The Journal of Chemical Physics

Quantum ESPRESSO toward the exascale F

Paolo Giannozzi, Oscar Baseggio, Pietro Bonfà, Davide Brunato, Roberto Car, Ivan Carnimeo, Carlo Cavazzoni, Stefano de Gironcoli, Pietro Delugas, Fabrizio Ferrari Ruffino, Andrea Ferretti, Nicola Marzari, Iurii Timrov, Andrea Urru, and Stefano Baroni

COLLECTIONS

Paper published as part of the special topic on Electronic Structure Software

F This paper was selected as Featured

View Online   Export Citation   CrossMark

ARTICLES YOU MAY BE INTERESTED IN

WIEN2k: An APW+lo program for calculating the properties of solids
The Journal of Chemical Physics **152**, 074101 (2020); https://doi.org/10.1063/1.5143061

Recent developments in the general atomic and molecular electronic structure system
The Journal of Chemical Physics **152**, 154102 (2020); https://doi.org/10.1063/5.0005188

Electronic structure software
The Journal of Chemical Physics **153**, 070401 (2020); https://doi.org/10.1063/5.0023185
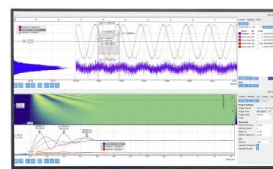
Quantum ESPRESSO toward the exascale

Paolo Giannozzi,[1,2,3,a] (iD) Oscar Baseggio,[4] (iD) Pietro Bonfà,[5,6] (iD) Davide Brunato,[4] Roberto Car,[7] (iD) Ivan Carnimeo,[4]
Carlo Cavazzoni,[3,8] (iD) Stefano de Gironcoli,[2,4] (iD) Pietro Delugas,[3,4] Fabrizio Ferrari Ruffino,[2] (iD)
Andrea Ferretti,[6] (iD) Nicola Marzari,[3,9] (iD) Iurii Timrov,[9] (iD) Andrea Urru,[4] (iD) and Stefano Baroni[2,3,4] (iD)

AFFILIATIONS

[1] Dipartimento di Scienze Matematiche, Informatiche e Fisiche, Università di Udine, via delle Scienze 206, I-33100 Udine, Italy, European Union

[2] CNR-IOM, Istituto dell'Officina dei Materiali, SISSA, I-34136 Trieste, Italy, European Union

[3] Quantum ESPRESSO Foundation, Cambridge Road Ind Estate, Milton, Cambridge CB24 6AZ, United Kingdom

[4] SISSA – Scuola Internazionale Superiore di Studi Avanzati, via Bonomea 265, I-34136 Trieste, Italy, European Union

[5] Dipartimento di Scienze Matematiche, Fisiche e Informatiche, Università di Parma, Parco Area delle Scienze 7/A, I-43124 Parma, Italy, European Union

[6] Centro S3, CNR-Istituto Nanoscienze, via Campi 213/A, I-41125 Modena, Italy, European Union

[7] Department of Chemistry, Princeton University, Princeton, New Jersey 08544, USA

[8] CINECA, Via Magnanelli 6/3, I 40033 Casalecchio di Reno, Bologna, Italy, European Union

[9] Theory and Simulation of Materials (THEOS), and National Centre for Computational Design and Discovery of Novel Materials (MARVEL), École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

Note: This article is part of the JCP Special Topic on Electronic Structure Software.
[a] Author to whom correspondence should be addressed: paolo.giannozzi@uniud.it

ABSTRACT

Quantum ESPRESSO is an open-source distribution of computer codes for quantum-mechanical materials modeling, based on density-functional theory, pseudopotentials, and plane waves, and renowned for its performance on a wide range of hardware architectures, from laptops to massively parallel computers, as well as for the breadth of its applications. In this paper, we present a motivation and brief review of the ongoing effort to port Quantum ESPRESSO onto heterogeneous architectures based on hardware accelerators, which will overcome the energy constraints that are currently hindering the way toward exascale computing.

Published under license by AIP Publishing. https://doi.org/10.1063/5.0005082

I. INTRODUCTION

The goal of this manuscript is to describe recent and ongoing work on the Quantum ESPRESSO software distribution for first-principles atomistic simulations. We focus, in particular, on the challenges posed by the new heterogeneous architectures, based on multi-core chips enhanced with multiple hardware "accelerators," coupling exceptional performances to an acceptable energy consumption. The large-scale adoption of these emerging architectures across different classes of computing systems will bring a paradigm shift into scientific computing, similar to what vector machines caused in the 1980s, parallel machines in the 1990s, and massively parallel machines more recently.

This paper is organized as follows:

In Sec. II, we briefly describe the history and the current status of Quantum ESPRESSO. We give a short overview of its features and capabilities by mostly referring to the relevant literature.

In Sec. III, we describe the challenges posed and opportunities offered by heterogeneous architectures. The opportunity is to reach what is dubbed "exascale computing": an unprecedented amount of computer power, opening new perspectives to computer simulations. The challenges, for scientific software, in general, and for Quantum ESPRESSO, in particular, are unprecedented as well. The amount of needed changes is much larger, and the effects of changes much deeper, than in previous transitions to vector, parallel, and massively parallel architectures. Moreover, several competing

architectures are hitting the marketplace, each one coming with a different software stack and tools.

Section III A describes ongoing work toward *performance portability*, that is, the ability to obtain comparably high performances on different computer architectures, minimizing the need for maintaining hardware-specific code versions. The work described in Section III A, which focuses on low-level libraries, would, however, be futile without more work at a higher level, aimed toward future maintainability of a large and complex scientific software project.

Section III B deals with recent and ongoing work aiming at a *sustainable development model*, that is, restructuring codes in a way that makes them easier to maintain, to extend, and especially to port to other architectures in a foreseeable future.

Section III C describes the current status, capabilities, and some benchmarks for QUANTUM ESPRESSO on NVIDIA graphics processing units (GPUs), one of the leading candidate architectures for the future "exascale computer." The benchmarks are designed for relatively small machines and do not aim at showing performances on large-sized systems. They aim instead at pointing out bottlenecks, inefficiencies, and the minimum size of calculations that saturate the computational power of a GPU.

Finally, Sec. IV contains our conclusions: a brief analysis of the achieved results and an outlook on forthcoming actions.

II. QUANTUM ESPRESSO AT THE TURN OF THE TWENTIES

The QUANTUM ESPRESSO project was started in 2002, with the merger of the following three packages for density-functional theory (DFT) simulations using plane waves and (ultrasoft) pseudopotentials, which had been under development since the mid-1980s:

- PWscf: code `pw.x` for self-consistent field (SCF) solution of Kohn–Sham equations and structural optimization, code `ph.x` for lattice-dynamical calculations using linear response, plus many other utility codes;
- CP: a code performing first-principles molecular dynamics (MD) simulations of the Car–Parrinello type, specialized to large supercells; and
- FPMD: a code similar to CP, but with a different and partially overlapping set of functionalities.

Innovation in quantum-mechanical materials modeling has always been one of the main concerns and distinctive features of this project. In particular, the linear-response code `ph.x` contained in PWscf was, to the best of the authors' knowledge, the first implementation[1] of density-functional perturbation theory (DFPT).[2–5] The elimination of virtual orbitals from linear-response and excited-state calculations, pioneered in this implementation, has remained one of the distinctive features of QUANTUM ESPRESSO, later to be adopted by several other codes. On the same line, CP and FPMD were both offsprings of the first implementation[6] of Car–Parrinello MD, now merged into a single code, `cp.x`.

Since the beginning, QUANTUM ESPRESSO was released under an open-source license (GNU GPL)[7] and was conceived as a "distribution": an integrated suite of packages, following loose programming guidelines, rather than a monolithic application. The rationale for these choices lies in the philosophy of fostering collaboration and sharing of code among scientists, creating a community of users and developers, while pursuing innovation in methods and algorithms. For more details on such aspects, we refer to Refs. 8 and 9.

The QUANTUM ESPRESSO distribution has much evolved over the years. On the one hand, more sophisticated theoretical methods and algorithmic advances have been implemented, in particular,

- projector-augmented waves (PAW);
- non-collinear and spin–orbit coupling (SOC) calculations;
- Hubbard-corrected functionals;
- nonlocal functionals and other semi-empirical and less-empirical corrections for van der Waals bonded systems; and
- hybrid functionals, also exploiting orbital localization.

On the other hand, QUANTUM ESPRESSO has extended its original scope with additional packages for more property calculations. We mention, in particular,

- activation energies and reaction barriers using the nudged-elastic-band (NEB) method;
- superconducting transition temperatures, electrical resistivity, and other effects related to electron–phonon interactions;
- phonon linewidths, thermal conductivity, and anharmonic effects;
- nuclear magnetic resonance (NMR) chemical shifts and electron paramagnetic resonance (EPR) g-tensors;
- transport in nano-wires in the ballistic regime;
- near-edge x-ray absorption spectra;
- implicit solvation and electrochemical models; and
- optical and electron energy loss spectra using time-dependent density-functional perturbation theory.

On the purely computational side, optimization for modern and massively parallel high-performance computing (HPC) architectures has been pursued using multiple parallelization levels involving both message-passing interface (MPI) and multi-threading (OpenMP).

All these advances, and related references, are documented in Refs. 8 and 9. In particular, Ref. 9 also documented a preliminary version of package HP, performing linear-response calculations of Hubbard parameters, which has been since released.[10]

In addition to being directly used for implemented property calculations, QUANTUM ESPRESSO is also used in more creative ways as follows:

- as a "quantum engine" for more sophisticated calculations, such as advanced MD calculations implemented in `i-Pi`,[11] genetic and evolutionary algorithms, and high-throughput calculations with `AiiDA`;[12] and
- in modified or patched versions, for specific purposes, as in "embedded" `eQE`.[13]

Finally, QUANTUM ESPRESSO is used to produce DFT data for further processing by other codes. We mention, in particular, codes performing Quantum Monte Carlo calculations (`QMCPack`[14]) and many-body perturbation theory (MBPT) (`Yambo`,[15] `BerkeleyGW`,[16] `WEST`,[17] and `Sternheimer-GW`[18]), determining maximally localized Wannier functions [`Wannier90`[19] and electron–phonon interactions (`EPW`[20])], and codes performing different types of data analyses (e.g., topological analysis with `critic2`[21] or computing transport properties (e.g., `WanT`[22] and `KGEC`[23]).

Stable versions of the source code—the latest is v.6.5 at the time this article was written—as well as development and experimental versions can be downloaded from http://www.quantum-espresso.org/download.

III. CHALLENGES AND OPPORTUNITIES OF NEW HETEROGENEOUS ARCHITECTURES

The push of quantum mechanical materials simulation software toward ever increasing levels of complexity, accuracy, and performance has been so far assisted by a constant downsizing of micro-processing units, allowing for a steady increase in the compute capacity of general-purpose architectures at constant power. This process is now hitting its ultimate physical limits and is just about to come to an end. To reverse this state of affairs, major and disruptive changes in hardware architecture are to occur. The constraints set by power consumption can only be met by heterogeneous architectures with specialized cores ("accelerators") that maximize efficiency for a small set of instructions, e.g., graphics processing units (GPUs), PEZY chips, tensor processors, and neuromorphic chips. A quick look at the first places of the Top500 supercomputers list clearly shows that heterogeneous architectures have become the *de facto* standard for new generation HPC systems.[24,25] On the single node scale, the ratio between the computational power provided by accelerators and traditional central processing units (CPUs) is found to grow at each procurement cycle over the last 10 years.[25] Most of the computational power of future "exascale" machines (that is, capable of 1 exaflop or 10^{18} floating-point operations per second) will come from accelerators.

The ensuing architectural complexity will set demanding requirements in terms of data movement, heterogeneous memory management, and fault tolerance, which will all require a major, possibly joint, re-design of circuits and algorithms and the adoption of different programming paradigms. In particular, extremely parallel applications will require rapid and substantial architectural shifts, including the handling of intra-node data movement between disjoint memory spaces and the explicit treatment of deeper memory hierarchies.

Porting community codes to novel hardware architectures has always required extensive re-coding in the past. This cannot be sustained any longer in view of the considerable complexity reached by QUANTUM ESPRESSO and similar community codes (several hundred thousands code lines each) and the forthcoming diversity, heterogeneity, and rapid evolution of the hardware architectures. The solution we have identified is to refactor QUANTUM ESPRESSO into multiple layers, resulting from the assembly of weakly coupled components (modules and libraries), which are to be maintained and enhanced independently from each other, shared among different codes, and designed to be as architecture-agnostic as possible. A bird's eye view of the code will reveal four main such layers (see Fig. 1):

- The top layer is a collection of *property calculators* to compute materials properties and to simulate processes, which is the ultimate goal of molecular and materials simulations. These property calculators may share global variables among themselves and with the quantum-engine modules (see below). This layer could be partially and progressively
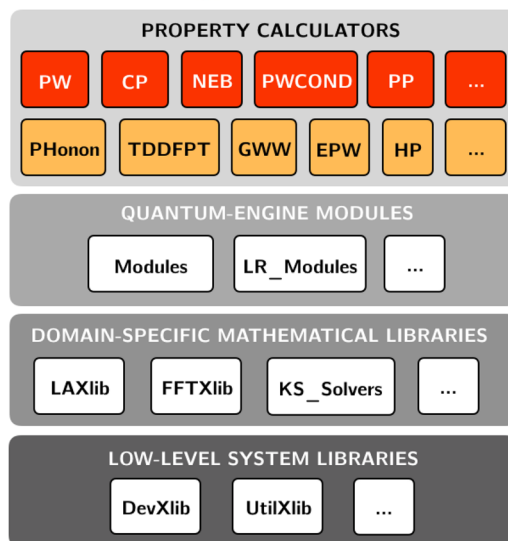
FIG. 1. Schematic illustration of the structure of the QUANTUM ESPRESSO distribution. The non-perturbative codes for SCF, MD, postprocessing, and other calculations are those in the first line of the "property calculators" section (highlighted in red), while the linear-response and MBPT codes are in the second line (highlighted in yellow). Low-level layers of the QUANTUM ESPRESSO distribution (as discussed in the text) are also shown.

made code-agnostic, while it should remain as architecture-agnostic as possible.

- The core layer contains *quantum-engine modules* that solve the one-particle Schrödinger equation and perform associated tasks, such as Hamiltonian builds (i.e., the application of the one-particle Hamiltonian to molecular orbitals and Bloch states), or other tasks related to density-functional, linear-response, and many-body perturbation theories. This layer is and will likely continue to be code-specific but should stay as architecture-agnostic as possible. Ideally, the same quantum-engine modules could be shared among different codes performing similar tasks, e.g., standard SCF computations and geometry optimizations, *ab initio* MD, and linear-response/MBPT calculations.

- A collection of *domain-specific mathematical libraries* are used to perform general-purpose tasks, such as 3D Fourier analysis and linear algebra, using both iterative and factorization algorithms, non-linear optimization, etc. Such mathematical libraries should be easily shared among different codes of a same class (e.g., adopting the same quantum-mechanical representation/basis set) and stay largely architecture-agnostic as above. While these libraries may themselves contain modules in order to ensure easy portability across different codes, the data exchange between them and the calling code will only occur via standard public application programming interfaces (APIs), which will make

minimal use of complex data structures and the derived data types.

- Finally, a collection of *low-level system libraries* are used to abstract the most recurrent architecture-specific constructs, such as data offload to/from specialized hardware, memory synchronization, and the like. These libraries will interface directly with the underlying hardware architectures and should be fine-tuned on each of them for optimal performance. They should be architecture-specific, and as much code-agnostic as possible, without interfering with the other layers.

This strategy is meant to accomplish what has come to be dubbed *separation of concerns*: ideally, method developers in science departments and research labs should be concerned with property calculators, disregarding architectural details, whereas scientists and research engineers in IT departments and HPC centers should focus on low-level mathematical and system libraries. The two categories of developers should work shoulder-to-shoulder to develop and maintain the quantum engine. Separation of concerns is the over-arching guideline for the action of the EU MAX Centre of Excellence (CoE) for HPC Applications,[26] whose mission is to foster the porting of important community codes for quantum materials modeling to heterogeneous architectures.

The implementation of the above strategy is bound to entail an extensive refactoring of community codes, which, in turn, will require a longer time than allowed by the pressing advance of heterogeneous architectures. For this reason, the course of action undertaken by the QUANTUM ESPRESSO developers follows two converging tracks. On the one hand, an accelerated version of `pw.x` working on NVIDIA GPUs is already available, and more components—initially `cp.x`, followed by `ph.x` and other linear-response codes—are being ported. On the other hand, the refactoring of the whole distribution into a collection of stand-alone modules and libraries is steadily progressing. The final goal to which both tracks will eventually converge is the identification and gradual encapsulation of architecture-specific computational kernels into low-level libraries, so as to make the migration to multiple architectures as seamless as possible. We aim, in particular, at using the forthcoming OpenMP-5 standard, which in addition to architecture-agnostic APIs (already in OpenMP 4.5) provides deep memory copies and a simple framework for exploiting unified shared memory when available.

In the following, we discuss in some detail the different logical layers that constitute the structure of the QUANTUM ESPRESSO distribution, as shown in Fig. 1, starting from the bottom layer. Low-level system libraries are discussed in Sec. III A, while the remaining layers are discussed in Sec. III B.

A. Performance portability

Performance portability across current and future heterogeneous architectures is one of the grand challenges in the design of HPC applications. General-purpose frameworks have been proposed,[27–30] but none of them has reached maturity and widespread adoption. In addition, Fortran support is still very limited or missing entirely. In this context, the MAX CoE is promoting and coordinating a collective effort involving the developers of various materials

modeling applications. Taking on this challenge with a domain-specific approach has the advantage of providing abstraction and encapsulation of a limited number of functionalities that constitute the building blocks of the most common operations performed on the accelerators in this field. This will allow us to prepare low-level architecture-specific implementations of a limited number of kernels that have been already characterized and isolated, thus keeping the source code of the various scientific applications untouched and reducing code branches when new systems will appear in the market.

Such an effort is still in the early stages but is under active development and is progressively entering the GPU port of QUANTUM ESPRESSO through the so-called `DevXlib` library. This library started off as a common initiative shared among MAX CoE codes (notably, QUANTUM ESPRESSO and `Yambo`), aimed at hiding CUDA Fortran extensions (see Sec. III C) in the main source base. Being used by different codes, the library has been rationalized and further abstracted, thus becoming a performance portability tool aimed at supporting multiple back-ends (support to OpenACC and OpenMP-5 foreseen, direct extension to CUDA C possible). The main features included in the library by design are the following:

- performance portability for Fortran codes,
- deal with multiple hardware and software stacks, programming models, and missing standards,
- wrap/encapsulate device specific code, and
- focused on limiting code disruption (to foster community support).

It is important to note that part of the library design includes the definition of which device-related abstract concepts need to be exposed to the scientific developers. To give an example, memory copy and synchronization to/from host/device memory are abstract operations that the developers of property calculators or of the quantum engine itself may need to control directly. Therefore, `DevXlib` exposes such control in the form of library APIs that are agnostic of the specific device back-end.

In practice, `DevXlib` provides the user with (i) interfaces to *memory handling* operations including creation and locking of memory buffers (`device_memcpy` and `device_buffers`); (ii) interfaces to *basic and dense-matrix linear algebra routines*, similarly to BLAS and Lapack (`device_linalg`); (iii) interfaces to more *domain-specific operations* (`device_auxfuncs`); and (iv) device-oriented *data structure* compatible with Fortran usage. In particular, memory handling allows the user to copy memory host-to-host, device-to-device, and also across memories, host-to-device and vice versa, thereby dealing also with memory off-load and synchronization. Importantly, both synchronous and asynchronous copies can be performed with explicit control. Moreover, the explicit handling of memory buffers is meant to ease or avoid the procedure of allocation and deallocation of auxiliary workspace.

Besides the interface to linear algebra optimized libraries, such as cuBLAS and cuSOLVER in the case of NVIDIA GPUs, `DevXlib` also provides interfaces to selected (more specialized) kernels that appear often in plane wave electronic structure codes (such as scalar products with **G**-vector remapping or specific matrix or vector updates, to name a few). While not covering all possible kernels of the same complexity, this is quite effective in helping to keep the code sources as clean as possible. Importantly, some efforts have also been directed to devise Fortran-compatible data structures

to handle the extra complexity of data stored on host and/or accelerator memories. In principle, these data structures need to be able to hide hardware complexity (e.g., being vendor agnostic for what concerns accelerator memory), to allow for seamless memory synchronization and transfer, and to be equipped with a set of methods to implement the most common data operations. Software engineering and experimentation of these data structures is currently ongoing.

In order to make the code more flexible and portable, some basic functionalities are accessed via common interfaces provided by a low-level library `UtilXlib`, including utilities for MPI and OpenMP parallelization, timing, error and memory handling. This library has been extended to include execution and data synchronization points for communications involving memory spaces located on the GPUs.

B. Toward a sustainable development, maintenance, and porting model

Quantum ESPRESSO has grown in size during the years, including as of v.6.5 almost 600 000 lines of Fortran code, 60 000 lines of C, python, shell scripts, plus a large amount of tests, examples, and documentation. While not especially large with respect to scientific software projects in other fields, Quantum ESPRESSO is sufficiently bulky to make its maintenance and extension a challenge that cannot be sustained without resorting to modern software engineering techniques. Much work, described in part in Sec. 3 of Ref. 9, has been done in the past along the following directions:

- provide a large set of automated tests to ensure the validity and portability of the results under different building and execution patterns;
- extend the interoperability with other codes via structured I/O, using an extended markup language (XML) with an industry-grade *schema* description for small human-readable, data files, and optionally a hierarchical format (HDF5) for large, binary, datasets; and
- collect large parts of the code base into modules and libraries in order to enhance its readability, ease of maintenance, and portability.

The work along the latter direction has been extended with the creation of distinct code layers: (i) *libraries* and (ii) *modules*. *Libraries* have a well-encapsulated inner data structure and exchange data with the rest of the code only through predefined APIs. As a consequence, libraries can be developed, distributed, and compiled independently of each other and then linked to the main code. *Modules* are reusable blocks of code whose functionalities are accessed via well-defined APIs as well but, mainly for reasons of computational efficiency, do not stick to such a clear-cut data encapsulation scheme as libraries and share a significant portion of their data structure with the rest of the code. For this reason, modules must be compiled together with the main code and are mainly intended for usage inside the Quantum ESPRESSO suite or other codes sharing the same global data structure.

1. Domain-specific mathematical libraries

Apart from improving the maintainability of the whole distribution, packaging the code base into libraries also has the advantage of making distributed development and maintenance easier and of

providing a wider community with effective tools for developing electronic-structure software ready to use in modern HPC infrastructures. For this reason, we aim at avoiding the usage of structured data type as arguments of the APIs as much as possible and at exposing the interfaces using included files rather than Fortran modules.

Currently, three major packages have been extracted from Quantum ESPRESSO and are ready to be distributed as stand-alone libraries, namely,

- `LAXlib`, performing parallel dense-matrix operations, including basic linear algebra and diagonalization;
- `FFTXlib`, for parallel distributed three-dimensional fast Fourier transforms; and
- `KS_Solvers`, a collection of iterative diagonalization algorithms to solve the Kohn–Sham equations.

A large part of the computations of a typical electronic-structure calculation is performed inside these libraries. The usage of machine-optimized mathematical libraries and the inclusion of further optimizations, either architecture-agnostic or architecture-specific, in these libraries will automatically profit to all codes and computations. It is at this level that the separation of concerns is most fruitful in terms of performance portability. While the original code targeted MPI and OpenMP parallelization on many CPUs, the extension to different programming paradigms for heterogeneous architectures has much progressed since, also thanks to contributions from IT experts.

`LAXlib` and `FFTXlib` libraries, with their completely encapsulated inner data structures, can be easily used by third parties. Their interfaces are transparent to the specific underlying architecture.

The iterative diagonalization algorithms collected inside the `KS_Solvers` are disentangled from the specific Hamiltonian builder, which is called by the library as an external routine; the definition of wavefunctions and their scalar products inside the Hamiltonian builder must be compatible with the one used inside `KS_Solvers`. For some of the algorithms, a Reverse Communication Interface (RCI) is also available, allowing one to directly pass the $H|\psi\rangle$ vectors to the library, leaving to the programmer the task of computing and converting them to the format expected by the RCI.

The goals of the activities described here largely overlap with those of the Electronic Structure Library (ESL) project at CECAM.[31] The possibility of decoupling the `KS_Solvers`, `LAXlib`, and `FFTXlib` libraries from their native codes was first demonstrated during a workshop organized in 2017 within the ESL initiative.[32] Moreover, both `KS_Solvers` and `LAXlib` may use another library maintained by ESL, ELPA (included in ELSI) for dense-matrix diagonalization. Quantum ESPRESSO may also use the `libxc` ESL library computing exchange-correlation functionals.

2. Quantum-engine modules

For other parts of the code, data encapsulation is difficult to achieve or even unfeasible, or may introduce inefficiencies. For those cases, it was chosen to refactor the code into general modules, still using the global data structure of the suite. These modules are currently meant to be used inside the distribution, but they are designed to be easily accessible for the development of further applications

built upon the QUANTUM ESPRESSO code base. Notable examples are `Modules` and `LR_Modules`.

`Modules` is a collection of Fortran modules and subroutines that implement various operations needed to solve self-consistently and iteratively the Kohn–Sham equations of DFT. In particular, Modules contains the following functionalities: (i) definition of global variables and universal constants, (ii) reading of input parameters and of pseudopotentials, (iii) definitions of Bravais and reciprocal lattices, (iv) symmetry analysis and symmetrization operations, (v) calculation of the exchange-correlation potential, (vi) generation of plane waves and of their mapping to FFT grids, (vii) generation of **k** points, and (viii) calculation of pseudopotential terms. Historically, `Modules` exists since the very first version of QUANTUM ESPRESSO, but it has continuously evolved in order to adapt to novel utilities and packages of the suite.

`LR_Modules` is a much more recent part of QUANTUM ESPRESSO, which appeared about 5 years ago and evolved significantly since that time. The reason for the creation of `LR_Modules` was to unify, harmonize, generalize, and refactor the functionalities that are common to all linear-response and MBPT codes of the suite. `LR_Modules` contains the following functionalities: (i) definition of global data structures for linear response, (ii) calculators of linear-response quantities (such as response density and potentials), (ii) iterative solvers (e.g., Lanczos recursive algorithms), (iii) response exchange-correlation kernel calculators, (iv) symmetrization routines, and (v) projectors on the empty-states manifold, to name a few. The functionalities of `LR_Modules` are used in the following packages:

- `PHonon` for calculation of lattice vibrational modes (phonons), Born effective charges, dielectric tensor, and other vibrational properties;[1–3,33–43]
- `TDDFPT` for calculation of optical absorption spectra of molecular systems[44–49] and collective excitations in solids such as plasmons[50–52] and magnons;[53]
- `EPW` for calculation of electron–phonon coupling, transport, and superconducting properties of materials;[20] and
- `HP` for the first-principles calculation of Hubbard parameters of the Hubbard-corrected DFT.[10]

The generalized and unified subroutines from `LR_Modules` have been refactored in such a way that they can be easily and straightforwardly employed in any other future linear-response or MBPT code of QUANTUM ESPRESSO or even in third-party codes. They can now be used generically to build perturbations, apply them to the occupied ground-state Kohn–Sham wavefunctions, and compute the related self-consistent first-order response properties either by solving the Sternheimer equations or by solving the Liouville quantum equations using the Lanczos recursion method.

3. Interoperability

Exportable output in QUANTUM ESPRESSO is based on the adoption of standard data formats: XML and HDF5. These two formats have the advantage of providing information about the hierarchy and the types of the data that may be automatically processed by external applications. The support of these features in modern scripting languages such as python makes them convenient for the development of postprocessing and analysis tools. For HDF5, the description of the hierarchy and of data types is contained in the

file; for XML files, we provide it under the form of XSD schemas.[54] In order to streamline the reading of XML files in python using the specifications of the schemas, we have also released a python package, `xmlschema`,[55] which converts the ASCII content of the XML file into a corresponding python dictionary whose structure and data types follow the indications given by the schema. The python dictionary can then be used directly by the reading application or saved as JSON of YAML files.

The coherence between the released schemas for XML and the effective output of the applications is guaranteed by a python set of tools that produce Fortran bindings for reading and writing XML data, starting from the XML schema. The tools generate the writing routines from the format specification and keep them automatically aligned with the schema. These tools have also been released as a separate package.[56]

The `qeschema`[57] package provides the APIs specific for reading and writing the XML, HDF5, and unified pseudopotential format (UPF) files used by the applications in the suite. This package also provides converters to map some structured data as crystal structures to and from other format commonly used in visualizers, structure databases, or atomistic simulation utilities as ASE[58] or pymatgen[59] or HDF5 charge densities that may be exported to other formats, for instance, the XSF format of XCrySDen.[60]

The `postQE`[61] package provides python extensions and APIs to use the postprocessing utilities of the PP package inside python applications. Many PP components are compiled as python extensions using F2PY. For those applications that may have a large computational load and memory footprint, `postQE` provides instead tools to extract data from the output files of the Fortran executables.

C. Evolution of the GPU-enabled version

QUANTUM ESPRESSO introduced support for accelerated systems as early as 2013, starting from v. 5.0, in the form of custom plugins for some of the codes of the suite.[62,63] This initial approach, based on CUDA C and *ad hoc* libraries for linear algebra,[62] proved to be successful in boosting the code performance,[64] but hardly sustainable from the maintenance and development points of view, mainly due to the substantial amount of "boilerplate" (replicated) code used to interface Fortran subroutines with CUDA C.

In light of this limitation, a new port, dubbed QE-GPU, has been recently rewritten from scratch, starting from the case study presented by Romero *et al.*[65] who ported v. 6.1 of `pw.x` to NVIDIA GPUs.[66] In Ref. 65, the authors detailed a new strategy based on CUDA Fortran—the Fortran analog of CUDA C—and demonstrated 2× to 3× speedup consistently achieved on a variety of platforms and using different benchmarks. The new GPU-enabled releases of QUANTUM ESPRESSO extend this work, but adopting a few design solutions to streamline future development and porting to other heterogeneous architectures, as detailed below.

While still being developed with CUDA Fortran, the last release of QE-GPU is almost entirely accelerated through a directive-based approach, using the so-called *cuf kernel* compiler directive, which generates a parallel architecture-specific code for loop-based structures. In some cases, this choice may limit the code performance, but it brings a number of positive consequences. First, it allows one to validate the GPU code on the CPU and, in principle, to retain a single source code for both CPU and GPU implementations.

Second, this design choice enables the straightforward adoption of other directive-based programming models, such as OpenACC or OpenMP, if required. As a consequence, even if the current implementation only runs on NVIDIA GPUs, special care has been paid to design the software in a way that minimizes the future effort needed to port the code to other accelerators that can be exploited through a directive-based programming model. In particular, attention has been paid not to introduce CUDA Fortran kernels in the high-level structure of the code, except for trivial data transformations, and to keep them confined to the domain-specific and system libraries described in Secs. II A and II B 1.

A further advantage of CUDA Fortran is explicit memory management. QUANTUM ESPRESSO organizes related data into modules and derived types. Allocation and synchronization of these kinds of data structures is a straightforward and concise operation in CUDA Fortran, while OpenACC and OpenMP support automatic synchronization of these data types only in the most recent releases. In addition, by retaining full control on memory allocations and synchronizations between the RAM and GPU's global memory, we provide a clear view of all intra-node memory operations to the programmer, thus facilitating future porting activities.

As detailed in Sec. III B, a few specific components of QUANTUM ESPRESSO have been extracted from the main code trunk, made independent from it and released as packaged libraries. Some of them have been targeted for GPU acceleration, namely, the parallel dense eigenvalue solver (LAXlib) and the parallel distributed FFT (FFTXlib). In this case, we abandoned directive based acceleration in favor of architecture specific APIs and libraries. Indeed, the last release of these libraries relies on cuBLAS for linear algebra kernels, cuFFT for 1D and 2D Fourier transforms, and cuSOLVER for solving real and complex generalized eigenvalue problems.[67]

As of version 6.5, the following set of functionalities benefit from GPU acceleration and operate on data residing on the GPU:

- electronic self-consistency for both magnetic and spin-unpolarized systems;
- iterative solution of the Kohn–Sham Hamiltonian using either the conjugate gradient or the Davidson method;
- calculation of atomic forces; and
- calculation of exact exchange terms for hybrid functionals.

The acceleration is obtained by exploiting numerous kernels that have been ported to GPU: local-potential, pseudopotential, kinetic-energy contributions to the Hamiltonian, preconditioner evaluation for iterative solvers, application of the overlap operator, wavefunction initialization, Hubbard component contribution to the effective potential, and charge density generation. For all the aforementioned operations, where applicable, both the case of real-valued wavefunctions ($\mathbf{k} = 0$ only sampling, useful to reduce the memory footprint and to speed up simulations) and the case with spinor wavefunctions (non-collinear magnetism) have been ported to GPU.

Currently, only pw.x can benefit from GPU acceleration, but other codes are being ported and will be available in future releases. The GPU-enabled version of pw.x is fully compatible with its CPU counterpart, provides the same features, undergoes the same regression testing suite, and converges to equivalent results within a given convergence criterion.

The speedup provided by the GPU implementation depends drastically both on the hardware and on the details of the input data. The extreme scale performance has been already detailed elsewhere;[68,69] thus, here we focus on smaller problems, consisting of tens of atoms and hundreds of electrons. This will allow us to identify the lower limit for the input cases that can benefit from GPU acceleration.

The qeinputgenerator[70] was used to prepare a set of benchmarks. This tool automatically generates input files for pw.x providing a few options to customize the accuracy of the simulation and using pseudopotentials from two sets, either standard solid-state pseudopotentials (SSSP) efficiency or SSSP precision.[71] These, in turn, include norm conserving,[72–74] ultrasoft,[75] and PAW[76,77] pseudopotentials, thus covering a significant fraction of pw.x functionalities. Sixteen structures having unit cell volume between 250 Å³ and 1000 Å³ were randomly selected from the Crystallography Open Database (COD).[78–83] Structures with fractional occupations and rare earth elements were discarded. All input and output data are available on the Materials Cloud archive.[84]

In Fig. 2, we compare the best time-to-solution obtained with a single 18-core Intel(R) Xeon(R) E5-2697 v4 @ 2.30 GHz (BDW) CPU and the same hardware accompanied by one NVIDIA's Tesla V100 GPU card. The ratio between the theoretical peak performance of these two units is roughly 1–10, but effective GPU acceleration can only be achieved with extremely data parallel workloads and the speedup provided by the graphic card can even become lower than 1 when this condition is not met. This limit is investigated with the present benchmark using a rather standard balance between GPU and CPU computational power for a HPC node.

The CPU results have been collected with QUANTUM ESPRESSO v. 6.5 compiled with Intel's 2019 suite, Intel's MPI implementation, and Intel's Math Kernel Library (MKL), while QE-GPU v. 6.5 was built with the PGI 19.10 Fortran compiler and linked to Intel's 2019 MKL. For the set of inputs detailed above, the pure MPI parallelization is the best strategy for the CPU version of the code; therefore, we performed all simulations for the CPU version disabling OpenMP. On the other hand, the GPU code requires OpenMP parallelism since the number of MPI processes is limited by the number of GPU cards installed on the system: each MPI process should be assigned just one accelerator.[85] For both the CPU and the GPU versions, only parallelism over $\mathbf{k}$-points (-NPOOL option) has been used—a reasonable choice, given the relatively small dimension of the dense eigenvalue problem to be solved during iterative diagonalization.

The two sets of results are numerically equivalent: the largest discrepancy between the total energies computed by the CPU and the GPU versions is $2 \cdot 10^{-8}$ Ry, while the largest difference in the total force is $3 \cdot 10^{-5}$ Ry/Bohr.

In Fig. 2, we report the wall time required to reach convergence (or 80 SCF cycles when convergence was not achieved) for the CPU and GPU versions. Only the best time to solution as a function of $\mathbf{k}$-point parallelism is reported. The total execution time also has contributions, not reported because much smaller, from the initialization step and by the final calculation of atomic forces. For the smallest test case, a 98-atom organic compound with 246 electrons that converges in less than 1 min on our hardware, the GPU speedup is just 1.4×. This is indeed the current limiting size for taking advantage of GPU acceleration. On the other hand, as the problem size grows, especially as a function of the basis set dimension,[86] the speedup can exceed 3×.
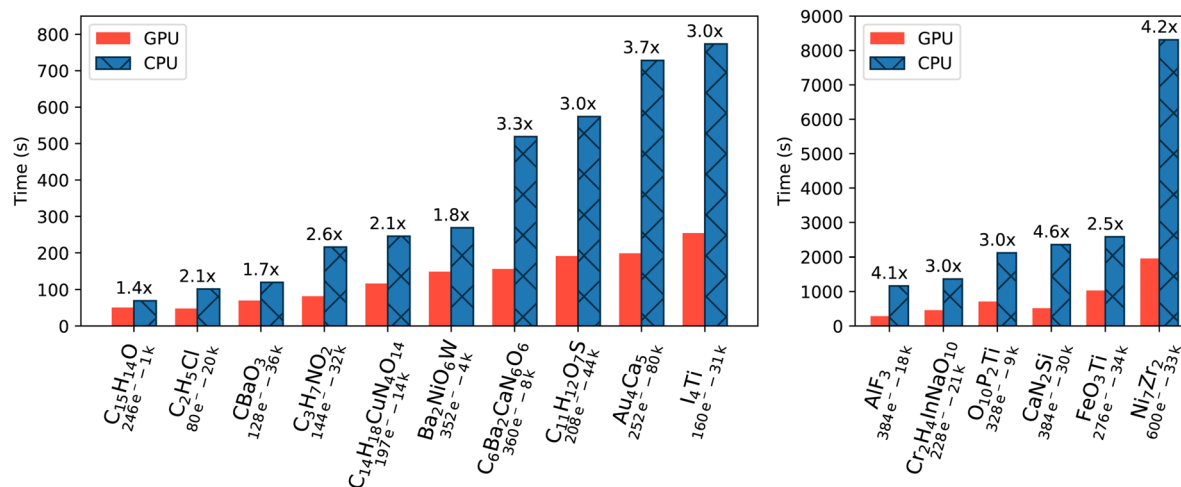
FIG. 2. Collection of benchmarks performed with QUANTUM ESPRESSO v. 6.5 and QE-GPU v. 6.5 on an 18-core Intel(R) Xeon(R) E5-2697 v4 @ 2.30 GHz or the same CPU plus a NVIDIA V100 card. The number of electrons per simulation cell and the number of **k**-points used for reciprocal space sampling are reported close to the chemical formula. For each simulation, the sum of the time taken by the initialization, the iterations during self-consistency, and the estimation of atomic forces is compared.

IV. OUTLOOK AND CONCLUSIONS

The work on the QUANTUM ESPRESSO distribution has since many years followed some well-established lines of action: (i) implementing new numerical algorithms, theoretical methods, and property calculations; (ii) improving interoperability with external codes; and (iii) making the codes more modular and easier to work with, more portable across architectures, without sacrificing performances. The arrival of heterogeneous, accelerated architectures has made the latter direction more urgent and wider in scope. It is no longer sufficient to isolate low-level computational kernels into simple mathematical libraries: performance portability must be ensured at the level of so-called "domain-specific" libraries. Without effective actions in the direction (iii), it will become difficult to implement, and even more difficult to maintain, further property calculations, new theories, and new algorithms.

The work described in Sec. III C is the starting point for the effective implementation of the plan described in Sec. III. The next significant step is the merger of the main QUANTUM ESPRESSO distribution with the CUDA Fortran version for NVIDIA GPUs. This will eliminate the constraint of keeping the latter version akin to the main distribution in order to simplify the "alignment" process as the main distribution evolves.

In parallel, more work is ongoing to achieve performance portability, as described in Sec. III A. In particular, support for OpenMP-5 is being introduced into the domain-specific mathematical libraries of Sec. III B and the low-level system libraries of Sec. III A. In this respect, we are working with Intel software engineers to guarantee complete compatibility for their future Ponte Vecchio GPU architecture, and QUANTUM ESPRESSO will be ready to run on these cards when they will appear on the market, toward the end of 2021.[87] Porting to AMD accelerated architectures is also ongoing, exploiting the public OpenACC stack. Work on ARM vectorized (SVE instruction set) architectures (EPI[88] and AFX64[89] processors) is also ongoing and on track to release an optimized

QUANTUM ESPRESSO version for these architectures. All these porting efforts would be hardly feasible without the contribution of IT experts and HPC centers, in the spirit of the separation of concerns. This solution does not target the maximum level of architecture-specific optimizations but leaves the possibility open to achieve them, once the hardware is finally installed in major HPC centers, using specially tailored patched versions.

The benchmarks presented in Sec. III C are realistic but relatively small-size use cases. The extension to large-scale calculations requires further work to identify and remove memory and computation bottlenecks. In particular, the amount of global memory per card is a relevant parameter that substantially impacts the performance of the accelerated version of the code. Some of the largest benchmarks in our set do saturate the memory available on a single card, depending upon the parallel options used to distribute plane-wave components. Similar issues will certainly show up also in extreme scale benchmarks. For single-point self-consistent or molecular-dynamics calculations, the most critical bottlenecks toward scaling to the whole exascale system will be the amount of memory in cards due to the superlinear complexity of plane-wave electronic structure calculations (the amount of memory and computation increases more than quadratically with the size of the system). It is foreseen that the best "exaflop" performances will be actually achievable for selected cases, such as "high-throughput" and "ensemble" large-scale calculations, which can be split into many smaller ones. The efficiency of each computation will thus be critical for fully exploiting the capability of the HPC systems.

The data that support the findings of this study are openly available in the Materials Cloud Archive at https://doi.org/10.24435/materialscloud:2020.0021/v1.[84]

REFERENCES

[1]S. Baroni, P. Giannozzi, and A. Testa, "Green's function approach to linear response in solids," Phys. Rev. Lett. **58**, 1861–1864 (1987).

[2]P. Giannozzi, S. de Gironcoli, P. Pavone, and S. Baroni, "*Ab initio* calculation of phonon dispersions in semiconductors," Phys. Rev. B **43**, 7231 (1991).

[3]S. Baroni, S. de Gironcoli, A. Dal Corso, and P. Giannozzi, "Phonons and related crystal properties from density-functional perturbation theory," Rev. Mod. Phys. **73**, 515–562 (2001).

[4]X. Gonze, "Adiabatic density-functional perturbation theory," Phys. Rev. A **52**, 1096–1114 (1995).

[5]X. Gonze, "Erratum: Adiabatic density-functional perturbation theory," Phys. Rev. A **54**, 4591 (1996).

[6]R. Car and M. Parrinello, "Unified approach for molecular dynamics and density-functional theory," Phys. Rev. Lett. **55**(22), 2471–2474 (1985).

[7]At the time, PWscf had already been released under the GPL.

[8]P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, S. Alexander, P. Umari, and R. M. Wentzcovitch, "Quantum ESPRESSO: A modular and open-source software project for quantum simulations of materials," J. Phys.: Condens. Matter **21**(39), 395502 (2009).

[9]P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. Buongiorno Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. Dal Corso, S. de Gironcoli, P. Delugas, A. R. DiStasio, Jr., A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-K. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. Otero-de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni, "Advanced capabilities for materials modelling with Quantum ESPRESSO," J. Phys.: Condens. Matter **29**, 465901 (2017).

[10]I. Timrov, N. Marzari, and M. Cococcioni, "Hubbard parameters from density-functional perturbation theory," Phys. Rev. B **98**, 085127 (2018).

[11]V. Kapil, M. Rossi, O. Marsalek, R. Petraglia, Y. Litman, T. Spura, B. Cheng, A. Cuzzocrea, R. H. Meißner, D. M. Wilkins, B. A. Helfrecht, P. Juda, S. P. Bienvenue, W. Fang, J. Kessler, I. Poltavsky, S. Vandenbrande, J. Wieme, C. Corminboeuf, T. D. Kühne, D. E. Manolopoulos, T. E. Markland, J. O. Richardson, A. Tkatchenko, G. A. Tribello, V. Van Speybroeck, and M. Ceriotti, "i-PI 2.0: A universal force engine for advanced molecular simulations," Comput. Phys. Commun. **236**, 214–223 (2019).

[12]G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky, "AiiDA: Automated interactive infrastructure and database for computational science," Comput. Mater. Sci. **111**, 218–230 (2016).

[13]A. Genova, D. Ceresoli, A. Krishtal, O. Andreussi, A. R. DiStasio, Jr., and M. Pavanello, "eQE: An open-source density functional embedding theory code for the condensed phase," Int. J. Quantum Chem. **117**(16), e25401 (2017).

[14]J. Kim, A. D. Baczewski, T. D. Beaudet, A. Benali, M. C. Bennett, M. A. Berrill, N. S. Blunt, E. J. L. Borda, M. Casula, D. M. Ceperley, S. Chiesa, B. K. Clark, R. C. Clay, K. T. Delaney, M. Dewing, K. P. Esler, H. Hao, O. Heinonen, P. R. C. Kent, J. T. Krogel, I. Kylänpää, Y. W. Li, M. G. Lopez, Y. Luo, F. D. Malone, R. M. Martin, A. Mathuriya, J. McMinis, C. A. Melton, L. Mitas, M. A. Morales, E. Neuscamman, W. D. Parker, S. D. Pineda Flores, N. A. Romero, B. M. Rubenstein, J. A. R. Shea, H. Shin, L. Shulenburger, A. F. Tillack, J. P. Townsend, N. M. Tubman, B. Van Der Goetz, J. E. Vincent, D. C. Yang, Y. Yang, S. Zhang, and L. Zhao, "QMCPACK: An open source ab initio quantum Monte Carlo package for the electronic structure of atoms, molecules and solids," J. Phys.: Condens. Matter **30**(19), 195901 (2018).

[15]D. Sangalli, A. Ferretti, H. Miranda, C. Attaccalite, I. Marri, E. Cannuccia, P. Melo, M. Marsili, F. Paleari, A. Marrazzo, G. Prandini, P. Bonfà, M. O. Atambo, F. Affinito, M. Palummo, A. Molina-Sánchez, C. Hogan, M. Grüning, D. Varsano, and A. Marini, "Many-body perturbation theory calculations using the yambo code," J. Phys.: Condens. Matter **31**(32), 325902 (2019).

[16]J. Deslippe, G. Samsonidze, D. A. Strubbe, M. Jain, M. L. Cohen, and S. G. Louie, "BerkeleyGW: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures," Comput. Phys. Commun. **183**(6), 1269–1289 (2012).

[17]M. Govoni and G. Galli, "Large scale GW calculations," J. Chem. Theory Comput. **11**, 2680–2696 (2015).

[18]M. Schlipf, H. Lambert, N. Zibouche, and F. Giustino, "SternheimerGW: A program for calculating GW quasiparticle band structures and spectral functions without unoccupied states," Comput. Phys. Commun. **247**, 106856 (2020).

[19]G. Pizzi, V. Vitale, R. Arita, S. Blügel, F. Freimuth, G. Géranton, M. Gibertini, D. Gresch, C. Johnson, T. Koretsune, J. Ibañez-Azpiroz, H. Lee, J.-M. Lihm, D. Marchand, A. Marrazzo, Y. Mokrousov, J. I. Mustafa, Y. Nohara, Y. Nomura, L. Paulatto, S. Poncé, T. Ponweiser, J. Qiao, F. Thöle, S. S. Tsirkin, M. Wierzbowska, N. Marzari, D. Vanderbilt, I. Souza, A. A. Mostofi, and J. R. Yates, "Wannier90 as a community code: New features and applications," J. Phys.: Condens. Matter **32**(16), 165902 (2020).

[20]S. Poncé, E. R. Margine, C. Verdi, and F. Giustino, "EPW: Electron-phonon coupling, transport and superconducting properties using maximally localized wannier functions," Comput. Phys. Commun. **209**, 116–133 (2016).

[21]A. Otero-de-la-Roza, E. R. Johnson, and V. Luaña, "Critic2: A program for real-space analysis of quantum chemical interactions in solids," Comput. Phys. Commun. **185**(3), 1007–1018 (2014).

[22]A. Ferretti, A. Calzolari, B. Bonferroni, and R. D. Felice, "Maximally-localized Wannier functions from PAW or ultrasoft pseudopotentials," J. Phys.: Condens. Matter **19**, 036215 (2007).

[23]L. Calderín, V. V. Karasiev, and S. B. Trickey, "Kubo–Greenwood electrical conductivity formulation and implementation for projector augmented wave datasets," Comput. Phys. Commun. **221**, 118–142 (2017).

[24]See https://www.top500.org/project/ for Top500 supercomputers, 2019.

[25]M. Feldman, New GPU-Accelerated supercomputers change the balance of power on the TOP500, June 2018. https://www.top500.org/news/new-gpu-accelerated-supercomputers-change-the-balance-of-power-on-the-top500.

[26]MaX: Materials at the eXascale. An EU Centre of Excellence for Supercomputing Applications. https://www.max-centre.eu.

[27]H. Carter Edwards, C. R. Trott, and D. Sunderland, "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns," J. Parallel Distrib. Comput. **74**(12), 3202–3216 (2014); S. Krishnamoorthy, J. Ramanujam, and P. Sadayappan, "Introduction to the JPDC special issue on domain-specific languages and high-level frameworks for high-performance computing," *ibid.* **74**, 3175 (2014).

[28]E. Zenker, B. Worpitz, R. Widera, A. Huebl, G. Juckeland, A. Knüpfer, W. E. Nagel, and M. Bussmann, *Alpaka—An Abstraction Library for Parallel Kernel Acceleration* (IEEE Computer Society, 2016).

[29]A. Matthes, R. Widera, E. Zenker, B. Worpitz, A. Huebl, and M. Bussmann, "Tuning and optimization for a variety of many-core architectures without changing a single line of implementation code using the Alpaka library," in *High Performance Computing* (Springer, 2017).

[30] R. Hornung, H. Jones, J. Keasler, R. Neely, O. Pearce, S. Hammond, C. Trott, P. Lin, C. Vaughan, J. Cook, R. Hoekstra, B. Bergen, J. Payne, and G. Womeldorff, ASC tri-lab co-design level 2 milestone report 2015, Report No. LLNL-TR-677453, September 23, 2015.

[31] See https://esl.cecam.org for ESL—the electronic structure library.

[32] See https://gitlab.e-cam2020.eu/esl/ESLW_Drivers for electronic structure library coding workshop: Drivers. Trieste; 10–21 July 2017.

[33] S. de Gironcoli, "Lattice dynamics of metals from density-functional perturbation theory," Phys. Rev. B **51**, 6773(R) (1995).

[34] A. Dal Corso, A. Pasquarello, and A. Baldereschi, "Density-functional perturbation theory for lattice dynamics with ultrasoft pseudopotentials," Phys. Rev. B **56**, R11369(R) (1997).

[35] A. Dal Corso and S. de Gironcoli, "*Ab initio* phonon dispersions of Fe and Ni," Phys. Rev. B **62**, 273 (2000).

[36] A. Dal Corso, "Density-functional perturbation theory with ultrasoft pseudopotentials," Phys. Rev. B **64**, 235118 (2001).

[37] A. Dal Corso, "Density functional perturbation theory for lattice dynamics with fully relativistic ultrasoft pseudopotentials: Application to fcc-Pt and fcc-Au," Phys. Rev. B **76**, 054308 (2007).

[38] A. Dal Corso, "Density functional perturbation theory within the projector augmented wave method," Phys. Rev. B **81**, 075123 (2010).

[39] A. Floris, S. de Gironcoli, E. K. U. Gross, and M. Cococcioni, "Vibrational properties of MnO and NiO from DFT + U-based density functional perturbation theory," Phys. Rev. B **84**, 161102(R) (2011).

[40] R. Sabatini, E. Küçükbenli, C. H. Pham, and S. de Gironcoli, "Phonons in nonlocal van der Waals density functional theory," Phys. Rev. B **93**, 235120 (2016).

[41] T. Sohier, M. Calandra, and F. Mauri, "Density functional perturbation theory for gated two-dimensional heterostructures: Theoretical developments and application to flexural phonons in graphene," Phys. Rev. B **96**, 075448 (2017).

[42] A. Urru and A. Dal Corso, "Density functional perturbation theory for lattice dynamics with fully relativistic ultrasoft pseudopotentials: The magnetic case," Phys. Rev. B **100**, 045115 (2019).

[43] A. Floris, I. Timrov, B. Himmetoglu, N. Marzari, S. de Gironcoli, and M. Cococcioni, "Hubbard-corrected density functional perturbation theory with ultrasoft pseudopotentials," Phys. Rev. B **101**, 064305 (2020).

[44] S. Baroni and R. Gebauer, "The Liouville-Lanczos approach to time-dependent density-functional (perturbation) theory," in *Fundamentals of Time-dependent Density Functional Theory, Lecture Notes in Physics*, edited by M. A. L. Marques, N. T. Maitra, F. M. S. Nogueira, E. K. U. Gross, and A. Rubio (Springer-Verlag, Berlin, Heidelberg, 2012), Vol. 837, Chap. 19, pp. 375–390.

[45] B. Walker, A. M. Saitta, R. Gebauer, and S. Baroni, "Efficient approach to time-dependent density-functional perturbation theory for optical spectroscopy," Phys. Rev. Lett. **96**(11), 113001 (2006).

[46] D. Rocca, R. Gebauer, Y. Saad, and S. Baroni, "Turbo charging time-dependent density-functional theory with Lanczos chains," J. Chem. Phys. **128**(15), 154105 (2008).

[47] O. B. Malcioğlu, R. Gebauer, D. Rocca, and S. Baroni, "turboTDDFT—A code for the simulation of molecular spectra using the Liouville–Lanczos approach to time-dependent density-functional perturbation theory," Comput. Phys. Commun. **182**(8), 1744–1754 (2011).

[48] X. Ge, S. J. Binnie, D. Rocca, R. Gebauer, and S. Baroni, "turboTDDFT 2.0 — Hybrid functionals and new algorithms within time-dependent density-functional perturbation theory," Comput. Phys. Commun. **185**, 2080–2089 (2014).

[49] I. Timrov, O. Andreussi, A. Biancardi, N. Marzari, and S. Baroni, "Self-consistent continuum solvation for optical absorption of complex molecular systems in solution," J. Chem. Phys. **142**, 034111 (2015).

[50] I. Timrov, N. Vast, R. Gebauer, and S. Baroni, "Electron energy loss and inelastic x-ray scattering cross sections from time-dependent density-functional perturbation theory," Phys. Rev. B **88**(6), 064301 (2013); **91**, 139901(E) (2015).

[51] I. Timrov, N. Vast, R. Gebauer, and S. Baroni, "turboEELS–A code for the simulation of the electron energy loss and inelastic X-ray scattering spectra using the Liouville–Lanczos approach to time-dependent density-functional perturbation theory," Comput. Phys. Commun. **196**, 460–469 (2015).

[52] O. Motornyi, N. Vast, I. Timrov, O. Baseggio, S. Baroni, and A. Dal Corso, "Electron energy loss spectroscopy of bulk gold with ultrasoft pseudopotentials and the Liouville–Lanczos method," Phys. Rev. B (unpublished) (2020).

[53] T. Gorni, I. Timrov, and S. Baroni, "Spin dynamics from time-dependent density functional perturbation theory," Eur. Phys. J. B **91**(10), 249 (2018).

[54] See https://www.w3.org/standards/xml/schema for XML schema definition (XSD).

[55] See https://pypi.org/project/xmlschema/ for the xmlschema package.

[56] See https://gitlab.com/QEF/xmltool for the xmltool package.

[57] See https://pypi.org/project/qeschema/ for the qeschema package.

[58] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—A python library for working with atoms," J. Phys.: Condens. Matter **29**, 273002 (2017).

[59] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, and G. Ceder, "Python materials genomics (pymatgen): A robust, open-source python library for materials analysis," Comput. Mater. Sci. **68**, 314–319 (2013).

[60] A. Kokalj, "XCrySDen—A new program for displaying crystalline structures and electron densities," J. Mol. Graphics Model. **17**(3), 176–179 (1999).

[61] See https://github.com/QEF/postqe for the postQE package.

[62] F. Spiga and I. Girotto, "phiGEMM: A CPU-GPU library for porting QUANTUM ESPRESSO on Hybrid systems," in Proceedings of the 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing (IEEE, 2012).

[63] F. Spiga, QE-GPU: GPU-Accelerated QUANTUM ESPRESSO, 2012–2017.

[64] I. Girotto, N. Varini, F. Spiga, C. Cavazzoni, D. Ceresoli, L. Martin-Samos, and T. Gorni, Enabling of QUANTUM ESPRESSO to petascale scientific challenges. PRACE, 2012.

[65] J. Romero, E. Phillips, G. Ruetsch, M. Fatica, F. Spiga, and P. Giannozzi, "A performance study of QUANTUM ESPRESSO's PWscf code on multi-core and GPU systems," in *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems* (Springer, 2018), pp. 67–87.

[66] M. Fatica, J. Romero, E. Phillips, and F. Spiga (2017). "GPU-accelerated QUANTUM ESPRESSO," Zenodo. https://doi.org/10.5281/zenodo.823200.

[67] Optionally, cuRAND can also be used for the generation of random wavefunctions.

[68] C. Cavazzoni, F. Affinito, U. Alekseeva, C. Cardoso, A. Degomme, P. Delugas, A. Ferretti, A. Garcia, L. Genovese, P. Giannozzi, A. Kozhevnikov, I. Marri, S. Mohr, and D. Wortmann, First report on code profiling and bottleneck identification, structured plan of forward activities. Deliverable D4.2 of the H2020 CoE MAX. EC Grant Agreement No: 824143, CINECA, Bologna, Italy (2019). max-centre.eu/project-repository.

[69] D. Wortmann, U. Alekseeva, S. Baroni, A. Degomme, P. Delugas, S. de Gironcoli, A. Ferretti, A. Garcia, L. Genovese, P. Giannozzi, A. Kozhevnikov, and I. Marri, First release of MAX software: report on the performance portability. Deliverable D2.1 of the H2020 CoE MAX. EC Grant Agreement No: 824143, Jülich, Germany (2019). max-centre.eu/project-repository.

[70] See https://www.materialscloud.org/work/tools/qeinputgenerator for QUANTUM ESPRESSO input generator and structure visualizer.

[71] G. Prandini, A. Marrazzo, I. E. Castelli, N. Mounet, and N. Marzari, "Precision and efficiency in solid-state pseudopotential calculations," npj Comput. Mater. **4**(1), 72 (2018).

[72] A. Willand, Y. O. Kvashnin, L. Genovese, Á. Vázquez-Mayagoitia, A. K. Deb, A. Sadeghi, T. Deutsch, and S. Goedecker, "Norm-conserving pseudopotentials with chemical accuracy compared to all-electron calculations," J. Chem. Phys. **138**(10), 104109 (2013).

[73] M. Schlipf and F. Gygi, "Optimization algorithm for the generation of ONCV pseudopotentials," Comput. Phys. Commun. **196**, 36–44 (2015).

[74] M. J. van Setten, M. Giantomassi, E. Bousquet, M. J. Verstraete, D. R. Hamann, X. Gonze, and G.-M. Rignanese, "The PseudoDojo: Training and grading a 85 element optimized norm-conserving pseudopotential table," Comput. Phys. Commun. **226**, 39–54 (2018).

[75] K. F. Garrity, J. W. Bennett, K. M. Rabe, and D. Vanderbilt, "Pseudopotentials for high-throughput DFT calculations," Comput. Mater. Sci. **81**, 446–452 (2014).

[76] A. Dal Corso, "Pseudopotentials periodic table: From H to Pu," Comput. Mater. Sci. **95**, 337–350 (2014).

[77] M. Topsakal and R. M. Wentzcovitch, "Accurate projected augmented wave (PAW) datasets for rare-earth elements (RE = La–Lu)," Comput. Mater. Sci. **95**, 263–270 (2014).

[78] M. Quirós, S. Gražulis, S. Girdzijauskaitė, A. Merkys, and A. Vaitkus, "Using SMILES strings for the description of chemical connectivity in the Crystallography Open Database," J. Cheminf. **10**(1), 23 (2018).

[79] A. Merkys, A. Vaitkus, J. Butkus, M. Okulič-Kazarinas, V. Kairys, and S. Gražulis, "COD::CIF::Parser: An error-correcting CIF parser for the Perl language," J. Appl. Crystallogr. **49**(1), 292 (2016).

[80] S. Gražulis, A. Merkys, A. Vaitkus, and M. Okulič-Kazarinas, "Computing stoichiometric molecular composition from crystal structures," J. Appl. Crystallogr. **48**(1), 85–91 (2015).

[81] S. Gražulis, A. Daškevič, A. Merkys, D. Chateigner, L. Lutterotti, M. Quirós, N. R. Serebryanaya, M. Peter, R. T. Downs, and A. Le Bail, "Crystallography open database (COD): An open-access collection of crystal structures and platform for world-wide collaboration," Nucleic Acids Res. **40**(D1), D420–D427 (2012).

[82] S. Gražulis, D. Chateigner, R. T. Downs, A. F. T. Yokochi, M. Quirós, L. Lutterotti, E. Manakova, J. Butkus, M. Peter, and A. Le Bail, "Crystallography Open Database—An open-access collection of crystal structures," J. Appl. Crystallogr. **42**(4), 726–729 (2009).

[83] R. T. Downs and M. Hall-Wallace, "The American Mineralogist crystal structure database," Am. Mineral. **88**, 247–250 (2003).

[84] P. Giannozzi, O. Baseggio, P. Bonfà, R. Car, I. Carnimeo, C. Cavazzoni, S. de Gironcoli, P. Delugas, F. F. Ruffino, A. Ferretti, N. Marzari, I. Timrov, A. Urru, and S. Baroni, "Quantum ESPRESSO toward the exascale," available in the Materials Cloud Archive at https://doi.org/10.24435/materialscloud:2020.0021/v1.

[85] Although as a rule the code should be executed with one MPI per GPU card, over-subscription of the GPU can lead to significant speedups, especially for small test cases. Nonetheless, over-subscription by more than a factor of 4 is seldom useful.

[86] This can be appreciated by comparing simulations that required similar computational time on the CPU but used different pseudopotential sets, for example, the $O_{10}P_2Ti$ and CaN_2Si columns on the right panel of Fig. 2.

[87] See https://www.hpcwire.com/2019/03/18/its-official-aurora-on-track-to-be-first-u-s-exascale-computer-in-2021/ for first US exascale computer.

[88] See https://www.european-processor-initiative.eu/project/epi/ for European processor initiative.

[89] See https://www.hpcwire.com/off-the-wire/fujitsu-begins-shipping-supercomputer-fugaku for Fujitsu supercomputer Fugaku.

[90] See http://foundation.quantum-espresso.org for the Quantum ESPRESSO foundation.