is therefore explained by the duration of one of these tasks. Each of these tasks is composed of the computations of the necessary quantities in the G_0W_0 formalism using an auxiliary mixed product basis. In all these computations, a product expansion is needed to perform the calculations, and therefore, the *expand_products* subroutine is a common call in all the calculations. Here we should remind that as a beyond DFT method, the G_0W_0 requires a previous calculation of the density matrix at the ground state (*scf_cycle*) which is not considered for the development of the exciting, Abinit, and FHI-aims mini-apps. The amount of calls to *expand_products* and its execution time depends on the subroutine and the system's size. For the benchmark, it was decided to focus only on the duration and performance of the *GW* implementation, not taking into account the ground state section. When all the bands are used, the subroutine *calcminm2* (magenta sections in Figure 3) called within *expand_products*, used for the calculation of the self-energy, is the most time-consuming section, increasing the total runtime to 3214 s. This subroutine alone takes 85% of the G_0W_0 task. Moreover, considering that *calcmwm* takes another 5% of the G_0W_0 task, the exciting mini-app presented here represents 90% of the G_0W_0 implementation for this specific benchmark. The other 10% is devoted to the calculation of other G_0W_0 quantities, which also execute the subroutine *expand_products* and are expected to become proportionally more relevant depending on the number of bands used in the calculation. In the reduced case (Figure 4), the routines included in the mini-app (*expand_products and calcwm*) still consume a 58% of the G_0W_0 task execution time.

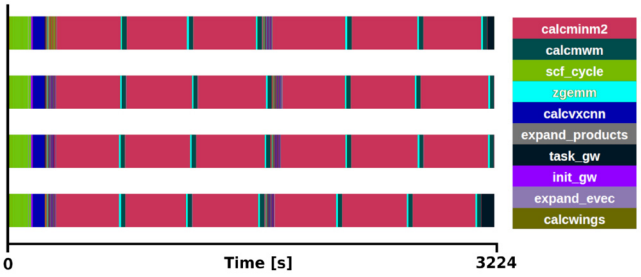


Figure 3. Trace of exciting full execution. ZrO_2 primitive cell (3 atoms) with 2×2×2 q/k-point grid and 800 bands. The runtime was 3214 s with 4 MPI processes and 48 threads each.

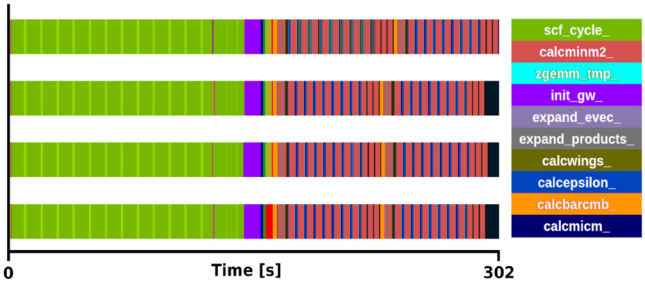


Figure 4. Trace of exciting full execution. ZrO_2 primitive cell (3 atoms) with 2×2×2 q/k-point grid and 12 bands. The runtime was 302 s with 4 MPI processes and 48 threads each.

Abinit mini-app

The Abinit mini-app isolates one iteration of the loop over q-points for the polarizability χ_0 calculation in the screening step. This is a triple loop that iterates over k-points, conduction, and valence bands, these quantities are nested forming the triple loop[29]. This kernel is located in the *m_chi0.F90* source code inside the *cchi0* subroutine which is called for every q-point different from the Γ-point. The benchmark was executed in 48 one-threaded MPI processes and the runtime was 924 s for the screening step, from which 88% executes the subroutine *cchi0* over 35 q-point iterations. Figure 5 shows the trace of this execution.

The mini-app includes the initialisation of the *bbp_ks_distrb* matrix, which reduces the size of the checkpoint by around two orders of magnitude, from GB to a few MB. Then, it executes a loop over k-points and bands to calculate the *chi0* matrix. Within this loop, two subroutines were identified as the main computational kernels, namely *rho_tw_g* and *assemblychi0_sym*, which together use 70% of the execution time for *cchi0*, while the remaining 30% are memory accesses and variable updates for each iteration in the nested loops. The selection of the kernel was done after profiling the

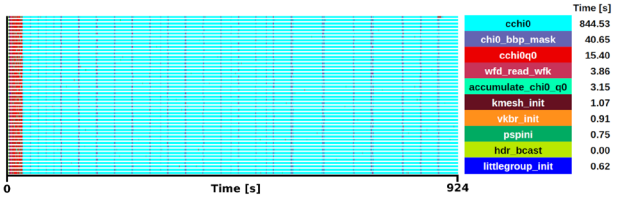


Figure 5. Trace of Abinit execution of the screening calculations for 35 q-points in a ZrO_2 system. The runtime was 924 s in 48 one-threaded MPI processes.

different steps involved in the *GW* calculations for a small 3-atom ZrO_2 and a large 11-atom $Zr_2Y_2O_7$ primitive cell system, from which it was observed that the calculation of the screening W_0 is the most time-consuming for the G_0W_0 implementation.

In this case, the extraction of only one of these q-point iterations and the portion executed in a single thread is enough to represent the full code execution, as there are no MPI communications within the selected kernel. With this selection of kernel, the mini-app reduces an execution of 15 minutes running in 48 MPI processes to 14 seconds in a single process, while it still represents the 88% of the code execution time. For simplicity, the $Zr_2Y_2O_7$ the calculation was done at 3 q-points, being one of the Γ-point. In this reduced case, the chi0 routine takes a 42% of the computational time, however, if we extrapolate this to a potentially extended calculation using a 36 q-point grid, we expect the routines in the mini-app to represent a 92% of the total execution.

FHI-aims mini-app

The FHI-aims mini-app isolates the routine that computes the LVL triple expansion coefficients in real space, and Fourier-transforms the LVL triple coefficients, which is called *gw_init_lvl_tricoeff_recip*. These coefficients expand the product of wavefunctions to the so-called Auxiliarity Basis Functions (ABF). These ABF are atomically-centred, and they are constructed similarly to the mixed product basis in the LAPW framework but without the plane-wave component in the interstitial region. This basis set expansion is used to achieve efficient implementations of Hartree-Fock, second-order Moller-Plesset perturbation theory (MP2), the Random Phase Approximation (RPA), and *GW* within the numerically tabulated atom-centered orbitals (NAO) basis set framework[32,40]. This renders the mini-app as a relevant kernel for all these methods. The FHI-aims mini-app tested a code version that is no longer current and has since been optimised, particularly in the LVL part. While this older version of the code is not distributed anymore, the mini-app can still be useful to benchmark different hardware platforms in a practical computational case. The selection of the kernel was done after profiling a simple Si system in 48 MPI processes and a more complex ZrO_2 system in up to 384 MPI processes. Checkpoint inputs for both cases have been included in the repository. For simplicity of visualisation, the trace of the Si benchmark is shown in Figure 6.

From the trace, two initial candidates were identified as potential kernels for the mini-app, with comparable execution times: *gw_init_lvl_tricoeff_recip*, which includes the subroutine *my_get_pairwise_coeff_3fn* (shown in magenta Figure 6) and *evaluate_periodic_gw_selfenergy*, which includes the subroutines *evaluate_polarisability_kspace* and *evaluate_gw_selfenergy_band_kpoint* (shown in orange and cyan respectively). Among these, the selection of *gw_init_lvl_tricoeff_recip* was made due to the limited influence of MPI communications (compared to *evaluate_periodic_gw_selfenergy*) and its shared applicability with several beyond-DFT methods. The behaviour and relative weight is consistent in both test cases. The selected kernel in the Si system has an execution time of 127.57 s, which is a 34% of the total execution, while in the ZrO_2 system, the execution time of the kernel was 978.53 s, accounting for a 45% of the total execution.

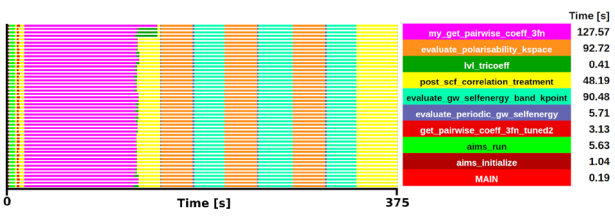


Figure 6. Trace of FHI-aims execution of a Si system. The runtime was 375 s in 48 one-threaded MPI processes.

Operation

The Mini-App Suite is intended to be easily deployed and executed on different machines. To facilitate this, the suite has an integrated compilation and execution scheme for all the mini-apps using a CMake build system. Instructions to build, compile and execute the suite are provided in more detail within the README file included in root of the repository.

Once the suite has been properly built, the checkpoint files (stored in Zenodo[41]) must be downloaded and copied into the benchmark folder found in the repository. This can be directly done by executing the bash script *get-ckpts.sh* inside the benchmarks folder. However, in any case, downloading of the checkpoints should happen automatically when attempting to execute any of the run scripts provided, which are designed to check for the presence of the checkpoints in the expected locations and download them in case they are not found. In case the machine has no internet access, the file can be downloaded following the link provided in the README file. If a user would like to use alternative material benchmark systems, these can be generated by copying the checkpoint generator files placed inside the repository folder *utils* onto the original code, recompiling it and executing the modified version of the code using the desired test case and input parameters. This will generate a new checkpoint file that can be loaded within the mini-apps.

Examples of execution scripts are provided with the benchmark of each mini-app. The corresponding binaries for the *GW* cases must be executed, providing a checkpoint file. The mini-app should be submitted to a queuing system; for that, submission (SLURM) scripts used in MareNostrum-5, MareNostrum-4, and CTE-POWER (BSC, Spain), Leonardo (CINECA, Italy), LUMI (CSC, Finland), Vega (IZUM, Slovenia) and Karolina (IT4I, Czech Republic) are provided. Benchmark scripts launch multiple executions, increasing the number of processes until they fill all the CPUs of the machine. Users should fill a configuration file with some of the specifications of the machine, the available toolchains and specific flags for their queue system. After their execution, the mini-apps will produce a summary report with information on performance metrics such as timings and numerical checks. The information produced from the reports can be readily used to get insights into the performance. The user can, for instance, compare

how the different mini-apps perform on different machines, analyse the potential degradation of the single-core performance with the occupancy of the node, or test different compiling options. More detailed performance analyses of the mini-apps for specific systems are left to be done by experienced users.

Conclusions

The NOMAD mini-app suite, presented in this article, includes four mini-apps extracted from a set of representative flagship codes within the *ab initio* electronic structure community. These mini-apps focus on critical computational kernels involved in the *GW* implementations of exciting, Abinit and FHI-aims, and the ELPA eigenvalue solver. This mini-app suite represents a pragmatic approach to facilitate co-design efforts, employing *ab initio* electronic structure applications as use cases. By isolating and targeting specific computational kernels, this suite not only offers a pathway for focused optimisation but also marks a significant stride in fine-tuning both software and hardware capabilities in tandem. This aspect is particularly relevant as we venture into the era of exascale computing.

The practical benefits of these mini-apps are further enhanced by their adaptability across various HPC platforms. Their user-friendly deployment, facilitated by a streamlined CMake build system, broadens their accessibility to a diverse group of researchers and developers. This accessibility is pivotal in fostering a collaborative environment, essential for the co-evolution of computational tools and hardware technologies. The NOMAD suite are being used to continuously benchmark the supercomputers that are currently being deployed by the European High Performance Computing Joint Undertaking (EuroHPC JU), all having different hardware architectures and compilers/software stacks. These outcomes of these activities provide useful feedback to integrators, system administrators and users. In addition, the ELPA mini-app being used as a case study for the development of novel hardware prototypes, which are currently being designed to be used in future (post-)exascale platforms. These experimental architectures perform normally on emulators or Field Programable Gate Array (FPGA) based single-node platforms with reduced clock time and limited availability in terms of compiler stacks and libraries. Therefore, the initial porting activities of complex codes to these frameworks is only feasible at the mini-app level. This is a clear example of co-design, in which the software is adapted to perform efficiently in these novel architectures, while we are providing constant feedback to the hardware architects and compiler developers at the early stage of development[42].

While these mini-apps serve as a valuable asset in electronic structure calculations, it is essential to recognise that their role is one piece of a giant puzzle. As we continue to explore the vast potential of exascale computing, it is crucial to maintain an ongoing dialogue within the community, ensuring that these tools evolve in response to emerging challenges and opportunities.

We should also recall that while DFT has been so far the main workhorse of the ab initio electronic structure community, its accuracy estimating properties such as electronic band gaps may need to be improved for some applications. Therefore, the use of more accurate (and computationally expensive) beyond-DFT methods such as GW is required. We strongly believe that with the increasing computational power that will come along with the arrival of the (post)exascale era, these methods will increasingly become more accessible and popular among the community, and this is the main reason that drove us to choose GW for the majority of the selection of these mini-apps. It is also important to point out that in addition to the current predominance of GPUs, the future of HPC will also increase in hardware heterogeneity (some of them based on CPUs, which could operate larger vectors, e.g. VPUs) and new programming models. Our mini-apps could also be helpful as a starting point for implementing new porting strategies and optimisations before merging them with the complete code.

The mini-apps suite, the codes that make part of it, and, of course, the whole HPC ecosystem are quite lively research fields and in constant development. Therefore, performance metrics should be properly documented and shared among the community to facilitate the co-design efforts. Our repository is open to incorporating new performance metrics when the mini-apps are executed on new machines. Contributions of new versions that attempt to optimise the existing kernels or new mini-apps addressing other kernels are also expected as the co-design activity develops.

Software availability

The initial release of the mini-apps suite, including the exciting, Abinit and ELPA mini-apps, has been openly stored on https://doi.org/10.5281/zenodo.10412696[41] under license: GNU General Public License v3.0. While the subsequent releases will be updated on this Zenodo repository, the history of changes and up to date developments can be checked from our GitLab repository https://gitlab.bsc.es/material-science/nomad-mini-apps-suite. The corresponding checkpoints for all the GW mini-apps are stored at https://doi.org/10.5281/zenodo.10142416[43] under license: Apache License 2.0 The FHI-aims mini-app is published under its mother code's license, so its access has to be managed through the FHI-aims team at fhi-aims.org.

Acknowledgements

We acknowledge the support from all the developing teams from the codes included in the mini-apps suite. We thank Andreas Marek (ELPA), Claudia Draxl, Andris Gulans and Alexander Buccheri (exciting), Xavier Gonze, Maryam Azizi and Matteo Giantomassi (Abinit), Patrick Rinke and Antonio Delesma (FHI-aims); all for providing the test cases and participating in fruitful discussions on the performance analyses of their respective codes. We also thank the Performance Tools team at the BSC, Judit Giménez and German Llort, and the EU Performance Optimisation and Productivity Centre of Excellence in HPC (PoP-CoE), for their assistance with the performance analyses of the mini-apps. We also thank Albert Farrés for initiating this task within the BSC Materials Science team.

References

1. Gutiérrez Moreno JJ: ***Ab initio* guided atomistic modelling of nanomaterials on exascale high-performance computing platforms.** *Nano Futures.* 2024; **8**(1): 012501.
Publisher Full Text
2. Chang C, Deringer VL, Katti KS, *et al.*: **Simulations in the era of exascale computing.** *Nat Rev Mater.* 2023; **8**(5): 309–313.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Gavini V, Baroni S, Blum V, *et al.*: **Roadmap on electronic structure codes in the exascale era.** *Modelling Simul Mater Sci Eng.* 2023; **31**(6): 063301.
Publisher Full Text
4. Kůs P, Marek A, Köcher SS, *et al.*: **Optimizations of the eigensolvers in the ELPA library.** *Parallel Comput.* 2019; **85**: 167–177.
Publisher Full Text
5. Lehtola S, Steigemann C, Oliveira MJT, *et al.*: **Recent developments in libxc—a comprehensive library of functionals for density functional theory.** *SoftwareX.* 2018; **7**: 1–5.
Publisher Full Text
6. Azizi M, Wilhelm J, Golze D, *et al.*: **Time-frequency component of the Greenx library: minimax grids for efficient RPA and *GW* calculations.** *J Open Source Softw.* 2023; **8**(90): 5570.
Publisher Full Text
7. Zhang L, Kozhevnikov A, Schulthess T, *et al.*: **Performance enhancement of APW+*lo* calculations by simplest separation of concerns.** *Computation.* 2022; **10**(3): 43.
Publisher Full Text
8. Yu VWZ, Campos C, Dawson W, *et al.*: **ELSI — an open infrastructure for electronic structure solvers.** *Comput Phys Commun.* 2020; **256**: 107459.
Publisher Full Text
9. Lin L, Lu J, Ying L, *et al.*: **Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems.** *Comm Math Sci.* 2009; **7**(3): 755–777.
Publisher Full Text
10. Lin L, Chen M, Yang C, *et al.*: **Accelerating atomic orbital-based electronic structure calculation via pole expansion and selected inversion.** *J Phys Condens Matter.* 2013; **25**(29): 295501.
PubMed Abstract | Publisher Full Text
11. Himanen L, Geurts A, Foster AS, *et al.*: **Data-driven materials science: status, challenges, and perspectives.** *Adv Sci (Weinh).* 2019; **6**(21): 1900808.
PubMed Abstract | Publisher Full Text | Free Full Text
12. Uhrin M, Huber SP, Yu J, *et al.*: **Workflows in AiiDA: engineering a high-throughput, event-based engine for robust and modular computational workflows.** *Comput Mater Sci.* 2021; **187**: 110086.
Publisher Full Text
13. Larsen AH, Mortensen JJ, Blomqvist J, *et al.*: **The Atomic Simulation Environment—a Python library for working with atoms.** *J Phys Condens Matter.* 2017; **29**(27): 273002.
PubMed Abstract | Publisher Full Text
14. Gjerding M, Skovhus T, Rasmussen A, *et al.*: **Atomic Simulation Recipes: a python framework and library for automated workflows.** *Comput Mater Sci.* 2021; **199**: 110731.
Publisher Full Text
15. Jain A, Ong SP, Chen W, *et al.*: **Fireworks: a dynamic workflow system designed for high-throughput applications.** *Concurr Comput.* 2015; **27**(17): 5037–5059.
Publisher Full Text
16. Cardwell SG, Vineyard C, Severa W, *et al.*: **Truly heterogeneous HPC: Co-design to achieve what science needs from HPC.** In: Jeffrey Nichols, Becky Verastegui, Arthur 'Barney' Maccabe, Oscar Hernandez, Suzanne Parete-Koon, and Theresa Ahearn, editors, *Driving Scientific and Engineering Discoveries Through the Convergence of HPC Big Data and AI.* Cham, Springer International Publishing, 2020; **1315**: 349–365.
Publisher Full Text
17. Páll S, Zhmurov A, Bauer P, *et al.*: **Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS.** *J Chem Phys.* 2020; **153**(13): 134110.
PubMed Abstract | Publisher Full Text
18. Shaw DE, Adams PJ, Azaria A, *et al.*: **Anton 3: Twenty microseconds of molecular dynamics simulation before lunch.** In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* SC '21, New York, NY USA, Association for Computing Machinery, 2021.
Reference Source
19. Heroux MA, Doerfler DW, Crozier PS, *et al.*: **Sandia report improving performance via mini-applications**.
20. Martineau M, McIntosh-Smith S: **The arch project: Physics mini-apps for algorithmic exploration and evaluating programming environments on HPC architectures.** *2017 IEEE International Conference on Cluster Computing (CLUSTER).* Institute of Electrical and Electronics Engineers Inc. 2017; **2017**: 850–857.
Publisher Full Text
21. Crozier PS, Thornquist HK, Numrich RW, *et al.*: **Improving performance via mini-applications**.
Publisher Full Text
22. Barrett RF, Crozier PS, Doerfler DW, *et al.*: **Assessing the role of mini- applications in predicting key performance characteristics of scientific and engineering applications.** *J Parallel Distrib Comput.* 2015; **75**: 107–122.
Publisher Full Text
23. Tsuji M, Kramer WTC, Sato M: **A performance projection of mini-applications onto benchmarks toward the performance projection of real-applications**. In: *2017 IEEE International Conference on Cluster Computing (CLUSTER).* IEEE, 2017; 826–833.
Publisher Full Text
24. NoMaD: **The novel materials discovery laboratory**.
Publisher Full Text
25. Marek A, Blum V, Johanni R, *et al.*: **The ELPA library: scalable parallel eigenvalue solutions for electronic structure theory and computational science.** *J Phys Condens Matter.* 2014; **26**(21): 213201.
PubMed Abstract | Publisher Full Text
26. Nabok D, Gulans A, Draxl C: **Accurate all-electron G_0W_0 quasiparticle energies employing the full-potential augmented plane-wave method.** *Phys Rev B.* 2016; **94**: 035118.
Publisher Full Text
27. Gulans A, Kontur S, Meisenbichler C, *et al.*: **Exciting: a full-potential all-electron package implementing density-functional theory and many-body perturbation theory.** *J Phys: Condens Matter.* 2014; **26**(36): 363202.
PubMed Abstract | Publisher Full Text
28. Gonze X, Beuken JM, Caracas R, *et al.*: **First-principles computation of material properties: the ABINIT software project.** *Comput Mater Sci.* 2002; **25**(3): 478–492.
Publisher Full Text
29. Gonze X, Jollet F, Araujo FA, *et al.*: **Recent developments in the ABINIT software package.** *Comput Phys Commun.* 2016; **205**: 106–131.
Publisher Full Text
30. Gonze X, Amadon B, Antonius G, *et al.*: **The ABINIT project: impact, environment and recent developments.** *Comput Phys Commun.* 2020; **248**: 107042.
Publisher Full Text
31. Blum V, Gehrke R, Hanke F, *et al.*: ***Ab initio* molecular simulations with numeric atom-centered orbitals.** *Comput Phys Commun.* 2009; **180**(11): 2175–2196.
Publisher Full Text
32. Ren X, Merz F, Jiang H, *et al.*: **All-electron periodic G_0W_0 implementation with numerical atomic orbital basis functions: algorithm and benchmarks.** *Phys Rev Mater.* 2021; **5**: 013807.
Publisher Full Text
33. Hedin L: **New Method for calculating the one-particle Green's function with application to the electron-gas problem.** *Phys Rev.* 1965; **139**: A796–A823.
Publisher Full Text
34. Golze D, Dvorak M, Rinke P: **The *GW* compendium: a practical guide to theoretical photoemission spectroscopy.** *Front Chem.* 2019; **7**: 377.
PubMed Abstract | Publisher Full Text | Free Full Text
35. Aryasetiawan F, Gunnarsson O: **The *GW* method.** *Rep Prog Phys.* 1998; **61**(3): 237.
Publisher Full Text
36. Extrae: **BSC performance tools**. 2023.
Reference Source
37. Mucci PJ, Browne S, Deane C, *et al.*: **Papi: a portable interface to hardware performance counters**. In: *Proceedings of the Department of Defense HPCMP Users Group Conference*. 1999; **710**.
38. Pillet V, Labarta J, Cortes T, *et al.*: **Paraver: a tool to visualize and analyze parallel code**. In: *Proceedings of WoTUG-18: Transputer and Occam Developments*. 1995; **44**: 17–31.
Reference Source
39. Paraver: **BSC Performance tools**. 2023.
Reference Source
40. Ihrig AC, Wieferink J, Zhang IY, *et al.*: **Accurate localized resolution of identity approach for linear-scaling hybrid density functionals and for many-body perturbation theory.** *New J Phys.* 2015; **17**(9): 093020.
Publisher Full Text
41. Mas Magre I, Grima Torres R, Gutiérrez Moreno J: **NOMAD mini-apps suite.** *Zenodo.* [Software]. 2023.
http://www.doi.org/10.5281/zenodo.10412696
42. Mantovani F, Vizcaino P, Banchelli F, *et al.:* **Software Development Vehicles to enable extended and early co-design: a RISC-V and HPC case of study**. *International Conference on High Performance Computing.* 2023; 526–537.
Publisher Full Text
43. Mas Magre I, Grima Torres R, Gutiérrez Moreno J: **NOMAD mini-app suite checkpoints.** [Dataset], *Zenodo.* 2023.
http://www.doi.org/10.5281/zenodo.10142416

Open Peer Review

Current Peer Review Status: ✔ ✔ ✔

Version 3

Reviewer Report 19 May 2025

https://doi.org/10.21956/openreseurope.21749.r53064

© **2025 Neuwirth S.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

✔ **Sarah Neuwirth**
Johannes Gutenberg University, Mainz, Germany

Thank you for submitting the revised version of your paper. I appreciate the thoughtful updates. All of my previous questions and concerns have been addressed. I therefore recommend the paper for acceptance.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Parallel I/O and storage systems, modular supercomputing, performance modeling and analysis, optimization, reproducible benchmarking, and parallel programming models

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 2

Reviewer Report 04 July 2024

https://doi.org/10.21956/openreseurope.19181.r41049

© **2024 Dufrechou E.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

✔ **Ernesto Dufrechou**
[1] Universidad de la Republica Uruguay, Montevideo, Montevideo Department, Uruguay

[2] Universidad de la Republica Uruguay, Montevideo, Montevideo Department, Uruguay

I think the article has improved. The rationale behind the selection of the miniapps is explained with more detail, and the experiments provide reasonable evidence that the performance of the miniapps and the full apps is closely related. Therefore, to integrate these miniapps in a co-design strategy for large-scale HPC platforms is reasonable, perhaps alongside other aspects such as BLAS-LAPACK performance or accelerator capabilities that are useful for other applications.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: I research in heterogeneous computing and HPC. My main lines of work revolve around the accelleration of linear algebra kernels and scientific applications using GPUs.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 04 July 2024

https://doi.org/10.21956/openreseurope.19181.r41048

© 2024 Proficz J. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Jerzy Proficz
[1] Politechnika Gdanska, Gdańsk, Pomeranian Voivodeship, Poland
[2] Politechnika Gdanska, Gdańsk, Pomeranian Voivodeship, Poland

The proposed improvements are sufficient for accepting the paper.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: HPC, energy efficiency, software/hardware optimization, distributed/parallel applications

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 03 May 2024

https://doi.org/10.21956/openreseurope.18286.r38376

© 2024 Neuwirth S. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Sarah Neuwirth
[1] Johannes Gutenberg University, Mainz, Germany
[2] Johannes Gutenberg University, Mainz, Germany

This article introduces the NOMAD mini-applications suite, which is based on kernels from ab initio electronic code structures participating in the NOMAD Center of Excellence. The NOMAD mini-apps suite focuses on four flagship applications, i.e., the ELPA eigensolver library and the GW implementations of the exciting, Abinit, and FHI-aims codes. The authors identify the most relevant kernels of those four applications by profiling runs with different configurations on the MareNostrum-4 supercomputer. The objective of the benchmark suite is to facilitate and enable co-design of future High-Performance Computing (HPC) systems with a focus on ab initio computational materials science codes.

Mini-applications serve as crucial benchmarks in HPC systems due to their ability to simulate specific aspects of real-world applications efficiently. These compact programs encapsulate key computational patterns and performance characteristics, allowing for targeted analysis and optimization. By focusing on critical computational kernels or algorithms, mini-applications provide insights into system behavior, scalability, and bottlenecks, aiding in the evaluation and comparison of HPC architectures and software stacks. Moreover, their simplicity facilitates rapid experimentation and tuning, enabling researchers and engineers to fine-tune system configurations for optimal performance, ultimately advancing the design and deployment of HPC solutions.

Overall, this article addresses a timely topic and describes the four application kernels well. The article has a clear structure and is mostly well written. Nevertheless, some questions remain unanswered after reading the paper and I would like to encourage the authors to clarify these points in a revision, specifically regarding the co-design aspect and applicability to large-scale systems.

Comments and suggestions:
In the introduction, the authors state that “mini-app benchmarking can inform the implementation of new HPC systems more effectively”. Furthermore, the authors say that “Mini-apps have the potential to support our efforts in co-design as the performance metrics will be easily transferable between selected HPC systems and pre/post-exascale prototypes more effectively”. While these statements make sense, there is no further clarification on how the presented mini-apps can benefit a co-design cycle. Also, the authors do not further elaborate about selected or targeted performance metrics, even though they mention the (simplified) Sustained System Performance ((S)SSP). To further emphasize the impact of mini-app suites such as the NOMAD suite, it may be helpful to further explain performance metrics for the broader audience.

Although the authors provide a good description of their kernel selection process based on the traces, it is not clear how representative these are for the overall performance of the application.

For example, although the ELPA mini-app explains why the trans_ev_tridi_to_band subroutine is selected, most of the application's time is spent in DGEMM, so a complete exclusion of the BLAS library seems questionable. Especially since BLAS is a widely used library.

Furthermore, it is not explained for any of the four applications why the authors chose particular configurations for tracing and to what extent these are representative of the actual applications. I would encourage the authors to further explain their methodology.

Although most modern scientific applications use GPUs, the four mini-apps presented focus on purely CPU-based codes. Especially with regard to the aforementioned co-design process, it would be beneficial if the NOMAD suite also integrated GPU-accelerated applications.

Finally, I would like to point out to the authors that it would be helpful for readers to include a small example of the applicability of the NOMAD mini-app suite. This example could be used to briefly explain what insights the mini-apps can provide (e.g., performance metrics and provided output/results) and how these can then be used in the context of co-design.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Parallel file and storage systems, reproducible benchmarking, holistic performance engineering (i.e., system monitoring, performance modeling, performance optimization), modular supercomputing (i.e., resource disaggregation and virtualization), high performance computing and networking, parallel I/O, and parallel programming models

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Author Response 07 May 2024

Julio Gutierrez Moreno

This article introduces the NOMAD mini-applications suite, which is based on kernels from

ab initio electronic code structures participating in the NOMAD Center of Excellence. The NOMAD mini-apps suite focuses on four flagship applications, i.e., the ELPA eigensolver library and the GW implementations of the exciting, Abinit, and FHI-aims codes. The authors identify the most relevant kernels of those four applications by profiling runs with different configurations on the MareNostrum-4 supercomputer. The objective of the benchmark suite is to facilitate and enable co-design of future High-Performance Computing (HPC) systems with a focus on ab initio computational materials science codes. Mini-applications serve as crucial benchmarks in HPC systems due to their ability to simulate specific aspects of real-world applications efficiently. These compact programs encapsulate key computational patterns and performance characteristics, allowing for targeted analysis and optimization. By focusing on critical computational kernels or algorithms, mini-applications provide insights into system behavior, scalability, and bottlenecks, aiding in the evaluation and comparison of HPC architectures and software stacks. Moreover, their simplicity facilitates rapid experimentation and tuning, enabling researchers and engineers to fine-tune system configurations for optimal performance, ultimately advancing the design and deployment of HPC solutions. Overall, this article addresses a timely topic and describes the four application kernels well. The article has a clear structure and is mostly well written. Nevertheless, some questions remain unanswered after reading the paper and I would like to encourage the authors to clarify these points in a revision, specifically regarding the co-design aspect and applicability to large-scale systems.

Comments and suggestions: Comment #1: In the introduction, the authors state that “mini-app benchmarking can inform the implementation of new HPC systems more effectively”. Furthermore, the authors say that “Mini-apps have the potential to support our efforts in co-design as the performance metrics will be easily transferable between selected HPC systems and pre/post-exascale prototypes more effectively”. While these statements make sense, there is no further clarification on how the presented mini-apps can benefit a co-design cycle. Also, the authors do not further elaborate about selected or targeted performance metrics, even though they mention the (simplified) Sustained System Performance ((S)SSP). To further emphasize the impact of mini-app suites such as the NOMAD suite, it may be helpful to further explain performance metrics for the broader audience.
Reply: We thank the reviewer for her insightful feedback on our paper. We appreciate the comments regarding the impact of our mini-app suites in the co-design process and agree with the need for further clarification in some of the sections. Some examples of how these mini-apps are currently used for co-design are now given in the conclusions section and highlighted in the updated Figure 1. These include the continuous benchmarking of several peta- and pre-exascale architectures in (pre)operational stages and the porting to prototype hardware that will be potentially used in future exa- and post-exascale platforms. Our mini-apps could also be helpful as a starting point for implementing new porting strategies and optimisations before merging them with the full code. The codes included in the suite use different ab initio approaches, typically characterised by different bottlenecks. The new release of the mini-apps, published along with the revised version of the manuscript, contains benchmark scripts that launch multiple executions, progressively increasing the number of independent (single-core or threaded) processes until they fill all the CPUs in a node. These scripts offer options for compilation and execution on MareNostrum-5, MareNostrum-4, and CTE-POWER (BSC, Spain), Leonardo (CINECA, Italy), LUMI (CSC,

Finland), Vega (IZUM, Slovenia) and Karolina (IT4I, Czech Republic). The user can compare how the different mini-apps perform on different machines, analyse the potential degradation of the single-core performance with the occupancy of the node, or test different compiling options simply by looking at the timestamps that are printed as outputs for all the different setups. This upgrade has been commented on in the operation section.

Comment #2: Although the authors provide a good description of their kernel selection process based on the traces, it is not clear how representative these are for the overall performance of the application. For example, although the ELPA mini-app explains why the trans_ev_tridi_to_band subroutine is selected, most of the application's time is spent in DGEMM, so a complete exclusion of the BLAS library seems questionable. Especially since BLAS is a widely used library.
Reply: We thank these comments on the mini-app selection, which are very much in line with some of the questions raised by reviewer #1. We agree that our performance analyses could have needed to have been sufficiently explained in some of the cases. In the specific case of ELPA, the original version showed routines at a lower level than the ones that were presented in the manuscript. We understand that the relevance of the routines included in the mini-app could not be straightforwardly understood and, therefore, tridi_to_band (abbreviation for trans_ev_tridi_to_band) was represented mainly by a child subroutine called compute_hh_trafo. In the new version, the figure has been updated, and the trans_ev_tridi_to_band routine is shown without the compute_hh_trafo child subroutine, which allows for a clear distinction of the relevance of this subroutine within the mini-app. This routine takes 32% of the total time in the small test case (8K-element squared matrix) and 39% in the large (100K) one. In addition to the computational time, more features were pointed to in selecting this mini-app, such as the independence of external functions and the effort the ELPA developers made to adapt this kernel to use efficiently vectorial instruction on different hardware. While measuring the performance of linear algebra functions, such as the ones in BLAS, is attractive for benchmarking purposes, the room for co-design is much more limited than with our selection of kernels.

Comment #3: Furthermore, it is not explained for any of the four applications why the authors chose particular configurations for tracing and to what extent these are representative of the actual applications. I would encourage the authors to further explain their methodology. **Reply:** All our test cases represent realistic executions of technologically relevant materials, and they have been decided upon and discussed with the developers of each of the codes. Based on that, for all the applications, we selected a common case performed on a ZrO_2 unit cell and a more complex one, which required more computational resources and was used to compare the relative weight of each routine in different situations. For exciting, we used examples with reduced and all-bands scenarios. For Abinit, we used a small ZrO_2 and a more extensive $Zr_2Y_2O_7$ system. Last, for FHI-aims, we found that ZrO_2 was already quite heavy computationally, so we also sampled pure Si to perform our comparison. Although all these tests were already made in the past, they were not extensively described in the initial manuscript, so the updated version has been accordingly revised in these lines.

Comment #4: Although most modern scientific applications use GPUs, the four mini-apps presented focus on purely CPU-based codes. Especially with regard to the aforementioned

co-design process, it would be beneficial if the NOMAD suite also integrated GPU-accelerated applications.
Reply: We completely understand this view, which was also commented by the other referees. As noted here, most of the largest HPC platforms nowadays are in fact accelerated by GPUs. Several of the ab initio codes represented by these mini-apps have offloaded to GPUs selected parts of their DFT sections, however, these new implementations still need to tackle the GW implementations. We believe that with the increasing computational power that will come along with the arrival of the exascale era, the computationally demanding beyond-DFT methods such as GW will increase their accessibility and popularity among the community, and this is the main reason that drove us to choose GW for the majority of the selection of these mini-apps. It is also important to point out that in addition to the current predominance of GPUs, the future of HPC will also increase in hardware heterogeneity (some of them based on CPUs, which could, for example, operate larger vectors than AVX512, e.g. RISC-V VPUs) and new programming models will arise. In addition to the mentioned benchmarks, our mini-apps could also be helpful as a starting point for implementing new porting strategies and optimisations before merging them with the complete code. A mention of this future scenario has been added to the conclusions.

Comment #5: Finally, I would like to point out to the authors that it would be helpful for readers to include a small example of the applicability of the NOMAD mini-app suite. This example could be used to briefly explain what insights the mini-apps can provide (e.g., performance metrics and provided output/results) and how these can then be used in the context of co-design.
Reply: We thank the referee again for her comments. The NOMAD mini-apps suite is being used to continuously benchmark some of the supercomputers that are being currently deployed by the European High-Performance Computing Joint Undertaking (EuroHPC JU), all having different hardware and compilers/software stacks. Moreover, we have started activities on porting and testing these mini-apps to prototype architectures that will be used in future systems. The outcomes of these activities provide useful feedback to integrators, system administrators, that will eventually benefit all users. The tests carried out on experimental platforms are only feasible with mini-apps, and they are clear examples of co-design, in which the software is adapted to perform efficiently in these novel architectures while we provide constant feedback to the hardware architects and compiler developers at the early stage of development. Comments on these lines have also been added to the conclusions section of the manuscript. We should also mention that along with the updated version of the manuscript, we have also released the new 1.1 version of the suite, accessible at the same repository. In this version, we provide submission scripts to run our benchmarks on several EuroHPC machines: MareNostrum-4, MareNostrum-5, CTE-Power, Leonardo, LUMI, Karolina and Vega. Therefore, the execution of the suite would be straightforward to a user with access to any of these machines. Indications on this have been added to the operations section.

Competing Interests: No competing interests were disclosed.

Reviewer Report 24 April 2024

https://doi.org/10.21956/openreseurope.18286.r38382

© **2024 Proficz J.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Jerzy Proficz
[1] Politechnika Gdanska, Gdańsk, Pomeranian Voivodeship, Poland
[2] Politechnika Gdanska, Gdańsk, Pomeranian Voivodeship, Poland

The NOMAD mini-apps consist of a small collection of benchmarks focusing on ab initio electronic calculations, intended for a broad range of HPC systems. The concept of co-design is outlined, alongside examples of typical usage scenarios. The fundamental theory underlying the solutions is discussed. While I find the software interesting, I believe further elaboration on the following points would enhance its utility:

1. It would be beneficial to include a figure or table delineating the flow from typical usage/problem description to the specific mini-applications.
2. I am eager to observe results from running the benchmarks on an actual HPC system, particularly a comparative analysis between CPS and GPU architectures.

Overall, the NOMAD mini-apps paper presents a promising approach to ab initio electronic calculations, offering a comprehensive overview of co-design principles and typical usage scenarios, while also demonstrating potential for further enhancement through concrete benchmarking on real HPC systems.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: HPC, energy efficiency, software/hardware optimization, distributed/parallel applications

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 26 Apr 2024

Julio Gutierrez Moreno

We would like to thank the journal's editorial team and reviewers for their time, comments, and suggestions, which we believe have helped us improve our manuscript. The revision was made according to the referee's requirements, and we answered all points made. Please see the responses below.

Comment #1: The NOMAD mini-apps consist of a small collection of benchmarks focusing on ab initio electronic calculations, intended for a broad range of HPC systems. The concept of co-design is outlined, alongside examples of typical usage scenarios. The fundamental theory underlying the solutions is discussed.
Reply: We thank the comments provided by the referee and his remarks on the relevance of our mini-apps as a benchmark tool and co-design vehicle for HPC.

Comment #2: While I find the software interesting, I believe further elaboration on the following points would enhance its utility: It would be beneficial to include a figure or table delineating the flow from typical usage/problem description to the specific mini-applications. **Reply:** We thank the reviewer for pointing out the aspects of the potential usage of these mini-apps, which may need to be explained in more detail. All our test cases represent realistic executions of technologically relevant materials, and they have been decided upon and discussed with the developers of each of the codes. In addition, the codes included in the suite use different approaches and basis sets, including a wide range of the most widely used ab initio methods (plane waves and localised orbitals, with all-electron and pseudopotentials) and represent different bottlenecks. These include continuously benchmarking several pre-exascale architectures in (pre)operational stages and porting to prototype hardware potentially used in future (post)-exascale platforms. Moreover, our mini-apps could also be helpful as a starting point for implementing new porting strategies and optimisations before merging them with the complete code. These aspects are now mentioned in the conclusions of the updated manuscript. Also, Figure 1 and its caption have also been updated to comment on these concrete usages.

Comment #3: I am eager to observe results from running the benchmarks on an actual HPC system, particularly a comparative analysis between CPS and GPU architectures.
Reply: We understand the point raised here by the referee. As he noted, most of the largest HPC platforms nowadays are accelerated by GPUs. However, we should mention that while some of the ab initio codes represented by these mini-apps have some portions ported to GPUs, these new implementations still need to tackle the GW parts. More details on the codes and computing methodology used in the suite are given in our previous response, comment #5 from referee #1. The NOMAD suite is being used to continuously benchmark some of the supercomputers that are being currently deployed by the European High-Performance Computing Joint Undertaking (EuroHPC JU) and prototype architectures that will be used in future systems, all having different hardware and compilers/software stacks. While the outcomes of these activities provide useful feedback to integrators, system administrators, and users, some of these machines are in the preliminary development stages, and some of the benchmarks we have carried out cannot be publicly disclosed. Also, considering that many of these HPC systems are still being fine-tuned, the outcomes from

the benchmarks can vary depending on the exact time the picture is taken. Nevertheless, all these are clear examples of co-design, in which the software is adapted to perform efficiently in these novel architectures while we provide constant feedback to the hardware architects and compiler developers at the early stage of development. Comments on these lines have also been added to the conclusions section of the manuscript. Along with the updated version of the manuscript, we have also released the new 1.1 version of the suite, accessible at the same repository. In this version, we provide submission scripts to run our benchmarks on several EuroHPC machines: MareNostrum-4, MareNostrum-5, CTE-Power, Leonardo, LUMI, Karolina and Vega. Therefore, for a user with access to any of these machines, the compilation (with different compilers) and the execution of the suite should be straightforward. Of course, these scripts can also be adapted to new environments. Indications on this have been added to the operations section.

Comment #4: Overall, the NOMAD mini-apps paper presents a promising approach to ab initio electronic calculations, offering a comprehensive overview of co-design principles and typical usage scenarios, while also demonstrating potential for further enhancement through concrete benchmarking on real HPC systems.
Reply: We thank you again for the comments and further suggestions for improving our manuscript.

Competing Interests: No competing interests were disclosed.

Reviewer Report 18 March 2024

https://doi.org/10.21956/openreseurope.18286.r38384

© 2024 Dufrechou E. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ernesto Dufrechou
[1] Universidad de la Republica Uruguay, Montevideo, Montevideo Department, Uruguay
[2] Universidad de la Republica Uruguay, Montevideo, Montevideo Department, Uruguay

The article discusses a software tool designed to assess the performance of High-Performance Computing (HPC) platforms. This tool utilizes small mini-apps to capture the most characteristic computations of larger scientific software applications, particularly excerpts from applications of ab initio quantum chemistry methods. The purpose of this tool is to enable HPC hardware designers to quickly estimate the performance of larger applications on their designs by running lightweight mini-apps.

Molecular dynamics problems and quantum chemistry are critical applications that motivate the development of HPC hardware capable of running these applications efficiently. However, utilizing large simulations to evaluate a prototype hardware platform can be time and resource-consuming, making it impractical. The mini-app suite is designed to address this issue.

While the description of the software tool is clear, there is room for improvement in certain areas. The four mini-apps that form the main components of the tool contain selected kernels of the respective larger applications. In some cases, it may be difficult to match the mini-app kernel with the traces. For instance, the ELPA mini-app's chosen kernel is trans_ev_tridi_to_band, but the trace reveals the function tridi_to_band that accounts for only 4% of the execution time. Furthermore, the trace's different green colors make it challenging to identify the portions of the trace corresponding to that function. Although the authors provide arguments for this choice, it remains unclear whether this function's performance characterizes the performance of the entire app. Therefore, choosing a set of these functions may be more appropriate. Additionally, the BLAS (Basic Linear Algebra Subprograms) performance is critical in ELPA's case.
In general, the article would benefit from a deeper explanation of how the performance of the mini-apps relates to the performance of the larger applications, and under what assumptions the suite can be used in a co-design workflow.

Another area that requires further clarification is the criteria for the trace configuration. Parameters such as the block size and the number of threads and processes are provided, but their explanations are absent. Moreover, the relative weight of each function may vary depending on the problem size. Unfortunately, Abinit and FHI-aims were tested for only one problem size (35 q-points for Abinit and omitted the case of FHI-aims), while the others were tested for two problem sizes. Conducting systematic tests with more problem sizes and processes/threads configurations (a scaling analysis) could lead to an estimation of the apps' scaling on a new system using the mini-apps.

Finally, the mini-app suite could benefit from the inclusion of GPU-accelerated codes. This technology is ubiquitous in modern supercomputers and could enhance the tool's versatility and applicability.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: I research in heterogeneous computing and HPC. My main lines of work revolve around the accelleration of linear algebra kernels and scientific applications using GPUs.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Author Response 11 Apr 2024

Julio Gutierrez Moreno

We would like to thank the journal's editorial team and reviewer for his time, comments and suggestions, which we believe have helped me to improve our manuscript. The revision was made according to the referee's requirements and we answered (in black) all points made. Please see the responses below.

Comment #1: The article discusses a software tool designed to assess the performance of High-Performance Computing (HPC) platforms. This tool utilizes small mini-apps to capture the most characteristic computations of larger scientific software applications, particularly excerpts from applications of ab initio quantum chemistry methods. The purpose of this tool is to enable HPC hardware designers to quickly estimate the performance of larger applications on their designs by running lightweight mini-apps. Molecular dynamics problems and quantum chemistry are critical applications that motivate the development of HPC hardware capable of running these applications efficiently. However, utilizing large simulations to evaluate a prototype hardware platform can be time and resource-consuming, making it impractical. The mini-app suite is designed to address this issue.
Reply: We thank the reviewer for the suggestions for improving our manuscript. We appreciate the recognition of the relevance of ab initio electronics structure calculations in HPC and the importance of having mini-apps to test these codes on prototype hardware platforms.

Comment #2: While the description of the software tool is clear, there is room for improvement in certain areas. The four mini-apps that form the main components of the tool contain selected kernels of the respective larger applications. In some cases, it may be difficult to match the mini-app kernel with the traces. For instance, the ELPA mini-app's chosen kernel is trans_ev_tridi_to_band, but the trace reveals the function tridi_to_band that accounts for only 4% of the execution time. Furthermore, the trace's different green colors make it challenging to identify the portions of the trace corresponding to that function. Although the authors provide arguments for this choice, it remains unclear whether this function's performance characterizes the performance of the entire app. Therefore, choosing a set of these functions may be more appropriate. Additionally, the BLAS (Basic Linear Algebra Subprograms) performance is critical in ELPA's case.
Reply: We thank the reviewers for the comments provided, and we agree that our performance analyses could have needed to have been sufficiently explained in some of the cases. In the specific case of ELPA, the figure in the previous version showed routines at a lower level than the ones that were presented in the manuscript. We understand that the relevance of the routines included in the mini-app could not be straightforwardly understood and, therefore, *tridi_to_band (abbreviation for trans_ev_tridi_to_band)* was represented mainly by a child subroutine called compute_hh_trafo. In the new version, the figure has been updated and the *trans_ev_tridi_to_band* routine is shown without

compute_hh_trafo child subroutine, which allows for a clear distinction of the relevance of this subroutine within the mini-app. This routine takes 32% of the total time in the small test case and 39% in the large one. In addition to the computational time, more features were pointed to in selecting this mini-app, such as the independence of external functions and the effort the ELPA developers made to adapt this kernel to use vectorial instruction on different hardware efficiently. While measuring the performance of linear algebra functions, such as the ones in BLAS, is attractive for benchmarking purposes, the room for co-design is much more limited than with our selection of kernels. We agree that the colour-coding use in the traces could have been clearer; therefore, we have updated the colours in Figures 3 and 4.

Comment #3: In general, the article would benefit from a deeper explanation of how the performance of the mini-apps relates to the performance of the larger applications, and under what assumptions the suite can be used in a co-design workflow.
Reply: The manuscript has been revised with a description of the test cases used for each code. Moreover, a short description of their performance compared to the test case selected for the mini-app has been included. More details are given in the next comment. The manuscript has been accordingly revised to include this. Some examples of how these mini-apps are currently used for co-design are also given in the conclusions section. These include the continuous benchmarking of several peta- and pre-exascale architectures in (pre)operational stages and the porting to prototype hardware that will be potentially used in future exa- and post-exascale platforms. Comments mentioning these uses have been added to the conclusions section.

Comment #4: Another area that requires further clarification is the criteria for the trace configuration. Parameters such as the block size and the number of threads and processes are provided, but their explanations are absent. Moreover, the relative weight of each function may vary depending on the problem size. Unfortunately, Abinit and FHI-aims were tested for only one problem size (35 q-points for Abinit and omitted the case of FHI-aims), while the others were tested for two problem sizes. Conducting systematic tests with more problem sizes and processes/threads configurations (a scaling analysis) could lead to an estimation of the apps' scaling on a new system using the mini-apps.
Reply: We should clarify that we have performed exhaustive performance analyses using different test cases for each of the applications and performed scaling tests on the number of MPI/OMP processes to accommodate the available memory resources within a reasonable time to solution. All our test cases represent realistic executions of technologically relevant materials, and they have been decided upon and discussed with the developers of each of the codes. Based on that, for all the applications, we selected a common case performed on a ZrO_2 unit cell and a more complex one, which required more computational resources and was used to compare the relative weight of each routine in different situations. More concretely, in the case of ELPA, we used squared matrices with sizes of 8K and 100K. For exciting, we used examples with reduced and all bands scenarios. For Abinit, we used a small ZrO_2 and a more extensive $Zr_2Y_2O_7$ system. Last, for FHI-aims, we found that ZrO_2 was already quite heavy computationally, so we also sampled pure Si to perform our comparison. Although all these tests were already made in the past, they were not extensively described in the initial manuscript, so the updated version has been accordingly revised in these lines.

Comment #5: Finally, the mini-app suite could benefit from the inclusion of GPU-accelerated codes. This technology is ubiquitous in modern supercomputers and could enhance the tool's versatility and applicability.
Reply: We understand the point raised here by the referee. As he noted, most of the largest HPC platforms nowadays are accelerated by GPUs. However, we should mention that, while some of the ab initio codes represented by these mini-apps have some portions ported to GPUs, these new implementations still need to tackle the GW parts. Here, we should recall that DFT has been the main workhorse of the ab initio electronic structure community; therefore, it is understandable that developers prioritise the porting of the DFT sections of the code to GPUs. However, the accuracy of DFT in estimating properties such as the band-gap may need to be improved for some applications, and more accurate (and computationally expensive) beyond-DFT methods such as GW are required. We strongly believe that with the increasing computational power that will come along with the arrival of the exascale era, these methods will increasingly become more accessible and popular among the community, and this is the main reason that drove us to choose GW for the majority of the selection of these mini-apps. It is also important to point out that in addition to the current predominance of GPUs, the future of HPC will also increase in hardware heterogeneity (some of them based on CPUs, which could operate larger vectors, e.g. VPUs) and programming models. Our mini-apps could also be helpful as a starting point for implementing new porting strategies and optimisations before merging them with the complete code. A mention of this future scenario, including part of the text in this response, has also been added to the conclusions.

Competing Interests: No competing interests were disclosed.