Between the idea and the reality / falls the … software!

SOFTWARE FOR QUANTUM SIMULATIONS OF TOMORROW

Paolo Giannozzi, Università di Udine e IOM-CNR Trieste

Published as *Il Nuovo Saggiatore* **35**, n.5-6, pp. 34-38 (2019)

Profilo autore: *Paolo Giannozzi graduated at the University of Pisa (1982) and obtained his Ph.D. at the University of Lausanne (1988). He spent various periods in EPF Lausanne, Scuola Normale Superiore Pisa, SISSA Trieste, IBM Research Laboratories Zurich, CECAM Lyon, CECAM Lausanne, Princeton University. Since 2006 he is associate professor of condensed-matter physics at the University of Udine. His research field is the developments of methodologies for density-functional theory and their application to nano-structures. He is currently one of the leading developers of the Quantum ESPRESSO distribution of software and one of the coordinators of its development.*

Scientific software allows to transform theories and methods into numerical results and physical insight. This is especially true for the study of realistic models of materials, that heavily rely on first-principle computer simulations. This article describes the problems encountered in the development and maintenance of scientific software and examines the possible solutions. The ongoing work towards better software for present and future simulations of materials, performed with the Quantum ESPRESSO distribution in the framework of the EU-supported MaX Centre of Excellence, is introduced.

1. On first-principle simulations

The field of first-principle quantum simulations (that is: from the electronic structure) of materials has witnessed an explosive growth since the 80's. In condensed-matter physics, the development of density-functional theory (DFT) has been one of the main drivers of the growth, leading to a wealth of applications in fields ranging from "classical" solid-state physics for simple semiconductors and metals, to more complex heterostructures and nanostructures, to exotic metallo-organic and biological systems. The scope of computable properties has meanwhile grown from ground-state properties, such as structural stability, electronic, mechanical and vibrational properties, into the realm of excited-states properties. In parallel, better-performing functionals have allowed a significant increase in the accuracy of results. As a quick and unscientific indicator of the success of DFT-based quantum simulations, one can browse any list of most cited physics papers: a surprising large fraction of them are DFT-related. The interested reader is invited to read Giovanni Pizzi's paper in this Journal [1] for a review of what can be achieved with DFT simulations.

In addition to the improvements in the theory and in the methodology, a crucial role in this success story is played by the enormous increase in computer power since the first supercomputers were available. Simulations that used to be at the frontier in the 80's can be now performed on a desktop PC. While the increase of clock speed has flattened out since several years, under the constraints of semiconductor physics and in the absence of major technological breakthroughs, the increase in the global computing power has progressed via the increase in the number and complexity of central processing units (CPU's).

2. On software in physics

There is however a third actor standing in the middle between theory and computers: the scientific software implementing those theories. Simulations of realistic condensed-matter systems can be time- and memory-consuming, thus requiring sophisticated numerical techniques, implemented into computer codes typically written in Fortran or C/C++, that may run on powerful high-performance computers (HPC).

The importance of proper development and maintenance of scientific software has been overlooked for a long time, especially in fields like quantum simulations where home-brewed software was, at least until ∼ 15 years ago, the norm. Individual researchers and research groups typically had a private version of some software, sometimes developed and maintained "in house", sometimes obtained from other individuals and groups and modified to fit local needs. This lead to much duplicated effort, diverging versions of the same code, and much code of questionable quality, often written in a hurry and with little or no documentation by Ph.D. students or post-docs with other priorities (typically: finding the next job) than good coding, eventually left to *bit-rotting*[1]. Overall, a very unsatisfactory situation.

Codes for early numerical simulations consisted of a few thousands of lines of instructions at most, plus a few basic mathematical libraries; today's codes often include several hundreds of thousands of lines of instructions. During the years, two main factors have pushed computer codes towards increasing complexity: i) the growth in sophistication of theories and methodologies and their extension to the calculation of more properties, and ii) the increased complexity of HPC machines. While the former factor is an unavoidable, and even desirable, consequence of scientific progress, the latter is something most physicists would prefer not to hear about (but too often they have to). During the years, many codes have been ported to or even re-written for all kinds of computer architectures, most of which long forgotten: vector machines (in the '80s), parallel machines (in the '90s), massively parallel machines (e.g., the Blue Gene), multi-core machines (until recently), ... The latest trends in HPC (see Carlo Cavazzoni's paper in this Journal) [2] are even more worrisome: with the advent of heterogeneous architectures, in which CPU's can offload computation to specialized hardware (GPU's: graphical processing units, or similar "accelerators"), a massive re-writing of codes seems unavoidable. Accelerators in fact do not accelerate anything without specific machine-dependent coding. The old joke known as 7th law of programming: *Program complexity grows until it exceeds the capability of the programmer who must maintain it*, seems increasingly close to reality, especially for small research groups.

3. On open-source software

One may argue that software has become a commodity: today, several high-quality commercial scientific software suites are offered by specialized software houses, typically academic spin-offs. After all, in the quantum-chemistry community the leading code for molecular calculations is developed and licensed - at a hefty price for companies, with a large discount for academia - by such a company.

Commercial software has the obvious advantage of providing the means to develop and maintain a complex code in a professional way. It is not however the solution for everybody and for every usage. In developing countries, or for very small groups, the price may already be an obstacle, but even if it is not, software in physics is often stretched to the limits of its capabilities, adapted to local needs, used for testing and development of new ideas and algorithms. All this requires free access to the sources, that not all commercial software suites provide.

There are other more "philosophical" aspects in favor of free access to sources

- Correctness. The probability to spot bugs, or errors in analytical formulae or numerical algorithms implemented in software, increases with the number of eyes that can look into the codes.
- Collaboration. Different research groups can more easily share knowledge and codes.
- Reproducibility. Science, including "numerical" experiments, is based upon it. Reproducibility of computer simulations, sometimes not obvious even under ideal conditions (e.g., obtaining the same molecular dynamics trajectories is in practice impossible), is seriously hindered by the lack of access to software and to the data needed to run it (e.g., pseudopotentials for electronic-structure calculations).

1 the mysterious decay phenomenon that makes software unusable after a few years or even months at rest.

All these factors have pushed many research groups towards the adoption of so-called "open-source" licenses for software. The most popular of them is likely the GNU General Public License (GPL).

More recently, the need for an analogous open access to data used or produced by software has started to be appreciated in all its importance (see [1] for more on this subject).

4. Origins of the Quantum ESPRESSO distribution

More than 20 years ago, a group of Italian physicists, based in or loosely connected to Trieste, considered the idea to merge different versions of the same set of codes for first-principle simulations already in use since several years, into a unified "community software", to be developed and maintained in a more professional manner using modern software. The main goals of such endeavor were

- innovation in theoretical methods and numerical algorithms,
- efficiency on modern computer architectures,
- formation of a user and developer community.

The release of code into the public domain with an open-source license was already common in other domains but still relatively new for scientific software. It took a few more years before the idea condensed into what is known as Quantum ESPRESSO (QE) [web site: www.quantum-espresso.org]. "ESPRESSO" stands for opEn-Source Package for Research in Electronic Structure, Simulation, and Optimization, while "Quantum" stresses both the quantum nature of simulation and the difference with another scientific software project, on soft matter, with a similar name. QE is not a single code or a single package. It is more alike to an integrated suite: a "distribution", to which everybody can contribute new packages. Contributions are required to loosely follow some general programming rules, not to stick to a rigid software architecture: no need to vex potential contributors with unnecessary restrictions.

The initial release of QE contained codes for self-consistent calculations, first-principle molecular dynamics (Car-Parrinello), linear response (phonons), plus a set of utilities and tools for post-processing of data produced by simulations. These codes were the results of years (literally) of efforts by permanent and temporary staff and graduate students at various institutions in Italy and abroad. They consisted of a few tens of thousands of Fortran-77 lines. There was a single parallelization level, on plane waves using the PVM (now obsolete) and MPI (message-passing interface) libraries. Parallel machines of the time were simple clusters of workstations, connected by fast, and sometimes not-so-fast, communication hardware.

5. Quantum ESPRESSO today

The need to keep QE aligned with theoretical and methodological progresses and with new computer architectures has during the years transformed QE into a much more complex distribution. QE has been converted to modern Fortran (90/95/2003), ported to all kinds of architectures, has seen the implementation of multiple parallelization levels and of new packages. Today the core distribution includes almost 600000 lines of Fortran code, 60000 lines of C and of shell and python scripts, plus a large amount of tests, examples, documentation.

The management of such a relatively large software project is close to impossible without resorting to the help of dedicated software tools. After the initial experience with CVS,[2] QE moved to Subversion (svn), hosted on a Contents Management System (CMS) known as "qe-forge.org", open to anybody desiring to develop open-source atomistic simulation software. Recently the development has moved to "git", a very sophisticated and complex software maintenance tool, complemented by an even more sophisticated and complex web interface [web site: GitLab.com/QEF/q-e].

2 Concurrent versioning system: a simple but reliable tool for keeping track of the software history and versions in a centralized repository

Initial support for QE came from the DEMOCRITOS National Simulation Center of the much-regretted National Institute for the Physics of Matter. Currently the maintenance and further development of QE is fostered by the Quantum ESPRESSO Foundation (QEF), a not-for-profit company incorporated in the UK, created to protect the name, logo (fig.1), web domains, and trademarks of QE. The QEF also raises funds for the development of QE via agreements with commercial entities, following the open-source model. This in fact does not forbid commercial exploitation as long as sources remain available and free: one does not sell the software itself but services related to the software.

Fig. 1 Quantum ESPRESSO logo

A peculiar aspect of QE is the large number of schools and tutorials organized around the world. The desire to bring first-principle techniques to a wider audience, and the presence of ICTP in Trieste with its mission to bring knowledge to developing countries, naturally lead to the organization of a series of "QE Schools" in various places of the world. 36 such schools have been organized to date, the first in 2004 in Beijing, the latest in September 2019 in Ljubljana.

The usage and diffusion of an open-source software is hard to measure. We have however some estimates and indicators:

- the number of subscribers to the mailing list (more than 1000)
- the total number of attendees to schools (no less than 1600)
- the number of downloads of each release (more than 4000)
- the number of citations received by the papers [3,4] documenting QE (15000 since 2009)

all pointing towards a significant base of users.

In addition to being used directly, QE is also in use in other independent software projects as a "quantum engine", or to produce data for subsequent processing. Graphical user interfaces (GUI's) to QE have been developed by other groups as well and there is an ongoing commercial agreement with a major scientific software vendor that distributes QE together with its proprietary GUI. Other manifestations of interest by commercial entities have also been received. It is fair to say that QE is one of the most successful software suites in the field of electronic structure calculations.

6. On exascale and exascale computing

Quantum simulations are computationally expensive. The size of the system that can be simulated, the configuration space that can be explored, the level of theory that can be used, the type of property that can be computed, are in practice constrained by computer resources. The development of efficient methodological and numerical techniques has always been a major goal in QE. Better theories and better algorithms must however be accompanied by an efficient exploitation of the computing power of available computers. The latter is an ever-shifting target, often conflicting with portability and making software maintenance harder.

A time-honored technique to achieve performances without sacrificing portability and maintainability is to offload computing-intensive kernels -- typically, linear-algebra and fast Fourier transforms (FFT) operations -- to machine-optimized mathematical libraries. On parallel machines, portability and maintainability were favored by the availability of a standard library (MPI) for inter-process communications. In QE, basic parallelization operations were thus encapsulated into a few routines, hiding the technical details of parallelization from the sight of developers (except the few people directly involved in parallel algorithms). For machines based on multiple-core computing nodes, the OpenMP standard for light-weight in-core parallelization (threads) allowed to achieve good performances via compiler directives, with minor changes to the source code.

The recent trends in HPC machines[2] are dictated by the need to reduce the energy consumption. Accelerators -- GPU's and the like, have a much better flops-per-watt ratio than conventional multi-core CPU's, thus holding the promise to make an "exascale" machine (that is: executing 1 exaflop, 10^{18} floating-point operations per second) feasible. Most supercomputers already have some form of acceleration, typically via one or more GPU's per compute node (a multi-core CPU). Being able to exploit such a computer power -- two orders of magnitude with respect to the most powerful machines available today -- opens entirely new perspectives in terms of what can be computed.

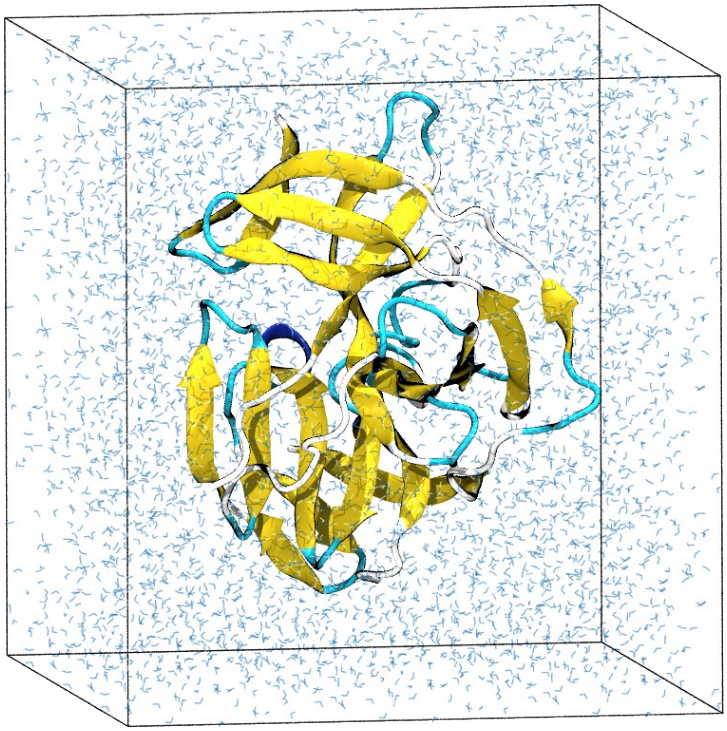

Fig. 2 The largest calculation made with `Quantum ESPRESSO`

An exascale machine is unlikely to be available before 2023, but pre-exascale machines are expected to hit the market by 2020-2021. It is however still unclear what they will look like: as of today, there are three competing candidate architectures. The road towards exascale is made hard not only by the uncertainty about future machines but also by the non-existence of anything like the standard mathematical libraries, MPI and OpenMP, that have made life easy, or less difficult, for developers of electronic-structure codes.

The new accelerated architectures require in fact a major re-organization of the way calculations are performed. Data need to be copied from CPU to GPU, where it is processed, then results copied back to CPU. The bottleneck is data communication: once data is on the GPU, computation is extraordinarily fast. Direct communication between different GPU's is still not possible or hard to use, so some strategies that have been successfully implemented for parallel computing, e.g. for three-dimensional distributed parallel FFT, may no longer be suitable and may require significant revisions.

7. Quantum ESPRESSO at the exascale

It would be a pity if the electronic-structure community will not be ready to exploit the power of exascale machines when they arrive. The EU-supported MaX (Materials design at the eXascale) Horizon-2020 Centre of excellence [web site: www.max-centre.eu], lead by CNR-NANO in Modena, addresses such issue. MaX is active since 2015 and is the second phase of a previous successful Horizon-2020 project, aimed towards establishing a software "computational infrastructure". The general and ambitious goal of MaX is to provide physicists with a sustainable framework for electronic-structure software development.

In particular, MaX aims to make the six "community codes" participating to the project: QE, Siesta, Yambo, BigDFT, Fleur, CP2K, ready for future exascale machines. Isolating and removing all bottlenecks, notably non-distributed computation and arrays, is a prerequisite for approaching exascale. A significant further effort is needed in order to achieve what is dubbed "separation of concerns", that is: to isolate the physics-related code layers, of interest to scientists, from computational kernels. The latter are the target of more aggressive optimization, possibly done independently from the original codes, with the help of HPC and information technology (IT) experts. The efforts of MaX, in conjunction with the five participating European HPC centers (CINECA, BSC, Juelich, CSCS, CEA), have already lead to the definition of important common libraries, for FFT and linear algebra, extracted from existing codes.

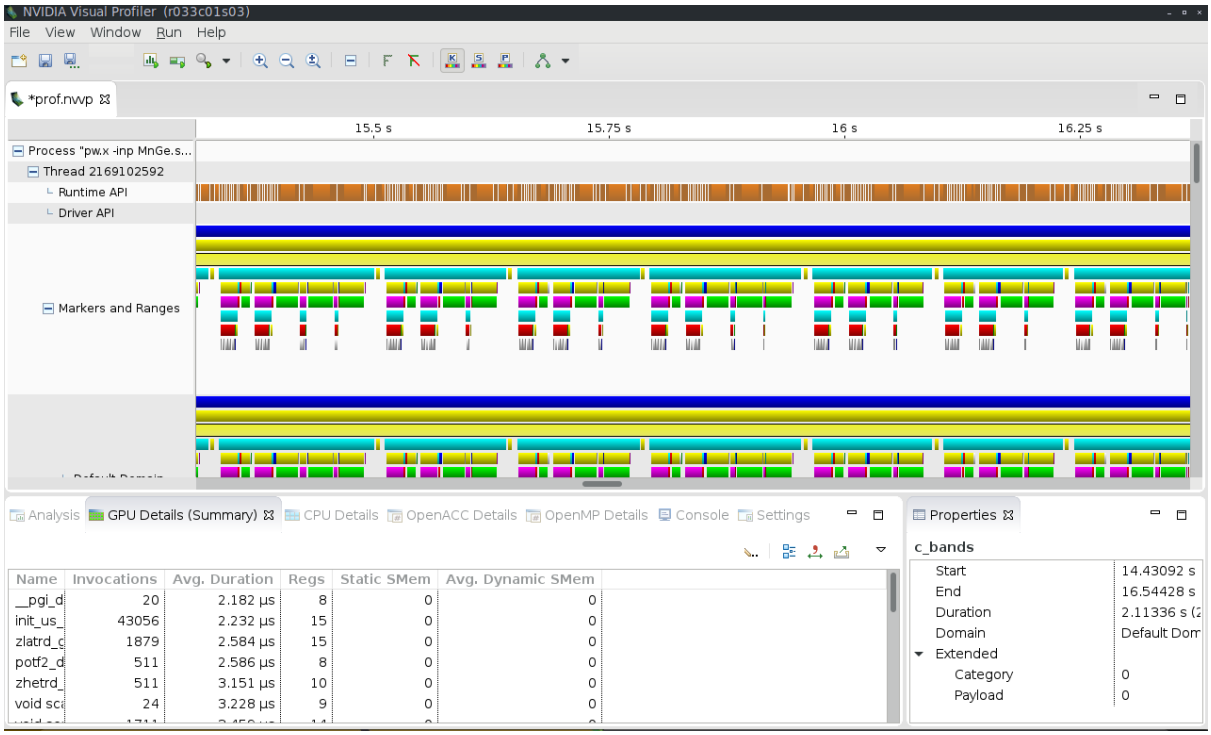

Fig. 3 Chasing for bottlenecks and inefficiency with a profiler

MaX is also active in the field of the so-called "co-design", that is: the involvement of all partners in HPC -- from hardware vendors to HPC centers and to scientists developing electronic-structure codes – towards a mutual understanding of what is important for producing effective software. This is the only possibility for scientists, whose "market share" is dwarfed by videogames and by the various AI applications, to have some leverage on the choices made by machine builders. A first result of the collaboration with IT experts at Nvidia (the leading GPU producer) is a QE version that works on Nvidia GPU's. This version is by now production-ready and is being continuously extended and improved (fig.3). Written in Fortran, the distribution of computation and of memory between CPU and GPU is managed with directives that are recognized by the PGI compiler (distributed for no charge by Nvidia). The code is being structured in a way that will allow in the future a relatively easy porting to different kinds of heterogeneous architectures.

8. Conclusions

Writing scientific software is still a task for scientists, but it can be made easier and more effective by the open-source approach. Writing maintainable, portable, extensible scientific software requires the collaboration of many IT actors and in particular of HPC centers. We cannot hope to build the software for tomorrow with yesterday's approaches.

As a final remark: writing software for use by other people requires a significant additional effort with respect to writing software for one's own usage. This comes from the need to provide user and developer documentation and to guarantee the stability of the APIs (application programming interfaces, in plain English: how developers can use the source code). The scientific community should attribute more dignity to software development and give more reward to the people doing code development.

Acknowledgments

The name ESPRESSO was proposed by Nicola Marzari while waiting for a delayed flight during the first Quantum ESPRESSO School in Beijing (2004). Quantum ESPRESSO today is the result of a collective effort by too many people to be mentioned here. I'll make an exception for a few people who have given a most significant and enduring contribution: Stefano Baroni, Stefano de Gironcoli, Carlo Cavazzoni; for the coordinator of the development in the last few years: Pietro Delugas, and for the main contributor to the current efforts on heterogeneous architectures: Pietro Bonfà.

This work is supported by EU H2020-INFRAEDI-2018-1 MaX "Materials Design at the Exascale. European Centre of Excellence in materials modelling, simulations, and design" (Grant No. 824143).

REFERENCES

1. G. Pizzi, "Discovering new materials with the computer", Il Nuovo Saggiatore **33**, n.1-2, p.53 (2017)

2. C. Cavazzoni, "Cineca - A leadership facility for Italian Computational Science", Il Nuovo Saggiatore **35**, n. 1-2, p.77 (2019)

3. P. Giannozzi *et (multis) al.*, "Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials", J. Phys.: Condens. Matter **21**, 395502 (2009)

4. P. Giannozzi *et (multis) al.*, "Advanced capabilities for materials modelling with Quantum ESPRESSO", J. Phys.: Condens. Matter **29**, 465901 (2017)